NEC

VRC4375 System Controller

Preliminary Data Sheet **October 1999**

Description

The VRC4375™ system controller is a software-configurable chip that interfaces directly with an NEC VR43xx™ 64-bit MIPS® RISC CPU and PCI bus without external logic or buffering. The system controller also interfaces with memory (SDRAM, EDO, fast-page DRAM, and flash/boot ROM) with minimal to no buffering. The memory bus can also interface with SRAM and general-purpose I/O devices. As an interface with the VR43xx CPU, the VRC4375 acts as a memory controller, DMA controller, and PCI bridge. As an interface with PCI agents, the VRC4375 acts as either a PCI bus master or a PCI bus target. Alternatively, the VRC4375 may be located on a PCI bus add-on board.

Features

- **CPU Interface**
  - Direct connection to the 66-MHz VR43xx CPU bus
  - 3.3-V I/O
  - Support for all VR43xx bus cycles
  - Little-endian or big-endian byte ordering modes
- **Memory Interface**
  - Support for boot ROM/flash memory, base memory, and up to two SIMMs
  - SIMM capacity of up to 128 MB
  - Programmable address ranges for base and SIMM memory
  - Support for 4-/16-Mb two-bank devices, 64-/128-/256-Mb 4-bank devices
  - CAS latency of 2 or 3 in base memory or SIMM SDRAM, programmable to support faster new devices or slower legacy devices
  - SIMM burst access time programmable in either 1 or 2 cycle(s)
  - 66-MHz memory bus
  - 64-MB base memory range: SDRAM and EDO DRAM
  - 256-MB SIMM memory range: SDRAM, EDO and fast-page DRAM
  - Several speed grades supported within each memory range
  - Open DRAM page maintained within base memory
  - 8-word (32-byte) write FIFO (CPU to memory)
  - 2-word (8-byte) prefetch FIFO (memory to CPU or memory to PCI)
  - On-chip DRAM and SDRAM refresh generation
  - Up to 64 MB of write-protectable boot ROM or up to 64 MB of flash ROM
  - Flash/boot ROM devices with 8-/16-/32-bit configuration support
  - Programmable timing to interface general-purpose I/O device or boot ROM in the boot ROM address range
  - Interfaceable RAM devices with programmable cycle time
  - I/O channel ready signal that throttles I/O device cycle times
  - Programmable timing for I/O channel ready signal
  - Boot ROM address and data signals driven on memory data/address bus
  - Alternative placement of boot ROM on SIMM slot 2; selectable memory size
  - 3.3-V inputs; 5-V-tolerant outputs

NEC VRC chipsets are designed for use with NEC VR Series™ microprocessors. NEC makes no claim as to the suitability of VRC chipsets for use with non-NEC microprocessors and does not warrant their performance, suitability or use in such applications.

- **PCI Interface**
  - Master and target capabilities
  - Host Bridge and Add-On Board modes
  - PCI bus arbiter with programmable arbitration scheme
  - Programmable arbitration scheme for PCI/CPU accessed to memory
  - Big-endian or little-endian byte ordering modes
  - 4-word (16-byte) bidirectional PCI master FIFO (CPU is PCI bus master)
  - 8-word (32-byte) bidirectional PCI target FIFO (memory is PCI bus target)
  - 33-MHz PCI bus clock rate
  - 132-MB/s burst transfers
  - Interrupt support for Add-On Board mode
  - 3.3-V inputs; 5-V-tolerant inputs/outputs
- **DMA Controller**
  - Four highly robust DMA channels
  - CPU-initiated block transfers between memory and PCI bus
  - 8-word (32-byte) bidirectional DMA FIFO
  - Sophisticated, programmable DMA channel arbitration priority scheme
  - Four sets of DMA Control registers for chained transfers
  - Next address pointer in each channel to support scatter/gather operation
  - Programmable DMA arbitration priority
  - Bidirectional unaligned transfers
  - Transfers at maximum PCI bandwidth of 132 MB/s
- **Interrupt Controller**
  - Nonmaskable interrupt and interrupt signals (NMI# and INT#)
  - Maskable interrupt-causing events
- **UART**
  - NY16550L Universal Asynchronous Receiver/Transmitter
  - Modem control functions
  - Separate receiver and transmitter FIFOs, 16 bytes each
  - Even-, odd- or no-parity bit generation
  - Fully prioritized interrupt control
- **Timers**
  - One 32-bit loadable watchdog timer that generates a nonmaskable interrupt
  - Two 32-bit loadable general-purpose timers that generate interrupts
  - Highly sophisticated timers with programmable clock, start/stop, auto reload/ restart, and enable/disable interrupt bits

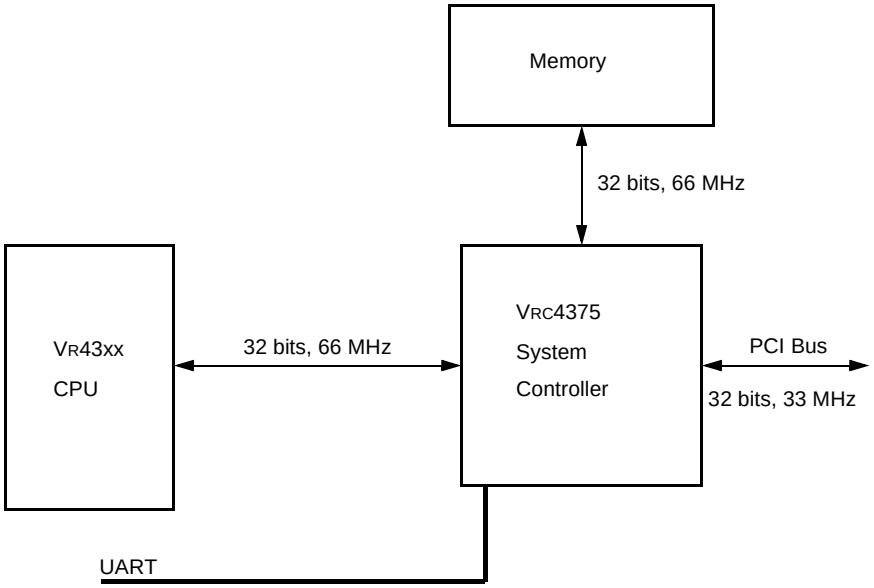
Ordering Information

Part Number	Package
VRC4375 µPD65948S1-068	256-pin TBGA

System Configuration

Figure 1 shows the controller used as a host bridge in a typical system. Alternatively, the controller can be located on a PCI bus Add-On Board.

Figure 1. System Connection



Note: F244- or F245-type buffers may be needed on the MuxAD bus and, for DIMM, on certain chip-select signals.

Terminology

In this document:

- *Signal names ending with # (such as NMI#)* are active-low signals.
- *Word* means 4 bytes. This definition of word differs from the definition in the *PCI Local Bus Specification*, where a word is 2 bytes.
- *B* means byte.
- *b* means bit.
- *CAS* means column address strobe.
- *Memory* means the local memory attached to the VRC4375 controller.
- *SIMM*™ and *DIMM*™ mean single and dual in-line memory module unless explicitly stated otherwise.
- *Module* means a set of chips, as in a SIMM or DIMM.
- *EDO DRAM* means extended data out dynamic random access memory.
- *SDRAM* means synchronous DRAM.
- *RDRAM*® means Rambus® DRAM, which is designed to conform to the interface that defines the Rambus Channel.

Reference Documents

The following documents were used in the creation of this data sheet. Unless otherwise specified, the latest version of each document applies.

- *MIPS® R4300 Preliminary RISC Processor Specification Revision 2.2* (available from MIPS Technologies, Inc.)
- *PCI Local Bus Specification Revision 2.1* and *PCI System Design Guide Revision 1.0* (available from the Peripheral Component Interconnect Special Interest Group)
- NEC *VR4300™ Microprocessor User's Manual* (document number U10504EJ6V0UM00)

Contents

1.0 Pin Configuration

Figure 2. Pin Configuration

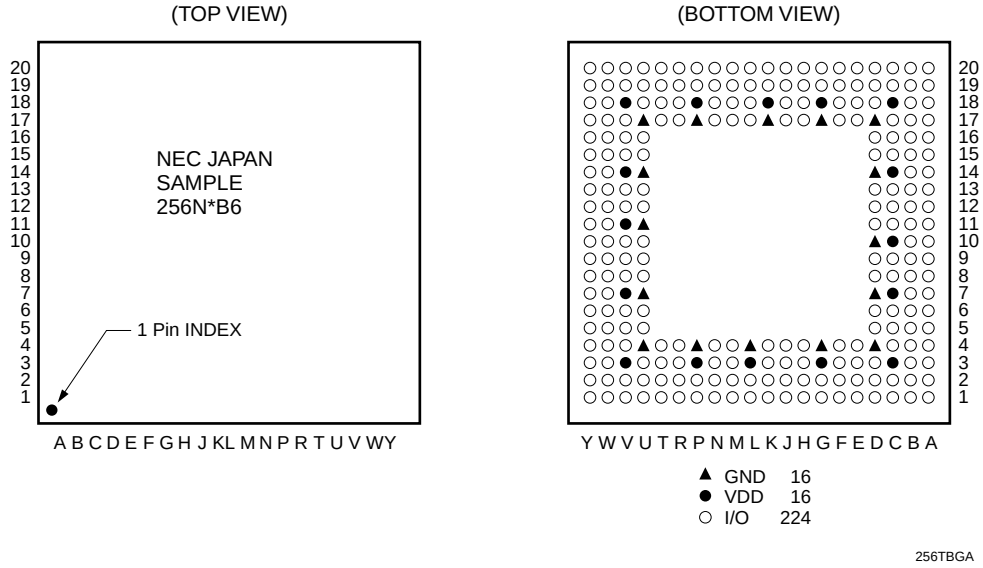


Table 1. Pin Assignment

Pin Number	Grid Number	Pin Name	Pin Number	Grid Number	Pin Name	Pin Number	Grid Number	Pin Name	Pin Number	Grid Number	Pin Name
1	A1	GND	45	P20	MuxAD[24]	89	P2	TRDY#	133	B14	MDa[26]
2	B1	CLK [0]	46	N20	ROE#	90	R2	AD[17]	134	B13	MRAS[0]#
3	C1	CLK[1]	47	M20	MuxAD[20]	91	T2	FRAME#	135	B12	MCASa[0]#
4	D1	GND[3]#	48	L20	BRAS#	92	U2	AD[18]	136	B11	GND
5	E1	INTA#	49	K20	VDD	93	V2	GND	137	B10	GND
6	F1	LOCK#	50	J20	MDa[15]	94	W2	CBE[3]#	138	B9	MuxAD[8]
7	G1	GNT[0]#	51	H20	MDa[13]	95	W3	VDD	139	B8	GND
8	H1	GND	52	G20	VDD	96	W4	AD[26]	140	B7	MuxAD[6]
9	J1	AD[1]	53	F20	MuxAD[19]	97	W5	AD[29]	141	B6	MuxAD[2]
10	K1	AD[5]	54	E20	MuxAD[18]	98	W6	SysAD[16]	142	B5	GND
11	L1	GND	55	D20	GND	99	W7	SysAD[13]	143	B4	MDa[18]
12	M1	VDD	56	C20	MuxAD[14]	100	W8	iochrdy	144	B3	MDa[2]
13	N1	AD[13]	57	B20	BBE[0]#	101	W9	SysAD[8]	145	C3	NMI#
14	P1	GND	58	A20	VDD	102	W10	GND	146	D3	GND
15	R1	DEVSEL	59	A19	MDa[23]	103	W11	SysAD[2]	147	E3	SERR#
16	T1	IRDY#	60	A18	MDa[22]	104	W12	RST#	148	F3	GNT[1]#
17	U1	GND	61	A17	MuxAD[11]	105	W13	GND	149	G3	REQ[2]#
18	V1	AD[20]	62	A16	GND	106	W14	PValid#	150	H3	AD[0]
19	W1	AD[22]	63	A15	BOE#	107	W15	SysAD[28]	151	J3	AD[3]
20	Y1	AD[21]	64	A14	MDa[27]	108	W16	SysAD[30]	152	K3	AD[6]
21	Y2	AD[23]	65	A13	MRAS[1]#	109	W17	SysAD[23]	153	L3	AD[8]
22	Y3	AD[24]	66	A12	MCASa[3]#	110	W18	GND	154	M3	AD[11]
23	Y4	GND	67	A11	MCASa[2]#	111	W19	SysAD[19]	155	N3	AD[15]
24	Y5	AD[28]	68	A10	MuxAD[9]	112	V19	VDD	156	P3	CBE[1]#
25	Y6	GND	69	A9	VDD	113	U19	SysCmd[4]	157	R3	STOP#
26	Y7	SysAD[14]	70	A8	MDa[7]	114	T19	SysCmd[2]	158	T3	CBE[2]#
27	Y8	INT#	71	A7	MDa[6]	115	R19	GND	159	U3	AD[19]
28	Y9	SysAD[9]	72	A6	MuxAD[10]	116	P19	MuxAD[25]	160	V3	AD[25]
29	Y10	SysAD[5]	73	A5	MuxAD[3]	117	N19	SDCKE[0]	161	V4	AD[27]
30	Y11	SysAD[3]	74	A4	VDD	118	M19	MuxAD[21]	162	V5	AD[30]
31	Y12	VDD	75	A3	MDa[19]	119	L19	GND	163	V6	SysAD[15]
32	Y13	EOK#	76	A2	MDa[3]	120	K19	BROMCS#	164	V7	SysAD[12]
33	Y14	GND	77	B2	GND	121	J19	GND	165	V8	SysAD[10]
34	Y15	SysAD[29]	78	C2	CLK[2]	122	H19	MDa[12]	166	V9	SysAD[7]
35	Y16	SysAD[26]	79	D2	IDSEL	123	G19	MDa[30]	167	V10	SysAD[4]
36	Y17	VDD	80	E2	REQ[3]#	124	F19	GND	168	V11	GND
37	Y18	SysAD[21]	81	F2	GNT[2]#	125	E19	MuxAD[17]	169	V12	REFCLK
38	Y19	SysAD[20]	82	G2	REQ[0]#	126	D19	MuxAD[15]	170	V13	MasterClock
39	Y20	VDD	83	H2	GND	127	C19	BBE[1]#	171	V14	SysAD[31]
40	W20	GND	84	J2	AD[2]	128	B19	GND	172	V15	SysAD[27]
41	V20	SysAD[18]	85	K2	AD[7]	129	B18	MDa[21]	173	V16	SysAD[25]
42	U20	GND	86	L2	CBE[0]#	130	B17	MuxAD[12]	174	V17	SysAD[22]
43	T20	SysCmd[3]	87	M2	AD[12]	131	B16	MDa[11]	175	V18	SysAD[17]
44	R20	UART_RxD	88	N2	AD[14]	132	B15	BWE#	176	U18	EValid#

Table 1. Pin Assignment (continued)

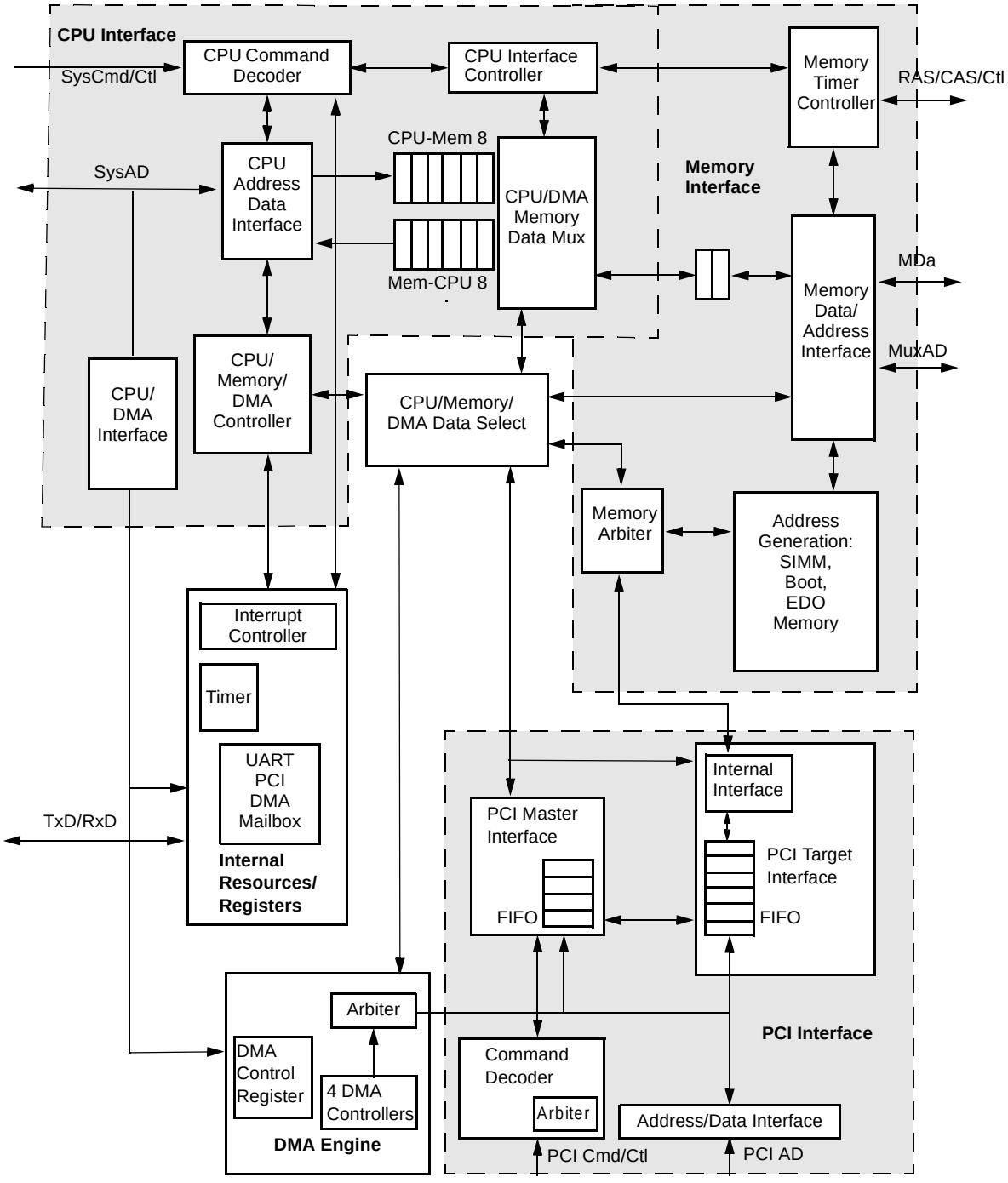
Pin Number	Grid Number	Pin Name	Pin Number	Grid Number	Pin Name	Pin Number	Grid Number	Pin Name	Pin Number	Grid Number	Pin Name
177	T18	SysCmd[1]	197	C11	MCASa[1]#	217	T4	AD[16]	237	L17	VDD
178	R18	UART_TxD	198	C10	MuxAD[7]	218	U4	GND	238	K17	SDCAS#
179	P18	UART_DSR	199	C9	MDa[5]	219	U5	AD[31]	239	J17	SDRAS#
180	N18	SDCKE[1]	200	C8	MuxAD[5]	220	U6	VDD	240	H17	GND
181	M18	MuxAD[22]	201	C7	MuxAD[1]	221	U7	SysAD[11]	241	G17	MDa[28]
182	L18	GND	202	C6	MDa[16]	222	U8	GND	242	F17	VDD
183	K18	SDCLK[1]	203	C5	MDa[17]	223	U9	SysAD[6]	243	E17	BBE[3]#
184	J18	SDCLK[0]	204	C4	MDa[1]	224	U10	VDD	244	D17	GND
185	H18	MDa[14]	205	D4	GND	225	U11	SysAD[1]	245	D16	MDa[9]
186	G18	MDa[29]	206	E4	PERR#	226	U12	SysAD[0]	246	D15	VDD
187	F18	MDa[31]	207	F4	VDD	227	U13	GND	247	D14	MDa[24]
188	E18	MuxAD[16]	208	G4	REQ[1]#	228	U14	TESTB	248	D13	GND
189	D18	BBE[2]#	209	H4	GND	229	U15	VDD	249	D12	MRAS[3]#
190	C18	MDa[20]#	210	J4	AD[4]	230	U16	SysAD[24]	250	D11	VDD
191	C17	MuxAD[13]	211	K4	VDD	231	U17	GND	251	D10	MDa[4]
192	C16	MDa[10]	212	L4	AD[9]	232	T17	SysCmd[0]	252	D9	MuxAD[4]
193	C15	MDa[8]	213	M4	AD[10]	233	R17	VDD	253	D8	GND
194	C14	MDa[25]	214	N4	GND	234	P17	UART_DTR	254	D7	MuxAD[0]
195	C13	MWE#	215	P4	PAR	235	N17	GND	255	D6	VDD
196	C12	MRAS[2]#	216	R4	VDD	236	M17	MuxAD[23]	256	D5	MDa[0]

2.0 Block Diagram

This section provides the block diagram for the system controller. For descriptions of each interface block, see:

- Section 5.0 "CPU Interface" on page 17
- Section 6.0 "Memory Interface" on page 18
- Section 7.0 "PCI Bus Interface" on page 54

Figure 3. Block Diagram



3.0 Signal Summary

The VRC4375 controller utilizes a 256-pin tape ball grid array (TBGA) package. Table 2 through Table 5 summarize the signal functions. The # symbol following a signal name indicates an active-low signal.

Table 2. CPU Interface Signals

Signal	Buffer Type (NEC Library)	I/O	Reset Value	Pull-up/ Pull-down Resistance (Ohms)	Max imum AC Load (pF)	Maximum DC Drive (mA)	Description
EOK#	B001	O	High		20	12	External ready. Signifies that the controller is capable of accepting a processor request.
EValid#	B001	O	High		20	12	External agent valid. Indicates that the controller is driving valid information on the SysAD and SysCmd buses.
INT#	B001	O	High		30	12	Interrupt request
MasterClock	B001	O	Toggle		20	12	66-MHz master clock to CPU
NMI#	B0UC	O	High	50 K pull-up	20	6	Nonmaskable interrupt; asserted when a PCI device asserts SERR# or by the Internal counter
PValid#	FIU1	I					Processor valid. Signifies that the VR43xx CPU is driving valid information on the SysAD and SysCmd buses.
SysAD[31:0]	B00C	I/O	Hi-Z		20	6	System address/data bus
SysCmd[4:0]	B00C	I/O	Hi-Z		20	6	System command/data ID bus

Table 3. Memory Interface Signals

Signal	Buffer Type (NEC Library)	I/O	Reset Value	Pull-up/ Pull-down Resistance (Ohms)	Maximum AC Load (pF)	Maximum DC Drive (mA)	Description
BOE#	B001	O	High		50	12	Base memory output enable. See Figure 4.
BRAS#	B001	O	High		50	12	Base memory row address strobe. See Figure 4.
BROMCS#	B001	O	High		30	12	Boot ROM chip select
BWE#	B001	O	High		50	12	Base memory write enable. See Figure 4.
ROE#	B001	O	High		30	12	Boot ROM/flash ROM output enable. Connect to ROM OE pin.
BBE[3:0]#	B001	O	High		30	12	Byte enable for PROM/flash ROM
MRAS[3:0]#	B001	O	High		70	12	Memory row address strobes. See Figure 5.
MCAS[3:0]#	B001	O	High		70	12	Memory column address strobes. See Figure 5.
MDa[31:0]	B00C	I/O	High		50	12	Memory data (even), boot ROM address
MuxAD[14:0]	B0D1	I/O	Hi-Z	50-K pull-down (internal)	70	12	Multiplexed row/column address; also lower boot ROM address bits
MuxAD[25:15]	B0D1	I/O	Hi-Z	50-K pull-down (internal)	50	12	Upper boot ROM address bits
MWE#	B001	O	High		30	12	Boot ROM and SIMM write enable
iochrdy	B001	I/O	High		30	12	I/O channel ready signal. Input during normal operation. See Section 6.6 for more details.
SDCAS#	B001	O	High		50	12	SDRAM column address strobe
SDRAS#	B001	O	High		50	12	SDRAM row address strobe
SDCKE[1:0]	B001	O	High		50	12	SDRAM clock enable
SDCLK[1:0]	B001	O	High		50	12	66-MHz SDRAM clock

Table 4. PCI Interface Signals

Signal	Buffer Type (NEC Library)	I/O	Reset Value	Pull-up/ Pull-down Resistance (Ohms)	Maximum AC Load (pF)	Maximum DC Drive (mA)	Description
AD[31:0]	BW01	I/O	Hi-Z		70	PCI, 12 Note	PCI multiplexed address and data bus
CBE[3:0]#	BW01	I/O	Hi-Z		50	12	PCI bus command and byte-enable
CLK[2:0]	BW01	O	Toggle		50	12	PCI clock, 33 MHz
DEVSEL#	BW01	I/O	Hi-Z		50	12	PCI device select
FRAME#	BW01	I/O	Hi-Z		50	12	PCI cycle frame
GNT[0]#	BW01	I/O	High		10	12	PCI bus grant
GNT[3:1]#	BW01	O	High		10	12	PCI bus grant
IDSEL	BW01	I					PCI initialization device select
INTA#	BW01	I/O			10	12	PCI interrupt A
IRDY#	BW01	I/O	Hi-Z		50	12	PCI initiator ready
LOCK#	BW01	I/O	Hi-Z		10	12	PCI lock atomic operation
PAR	BW01	I/O	Hi-Z		50	12	PCI parity of A/D[31:0] and C/BE[3:0]#
PERR#	BW01	I/O	Hi-Z		10	12	PCI parity error
REQ[0]#	BW01	I/O			10	12	PCI bus request
REQ[3:1]#	BW01	I					PCI bus request
RST#	BW01	I					PCI reset
SERR#	BW01	I/O	Hi-Z		10	12	PCI system error
STOP#	BW01	I/O	Hi-Z		50	12	PCI stop request from target
TRDY#	BW01	I/O	Hi-Z		50	12	PCI target ready

Note: Compatible with PCI specification.

Table 5. Utility Signals

Signal	Buffer Type (NEC Library)	I/O	Reset Value	Pull-up/ Pull-down Resistance (Ohms)	Maximum AC Load (pF)	Maximum DC Drive (mA)	Description
REFCLK	F1V1	I	Toggle				66-MHz system reference clock
UART_TXD	BWDC	I/O	Hi-Z	50 K (to GND, internal)	50	6	UART transmit data
UART_RXD	F1V1	I	Hi-Z		50	6	UART receive data
UART_DTR	BWDC	I/O	Hi-Z	50 K (to GND, internal)	50	6	UART data terminal ready
UART_DSR	F1V1	I	Hi-Z		50	6	UART data set ready
UART test/ MuxAD 15	B0D1	I	Hi-Z		50	6	UART macro test invoke
TESTB	B0D1	I	Hi-Z		50	6	Input used by UART. Pulled high for normal operation.

4.0 Registers, Resources, and Implementation

4.1 Register Summary

Table 6 summarizes the controller's register set (base memory address 0x0F00_0000 in system memory). Accesses above offset 0x1FF return 0 with the data error bit set on SysCmd[0], update the controller's Bus Error Status Register (Section 9.1.1), and cause an interrupt (INT#), if enabled.

Table 6. Register Summary

Offset from Base Memory Address 0x0F00_0000	Register Name	Size (Bytes)	CPU Bus R/W	PCI Bus (R/W)	Reference
0x0	Base Memory Control register	4	R/W	*Not accessible*	Section 6.7.1 on page 30
0x4	SIMM Memory Control Register 1	4	R/W	*Not accessible*	Section 6.8.1 on page 37
0x8	*Reserved*	4		*Not accessible*	
0xC	SIMM Memory Control Register 2	4	R/W	*Not accessible*	Section 6.8.1 on page 37
0x10	*Reserved*	4		*Not accessible*	
0x14	PCI Master Address Window Register 1	4	R/W	*Not accessible*	Section 7.3.1 on page 56
0x18	PCI Master Address Window Register 2	4	R/W	*Not accessible*	Section 7.3.1 on page 56
0x1C	PCI Target Address Window Register 1	4	R/W	*Not accessible*	Section 7.4.1 on page 58
0x20	PCI Target Address Window Register 2	4	R/W	*Not accessible*	Section 7.4.1 on page 58
0x24	PCI Master I/O Window register	4	R/W	*Not accessible*	Section 7.3.1 on page 56
0x28	PCI Configuration Data register	4	R/W	*Not accessible*	Section 7.5 on page 60
0x2C	PCI Configuration Address register	4	R/W	*Not accessible*	Section 7.5 on page 60
0x30	PCI Mailbox Register 1	4	R/W	R/W	Section 7.11 on page 71
0x34	PCI Mailbox Register 2	4	R/W	R/W	Section 7.11 on page 71
0x38	DMA Control Register 1	4	R/W	*Not accessible*	Section 8.3.1 on page 76
0x3C	DMA Memory Address Register 1	4	R/W	*Not accessible*	Section 8.3.3 on page 79
0x40	DMA PCI Address Register 1	4	R/W	*Not accessible*	Section 8.3.4 on page 80
0x44	DMA Control Register 2	4	R/W	*Not accessible*	Section 8.3.1 on page 76
0x48	DMA Memory Address Register 2	4	R/W	*Not accessible*	Section 8.3.3 on page 79
0x4C	DMA PCI Address Register 2	4	R/W	*Not accessible*	Section 8.3.4 on page 80
0x50	Bus Error Status register	4	R	*Not accessible*	Section 9.1.1 on page 84
0x54	Interrupt Control and Status Register 1	4	R/W	*Not accessible*	Section 9.1.2 on page 84
0x58	DRAM Refresh Counter register	4	R/W	*Not accessible*	Section 6.9.1 on page 43
0x5C	Boot ROM Write-Protect register	4	R/W	*Not accessible*	Section 6.5.3 on page 26
0x60	PCI Exclusive Access register	4	R/W	*Not accessible*	Section 7.12.1 on page 72
0x64	DMA Words Remaining register	4	R	*Not accessible*	Section 8.3.6 on page 81
0x68	DMA Current Memory Address register	4	R	*Not accessible*	Section 8.3.7 on page 81
0x6C	DMA Current PCI Address register	4	R	*Not accessible*	Section 8.3.8 on page 81
0x70	PCI Retry Counter	4	R	*Not accessible*	Section 7.8 on page 69
0x74	PCI Enable register	4	R/W	*Not accessible*	Section 7.10 on page 70
0x78	Power-on Memory Initialization register	4	R/W	*Not accessible*	Section 6.11.1 on page 44
0x7C	Endian Mode register (EM)	4	R/W	*Not accessible*	Section 12.0 on page 93
0x80	DMA/CPU/PCI Memory Arbiter Priority Selection register	4	R/W	*Not accessible*	Section 8.3.2 on page 78
0x84	UART Receiver Data Buffer register (UARTRBR)	4	R	*Not accessible*	Section 6.12.1 on page 46
0x84	UART Transmitter Data Holding register (UARTTHR)	4	W	*Not accessible*	Section 6.12.2 on page 46
0x88	UART Interrupt Enable register (UARTIER)	4	R/W	*Not accessible*	Section 6.12.3 on page 46
0x84	UART Divisor Latch LSB (UARTDLL)	4	R/W	*Not accessible*	Section 6.12.4 on page 47

Table 6. Register Summary (continued)

Offset from Base Memory Address 0x0F00_0000	Register Name	Size (Bytes)	CPU Bus R/W	PCI Bus (R/W)	Reference
0x88	UART Divisor Latch MSB register (UARTDLM)	4	R/W	*Not accessible*	Section 6.12.5 on page 47
0x8C	UART Interrupt ID register (UARTIIR)	4	R	*Not accessible*	Section 6.12.6 on page 47
0x8C	UART FIFO Control register (UARTFCR)	4	W	*Not accessible*	Section 6.12.7 on page 48
0x90	UART Line Control register (UARTLCR)	4	R/W	*Not accessible*	Section 6.12.8 on page 49
0x94	UART Modem Control register (UARTMCR)	4	R/W	*Not accessible*	Section 6.12.9 on page 50
0x98	UART Line Status register (UARTLSR)	4	R/W	*Not accessible*	Section 6.12.10 on page 51
0x9C	UART Modem Status register (UARTMSR)	4	R/W	*Not accessible*	Section 6.12.11 on page 52
0xA0	UART Scratch register (UARTSCR)	4	R/W	*Not accessible*	Section 6.12.12 on page 53
0xA4	DMA Control Register 3	4	R/W	*Not accessible*	Section 8.3.1 on page 76
0xA8	DMA Memory Address Register 3	4	R/W	*Not accessible*	Section 8.3.3 on page 79
0xAC	DMA PCI Address Register 3	4	R/W	*Not accessible*	Section 8.3.4 on page 80
0xB0	DMA Control Register 4	4	R/W	*Not accessible*	Section 8.3.1 on page 76
0xB4	DMA Memory Address Register 4	4	R/W	*Not accessible*	Section 8.3.3 on page 79
0xB8	DMA PCI Address Register 4	4	R/W	*Not accessible*	Section 8.3.4 on page 80
0xBC	DMA Next Record Pointer Register 1	4	R/W	*Not accessible*	Section 8.3.5 on page 80
0xC0	DMA Next Record Pointer Register 2	4	R/W	*Not accessible*	Section 8.3.5 on page 80
0xC4	DMA Next Record Pointer Register 3	4	R/W	*Not accessible*	Section 8.3.5 on page 80
0xC8	DMA Next Record Pointer Register 4	4	R/W	*Not accessible*	Section 8.3.5 on page 80
0xCC	Set Timer Counter Register 1	4	R/W	*Not accessible*	Section 9.1.3 on page 87
0xD0	Set Timer Counter Register 2	4	R/W	*Not accessible*	Section 9.1.3 on page 87
0xD4	Set NMI Timer register	4	R/W	*Not accessible*	Section 9.1.3 on page 87
0xD8	Read Timer Counter Register 1	4	R	*Not accessible*	Section 9.1.3 on page 87
0xDC	Read Timer Counter Register 2	4	R	*Not accessible*	Section 9.1.3 on page 87
0xE0	Read NMI Timer register	4	R	*Not accessible*	Section 9.1.3 on page 87
0xE4	Timers/PCI INTA# Interrupt Control and Status Register 2	4	R/W	*Not accessible*	Section 9.1.3 on page 87
0xE8	General-Purpose I/O Timing Control register	4	R/W	*Not accessible*	Section 6.6.1 on page 29
0xEC	*Reserved*				
0xF0:0xFF	*Reserved*				
0x100:0x1FF	PCI Configuration Space registers (Host Bridge mode)	1, 2, 4	R/W	*Not accessible*	Section 7.5 on page 60
0x100:0x1FF	PCI Configuration Space registers (Add-On Board mode, where the controller is located on a PCI bus board rather than on the motherboard)	1, 2, 4	*Not accessible*	R/W	Section 7.7 on page 66

4.2 Resource Accessibility

Table 7 summarizes the accessibility of the controller's internal registers, memory ranges, and PCI bus resources from the CPU and from PCI bus masters.

Table 7. Resources Accessible Through The VRC4375 System Controller

Resource	Accessible from CPU	Accessible from PCI Bus	Reference
CPU	—	No	Section 5.0 on page 17
Controller's internal registers (except PCI mailboxes)	Word	No	Section 6.0, Section 7.0, Section 8.0, Section 9.0
Boot ROM	Byte writes Word, halfword, or byte reads	No[1]	Section 6.5 on page 24
Base memory	Any CPU burst [2]	Any PCI burst[3]	Section 6.7 on page 30
SIMM memory	Any CPU burst[1]	Any PCI burst[2]	Section 6.8 on page 37
PCI mailboxes	Word	Word[4]	Section 7.11 on page 71
PCI Configuration Space registers	Word, halfword, or byte[5]	Word, halfword, or byte in Add-On Board mode only	Section 7.5 on page 60, Section 7.7 on page 66
PCI memory space	Any CPU burst of 4 words or less[1]	No	*PCI Local Bus Specification*
PCI I/O space	Any CPU burst of 4 words or less[1]	No	*PCI Local Bus Specification*
PCI configuration space	Word, halfword, or byte[5]	Word, halfword, or byte	*PCI Local Bus Specification*

Notes:
1. Because the boot ROM does not support burst transfers, it cannot be accessed from the PCI bus. The PCI interface issues cache-line reads to the target inside the controller.
2. Alignment and burst length as defined by the VR43xx CPU.
3. Any size burst length, any alignment. Burst may be disconnected by the controller.
4. The controller accepts bursts of words to the PCI mailboxes. However, the controller performs a target disconnect without data after each data transfer.
5. Any size access less than or equal to one word, aligned as defined by the VR43xx CPU.

4.3 Implementation Summary

To create a system using the VRC4375 system controller:

1. Configure the hardware using the information provided throughout this data sheet.
2. Power-up and initialize the memory, following the steps in Section 6.11 on page 44.
3. Initialize the PCI bus interface, using the configuration information provided in Section 7.0 on page 54.

5.0 CPU Interface

The controller interfaces directly with the VR43xx Series CPU, in full compliance with the *MIPS R4300 Preliminary RISC Processor Specification, Revision 2.2*. The connection is via the CPU's 66-MHz SysAD bus using a 3.3-volt I/O. All of the CPU's SysAD bus operations are supported.

5.1 Endian Configuration

The BE bit in the VR43xx CPU's Configuration register specifies the CPU's byte ordering at reset. BE = 0 configures little-endian order; BE = 1 configures big-endian order.

The BC bit in the VR43xx CPU's Configuration register specifies the PCI byte ordering at reset. BC = 0 configures little-endian order; BC = 1 configures big-endian order.

The VRC4375 controller's CPU interface supports either big- or little-endian byte ordering on the SysAD bus. The endianness for the CPU depends on the state of the MuxAD[11] signal at reset; the endianness for the PCI depends on the state of the MuxAD[12] signal at reset, as described in Section 11.0. All of the controller's other interfaces operate only in little-endian mode. All of the internal configuration, command, status and control registers are considered little endian. The software implications of this, and some related PCI device examples, are described in Section 12.0.

5.2 Data Rate Control

The controller-to-CPU data rate is determined by the EValid# signal. The CPU to controller data rate is programmable in the EP field (bits 27:24) of the CPU's Configuration register. Although the CPU supports both D and Dxx data rates, the controller only supports the D data rate.

5.3 Address Decoding

The controller latches the address on the SysAD bus. It then decodes the address and SysCmd signals to determine the transaction type. Ten address ranges can be decoded:

- Two ranges for boot ROM
- Boot ROM address ranges for I/O devices. (Do not put I/O devices in the "fault recovery address" range.)
- One range for the controller's internal Configuration registers
- One range for base memory
- Two ranges for SIMM/DIMM memory
- Two ranges for the PCI Master Address Windows
- One range for the PCI I/O address window

Boot ROM is mapped according to its size, as specified in Table 14 on page 24. The controller's internal registers are fixed at base memory address 0x0F00_0000, to allow the CPU to access them during boot, before they have been configured. All other decode ranges are programmable.

5.4 Trace Requirements

All traces between the CPU and the controller must be limited to 3 inches or less. TCLK is not used. See Section 10.0 on page 91 for details on clocking.

6.0 Memory Interface

The CPU accesses memory attached to the controller in the normal way, by addressing the system memory space. For large block transfers, the CPU can also initiate DMA transfers between memory and the PCI (bidirectionally), as described in Section 8.0. External PCI bus masters access the controller's memory through the PCI Target Address Windows, as described in Section 7.4.

The controller's memory interface has the following internal FIFOs that support transfers between memory and the various sources and destinations:

- 8-word (32-byte) write FIFO (CPU to memory)
- 2-word (8-byte) prefetch FIFO (memory to CPU or memory to PCI)
- 8-word (32-byte) bidirectional DMA FIFO (PCI to memory or memory to PCI)

6.1 Memory Regions and Devices

The controller connects directly to memory and manages the addresses, data, and control signals for the following address ranges.

- Two boot ROM ranges: standard and fault recovery
- One base memory range (programmable)
- Two SIMM memory ranges (programmable). DIMM modules can also be used.
- General-purpose I/O device within the boot ROM range

The following memory modules are examples of what can be used.

- Flash in boot ROM and/or SIMM 2 memory ranges
- EDO DRAM for base or SIMM memory; maximum of 64 MB in SIMM range using EDO DRAMs
- Synchronous DRAM (SDRAM); maximum of 256 MB in SIMM range
  - 16 Mb for base or SIMM memory (NEC part numbers µPD4516421, µPD4564841, µPD4564163, and µPD4516821)
  - 64-Mb, 4-bank devices for base or SIMM memory (NEC part numbers µPD4564441 and µPD4564841)
  - 64-Mb, 4 M x 16 devices
  - 128-Mb, 4-bank devices for base or SIMM memory (NEC part number µPD45128841)
  - 2 M x 32-bit devices in base and SIMM memory

Boot ROM can be configured with 85-ns or slower flash chips. In addition to its standard boot address range, boot ROM can also be mapped to a fault-recovery range in SIMM memory slot 2, if that slot is configured with 85-ns flash chips. Prior to accessing boot ROM, software must configure this address range, as described in Section 6.5.

Boot ROM timing is flexible; the timing is controlled by two registers as described in Section 6.6.

Base memory can include 4-Mb EDO or NEC 8-/16-Mb SDRAM chips. If SDRAM is used for base memory, it cannot be bank interleaved. Prior to accessing base memory, software must configure this address range, as described in Section 6.7.

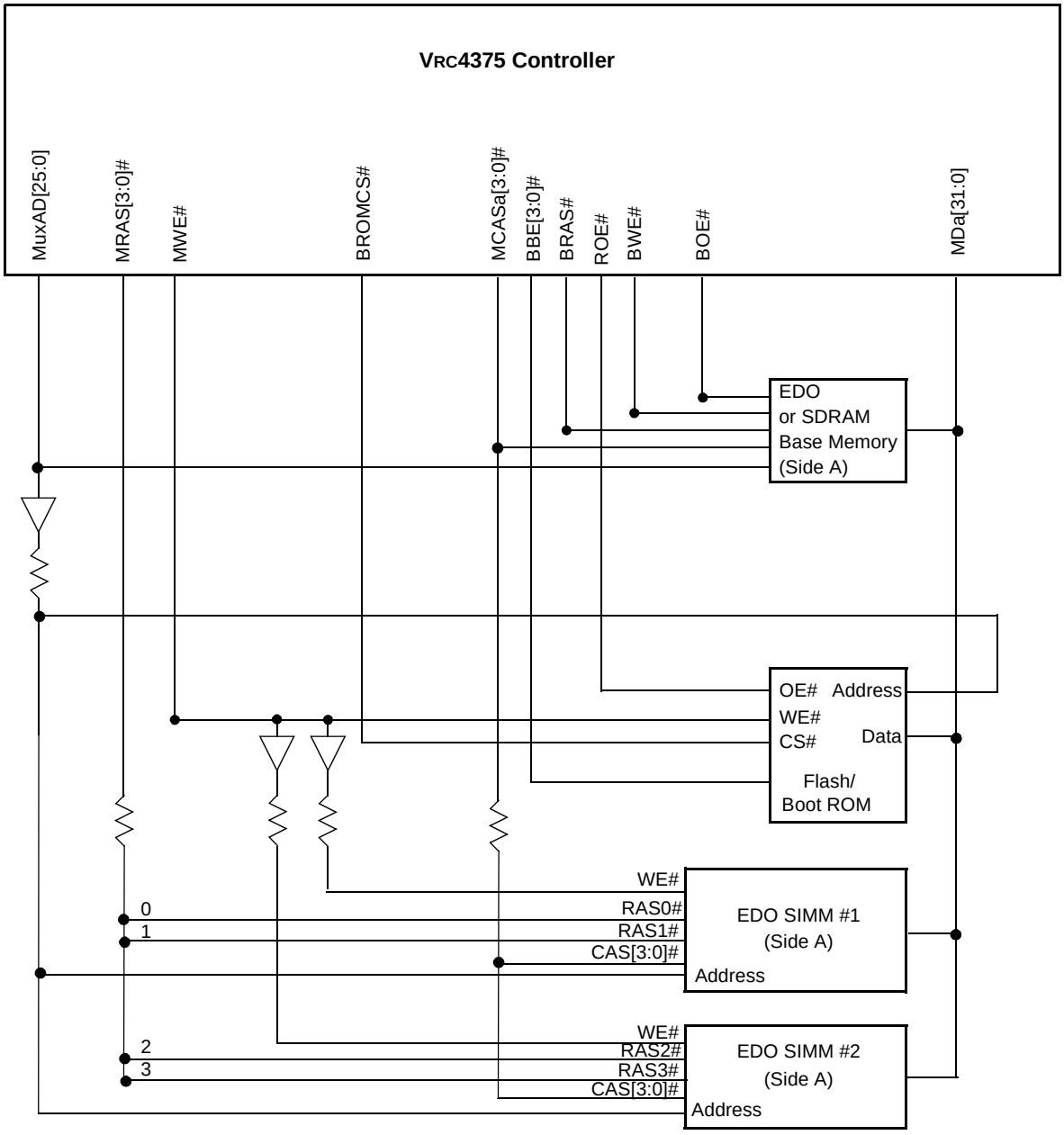
The two SIMM memory ranges can be single sided (SIMM) or double sided (DIMM), Page mode or Non-page mode, and may include any of the supported memory types.

The 100-pin DIMM package is the only DIMM package supported. Prior to accessing SIMM memory, software must configure this address range, as described in Section 6.8.

Figure 4 shows a block diagram of controller-to-memory connections for DRAM. Figure 6 on page 35 and Figure 7 on page 36 show examples of controller-to-memory connections for SDRAM.

A typical system with SDRAM in the base memory address range and SIMM memory address range is shown in Figure 4 on page 19.

Figure 4. Memory Block Diagram



6.2 Address Multiplexing Modes

The controller supports four address multiplexing modes (mux modes) in the base memory and SIMM memory ranges. Table 8 shows these modes and the row address x column address configurations they support.

Table 8. Address Multiplexing Modes

Address Multiplexing Mode	Row Address x Column Address Configurations
Mux Mode 0	9 x 9
Mux Mode 1	10 x 9, 10 x 10
Mux Mode 2	11 x 9, 11 x 10, 11 x 11
Mux Mode 3	12 x 9, 12 x 10, 12 x 11, 12 x 12
Mux Mode 4	14 x 14 (64-/128-Mb SDRAMs only)
Any mux mode	12 x 8, 11 x 8, 10 x 8

Configuration of the address multiplexing modes is done in the Base Memory Control register (Section 6.7.1) and SIMM memory control registers (Section 6.8.1). The selection of mode determines which system address bits are output from the controller on the memory interface MuxAD bus during row and column addressing.

Table 9 shows the MuxAD-to-SysAD mapping for DRAM. When EDO DRAM is used for base memory, either the 12 x 8 or 10 x 10 configuration may be used. 16-/64-/128-Mb EDO devices are supported.

Table 9. MuxAD-to-SysAD Address Mapping for EDO DRAM

MuxAD Signals	SysAD Mapping				
	Row	Column			
		Mode 0 (x9)	Mode 1 (x10)	Mode 2 (x11)	Mode 3 (x12)
0	11	3	3	3	3
1	12	4	4	4	4
2	13	5	5	5	5
3	14	6	6	6	6
4	15	7	7	7	7
5	16	8	8	8	8
6	17	9	9	9	9
7	10	2	2	2	2
8	18	19	20	21	22
9	19	20	21	22	23
10	20	21	22	23	24
11	21	22	23	24	25

Table 10 shows the MuxAD-to-SysAD mapping for SDRAM. When SDRAM is used in base memory, Mux Mode 3 must be used and 16-/64-/128-Mb SDRAM devices can be used.

Table 10. MuxAD-To-SysAD Address Mapping for SDRAM

MuxAD Signals	SysAD Mapping				
	Row	Column			
		Mode 3 (16-Mb SDRAM)	Mode 4[1] (64-/128-Mb SDRAM)	Mode 5[2] (128-Mb SDRAM)	Mode 6[3] (256-Mb SDRAM)
0	11	2	2	2	2
1	12	3	3	3	3
2	13	4	4	4	4
3	14	6	6	6	6
4	15	7	7	7	7
5	16	8	8	8	8
6	17	9	9	9	9
7	10	5	5	5	5
8	18	22	24	24	24
9	19	23	25	25	25
10	20	*Hardwired to 0*	*Hardwired to 0*	*Hardwired to 0*	*Hardwired to 0*
11	21	21	21	26	27
12	22	21	22	22	22
13	23		23	23	23
14	24		24	24	24

Notes:
1. 16-Mb SDRAM can be used in either the base memory or SIMM ranges. 64-Mb, 4-bank SDRAMs are supported only in SIMM and base memory regions. It can also drive 1 M x 32-bit memory devices.
2. Mux Mode 5 is used to drive 4-bank, 128-Mb memory devices. These devices can be organized in x4, x8, x16, and x32 configurations.
3. Mux Mode 6 is used to drive 4-bank, 256-Mb devices. These devices can be organized in x4, x8, x16, and x32 configurations.

6.3 Memory Performance

Memory access speed is determined by memory type and speed. Table 11 lists examples of 66-MHz memory bus clock cycles required for each 8-word (32-byte) CPU instruction cache line fill transfer. The first number in the “SysAD CPU Clocks (66 MHz)” column is for the first word; the remaining numbers are for the subsequent words. Only the most common combinations are shown.

Refer to more details in Section 6.7.1 and Section 6.8.1.

Read performance is calculated by counting the rising edge for TCLK, where the read command is issued by the CPU. Because the CPU issues write data with no wait states once the write command is issued, the numbers in the table represent the rate at which data is written to memory. The sum of the numbers represents the number of cycles between when the write operation was issued and when the next CPU memory operation can begin. Table 11 provides examples of memory performance.

Table 11. Examples of Memory Performance

Memory Type/ Speed	CAS Latency	Page Mode	Page Hit	R/W	Sequential Addresses	Base or SIMM Memory	SysAD CPU Clocks (66 MHz)
SDRAM, 10 ns	No, CAS = 3	No	No	R	No	Base	7-1-1-1-1-1-1-1
SDRAM, 10 ns	No, CAS = 3	No	No	W	No	Base	5-1-1-1-1-1-1-1
SDRAM, 10 ns	No, CAS = 3	No	No	R	No	SIMM	10-2-2-2-2-2-2-2 or 10-1-1-1-1-1-1-1
SDRAM, 10 ns	No, CAS = 3	No	No	W	No	SIMM	6-2-2-2-2-2-2-2 or 6-1-1-1-1-1-1-1
SDRAM, 10 ns	No, CAS = 2	No	No	R	No	Base	6-1-1-1-1-1-1-1
SDRAM, 10 ns	No, CAS = 2	No	No	W	No	Base	5-1-1-1-1-1-1-1
SDRAM, 10 ns	No, CAS = 2	No	No	R	No	SIMM	8-1-1-1-1-1-1-1
SDRAM, 10 ns	No, CAS = 2	No	No	W	No	SIMM	6-1-1-1-1-1-1-1
EDO, 60 ns	No	No	No	R		Base	9-2-2-2-2-2-2-2
EDO, 60 ns	No	No	No	W	*Not applicable*	Base	7-2-2-2-2-2-2-2
EDO, 60 ns	No	No	No	R		SIMM	11-4-4-4-4-4-4-4
EDO, 60 ns	No	No	No	W		SIMM	9-4-4-4-4-4-4-4
Fast-page, 70 ns	No	No	No	R		SIMM	11-5-5-5-5-5-5-5
Fast-page, 70 ns	No	No	No	W		SIMM	10-5-5-5-5-5-5-5
Flash/SRAM/I/O	No	No	No	R			13-7-7-7-7-7-7-7
Flash/SRAM/I/O	No	No	No	W			11-6-6-6-6-6-6-6

6.4 Placement, Loading, and Example Delays

Figure 5 shows the physical placement recommendations for motherboard chips. Table 12 shows minimum and maximum AC loadings for the memory interface signals. Table 13 shows example trace delays between memory and the controller.

Figure 5. Memory Placement

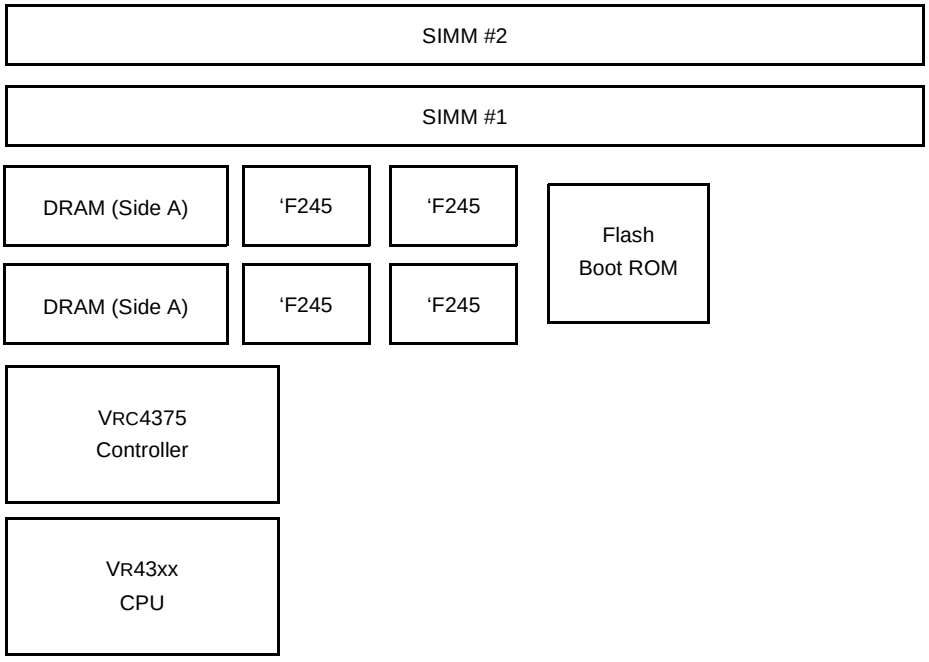


Table 12. Memory Signal AC Loading

Signal	Min. (pF)	Max. (pF)	Description
MuxAD[25:15]	10	50	ROM address upper bits
MuxAD[14:00]	10	70	Multiplexed row/column address, ROM address low bits
BRAS#	5	50	Base memory row address strobe
MRAS[3:0]#	15	70	SIMM memory row address strobes
MCASa[3:0]#	20	70	Column address strobe, even addresses
MWE#	10	30	Boot ROM and SIMM write enable
BOE#	10	50	Base memory output enable
ROE#	10	30	ROM/flash output enable
BWE#	10	50	Base memory write enable
BROMCS#	10	30	Boot ROM chip select
MDa[31:0]	20	70	Memory data, boot ROM data
SDCLK[1:0]	10	50	66-MHz SDRAM clock
SDCKE[1:0]	10	70	SDRAM clock enable
SDCS[1:0]#	10	50	SDRAM command select
SDCAS#	20	50	SDRAM column address strobe
SDRAS#	20	50	SDRAM row address strobe

Table 13. Example SIMM DRAM Delays

Signal	Source	Destination	Delay Subtotal (120 ps/inch)	RC Delay (2 x R x C)	Total Delay[1]
MCASa[3:0]#	Controller	SIMM DRAM	1.0 ns (8 in)	2 x 33 x 100 pF = 6.6 ns	7.6 ns
MuxAD[14:0][2]	Controller Buffer input Buffer output	Buffer Buffer output (100 pF) SIMM DRAM	0.5 ns 14.0 ns 2.0 ns	 2 x 10 x 250 pF = 5.0 ns	21.5 ns
MDa[31:0]	Controller	SIMM DRAM	2.0 ns		2.0 ns
MDa[31:0]	SIMM DRAM	Controller	2.0 ns		2.0 ns
MRAS#	Controller	SIMM DRAM	2.2 ns	2 x 33 x 180 pF = 12 ns	14.2 ns
MWE# [2]	Controller Buffer input Buffer output	Buffer Buffer output (130 pF) SIMM DRAM	1.3 ns 11.0 ns 2.0 ns	 2 x 10 x 130 pF = 2.6 ns	16.9 ns
Notes: **1.** From controller to SIMM DRAM. **2.** To accomodate SIMM loading, F244 or F245 buffers must be used on these signals.					

6.5 Boot ROM

Boot ROM can be configured with 85-ns flash chips; access time should be 200 ns or less. The controller supports 8-/16-/32-bit boot ROM at locations 0x1C00_0000 through 0x1FFF_FFFF in the system memory space. In addition to this standard boot-address range, boot ROM can also be mapped to a fault-recovery range in SIMM memory slot 2, if that slot is configured with 85-ns flash chips. The boot ROM does not support CPU cache operations—it can not operate in burst mode—only during one address/data cycle at a time. It is not accessible from the PCI bus.

During boot ROM accesses, MuxAD[25:0] provide a 26-bit address. This allows a user to place up to 64 MB of boot ROM on the motherboard. A user can also place up to 16 MB of boot ROM on the board and an additional 48 MB on the SIMM slot 2, for a total of 64 MB. SIMM slot 2 is used to boot from in Fault-Recovery mode (Section 6.5.6 on page 27). The boot ROM size and boot base memory address are configured at the rising edge of RST# by the state of MuxAD[2:0], as shown in Table 14.

Write and read cycles may be in word, halfword, or byte sizes. During a read operation, the controller assembles four consecutive byte read cycles into words. The boot ROM data is connected to MDa[31:0].

Table 14. Boot ROM Size Configuration at Reset

MuxAD[2:0]	Boot ROM Size	Address Range
000	0.5 MB	0x1FC0_0000 through 0x1FC7_FFFF
001	1.0 MB	0x1FC0_0000 through 0x1FCF_FFFF
010	2 MB	0x1FC0_0000 through 0x1FDF_FFFF
011	4 MB	0x1FC0_0000 through 0x1FFF_FFFF
100	8 MB	0x1F80_0000 through 0x1FFF_FFFF
101	16 MB	0x1F00_0000 through 0x1FFF_FFFF
110	32 MB	0x1E00_0000 through 0x1FFF_FFFF
111	64 MB	0x1C00_0000 through 0x1FFF_FFFF

The controller asserts the boot ROM chip select (BROMCS#) in the address range 0x1C00_0000 through 0x1FFF_FFFF. When write cycles are performed to the boot ROM/flash memory space, the controller asserts MWE# in conjunction with BROMCS#. When read cycles are performed, the controller asserts BBE# and ROE# in conjunction with BROMCS#.

If the CPU attempts to access boot ROM addresses outside the defined size of the boot ROM, the controller returns 0 with the data error bit set on SysCmd[0]. In addition, the controller's Bus Error Status register (Section 9.1.1) is updated and an interrupt is asserted on the INT# signal, if the interrupt is enabled in the Interrupt Control and Status register (Section 9.1.2).

6.5.1 Boot ROM Write Protection

Boot ROM can be protected in hardware and/or software. Hardware protection is implemented at boot time. Software protection is implemented by programming the Boot ROM Write-Protect register (Section 6.5.3).

6.5.2 Hardware versus Software Protection

Hardware can implement write protection on up to 960 KB of the boot ROM, in blocks of 64 KB at boot time. On the rising edge of reset (RST#), four bits of the MuxAD bus, MuxAD[7:3], are sampled and used to determine the number of 64-KB blocks to be protected. Up to 15 blocks can be protected; 0000 indicates one 64-KB block, 0001 indicates two blocks, and so on. A value of 1111 indicates that no blocks are to be protected.

For boot ROM sizes less than or equal to 4 MB, the base memory address of the ROM and the base memory address for hardware protection are both 0x1FC0_0000. For boot ROM sizes of 8 MB, 16 MB, and 64 MB, the base memory addresses of ROM are 0x1F80_0000, 0x1F00_0000, and 0x1C00_0000, respectively. The base memory address for hardware protection is 0x1FC0_0000 when boot ROM size is less than or equal to 4 MB; 0x1F80_0000 when boot ROM size is equal to 8 MB; 0x1F00_0000 when boot ROM size is equal to 16 MB; 0x1E00_0000 when boot ROM size is equal to 32 MB; and 0x1C00_0000 when boot ROM size is equal to 64 MB, as shown in Table 14.

Software can also implement write protection by writing to the Boot ROM Write-Protect register. The 7 least-significant bits of this register provide additional protection for up to 127 64-KB blocks, totaling 1984 KB. The base memory address for software protection is the same as for hardware protection (Table 14); the protected range consists of a combination of software-implemented protection and hardware-implemented protection. Software can override or re-enable hardware write protection as described in the Boot ROM Write-Protect register's key field (Section 6.5.3).

6.5.3 Boot ROM Write-Protect Register

The Boot ROM Write-Protect register stores a word at offset 0x5C. The register is initialized to 0xFFFF_FF7F at reset. Software can override the hardware write protection configured at reset by writing 0xC0DE_99xx to this register, with the least-significant 7 bits containing the desired software write protection value. Care must be taken not to change the protection while data is being written to the boot ROM. To prevent this, read a word from the boot ROM immediately before changing this register.

The register has the following fields:

Bits 6:0	WProt	*Write protection value* The number of 64-KB blocks, minus 1, to be write protected (up to 127 blocks). 0 = Protects 1 block All 1s = Disables protection
Bit 7	Reserved	*Hardwired to 0*
Bits 31:8	Key	*Hardware protection override key* 0xC0DE_99 = Override the hardware protection configured at reset To re-enable hardware protection, software must write a value other than 0xC0DE_99 to this field. After re-enabling hardware protection, the key field changes to all 1s.

6.5.4 Boot ROM Protection Ranges

The boot ROM protection ranges as defined by the protection bits are as follows.

Table 15. Boot ROM Protection Ranges

MuxAD [2:0]	Boot ROM Size	Protection Ranges
000	0.5 MB	Start at 0x1fc0_0000 + 6-bit protection value x 64 KB
001	1 MB	Start at 0x1fc0_0000 + 6-bit protection value x 64 KB
010	2 MB	Start at 0x1fc0_0000 + 6-bit protection value x 64 KB
011	4 MB	Start at 0x1fc0_0000 + 6-bit protection value x 64 KB
100	8 MB	Start at 0x1f 80_0000 + 7-bit protection value x 64 KB
101	16 MB	Start at 0x1f00_0000 + 7-bit protection value x 64 KB
110	32 MB	Start at 0x1e00_0000 + 7-bit protection value x 64 KB
111	64 MB	Start at 0x1c00_0000 + 7-bit protection value x 64 KB

The boot ROM may be configured with two distinct address spaces: standard and fault recovery. The standard address space is used for normal operation. The fault-recovery space is used when the standard flash ROM is corrupted or requires updating.

6.5.5 Standard Space

The standard boot ROM address space consists of a hardware-protected boot block and a software-protected address range (Section 6.5.1). Table 14 shows the boot ROM memory map during standard operation for a 1-MB ROM.

Table 16. Standard Mode Example: 1-MB Boot ROM Memory Map

A[31:28]	Address A[27:0]	Size	Description
1	E00_0000 to FBF_FFFF	28 MB	*Unused*
	FC0_0000 to FC0_FFFF	64 KB	Boot block, hardware protected against writes
	FC1 0000 to FCF_FFFF	1986 KB	Regular flash ROM, software protected only
	FD0_0000 to FFF_FFFF	3 MB	*Unused*

6.5.6 SIMM Slot 2 Fault-Recovery Space

The fault-recovery boot ROM address space can be accessed when the standard flash/boot ROM is corrupted or requires updating. The controller may be configured to boot from SIMM slot 2 instead of from boot ROM. This is done by driving MuxAD[8] high during reset. When the controller detects that this signal is high on the rising edge of RST#, the boot ROM address range (0x1C00_0000 through 0x1FFF_FFFF) is mapped into SIMM slot 2. The actual starting address and range depends on the boot ROM size selected per Table 15. When booting the system in this manner, the controller assumes that SIMM slot 2 is configured with an 85-ns flash SIMM. When booting from SIMM slot 2, the boot ROM will be forced into address 0x1E00_0000, 0x1E80_0000, or 0x1C00_0000, depending on its size.

Table 17 shows the boot ROM memory map during Fault-Recovery Mode 1 operation. Table 18 shows the required values in SIMM Memory Control Register 2 for booting from SIMM 2. In this mode, SIMM Memory Control Register 2 returns the value 0x1F00_100E when read. The hardware protection for the boot ROM remains in place even when the ROM is remapped for booting from SIMM 2.

Table 17. Fault Recovery Mode Example: 1-MB Boot ROM Memory Map

A[31:28]	Address A[27:0]	Size	Description
1	E00_0000 to EBF_FFFF	12 MB	*Unused*
	EC0_0000 to EC0_FFFF	64 KB	Boot block alternate address, H/W protected against writes
	EC1 0000 to ECF_FFFF	960 KB	Regular flash ROM alternate address, S/W protected only
	ED0_0000 to EFF_FFFF	3 MB	*Unused*
	F00_0000 to FFF_FFFF E00_0000 to FFF-FFFF C00_0000 to FFF_FFFF	16 MB 32 MB 64 MB	SIMM slot 2, configured for 85-ns flash, double-sided SIMM slot 2, configured for 85-ns flash, double-sided SIMM slot 2 configured for 85-ns flash, double-sided

Table 18. SIMM Memory Control Register 2 Values for Booting from SIMM Slot 2

Register Field	Value
Memory type	2 = Flash
Number of sides	1 = Double-sided
SIMM memory enable	1 = Enabled
Address Multiplexing mode	b100 = 14 x 11 (Mux Mode 4; bit 12 = 1, bit 5 = 0; bit 4 = 0)
EDO Identification mode	0 = Normal mode
MDa bit 31 during EDO ID	0 = Read only
Bank interleaving	0 = Non-bank interleaved
Physical address mask	0x00 = Size of 64 MB
SIMM memory base memory address[Note]	0x78 = Base memory address 0x0f00_0000
Note: When booting from SIMM mode, MuxAD[8] must be set to one to access SIMM2.	

6.6 Programmable ROM/PROM or General-Purpose I/O Timing Interface

The boot ROM interface timing is programmable. This can be used to connect a slow PROM, flash memory, or I/O device. The cycle time is controlled by the General-Purpose I/O Timing Control register at offset 0xE8. The I/O channel ready input signal (iochrdy) is optionally used to sense when an I/O device is ready with data so the controller can terminate the cycle. If a device has predictable timing, this signal can be pulled high.

6.6.1 General-Purpose I/O Timing Control Register

The General Purpose I/O Timing register lengthens the write and read cycle times of ROM, PROM, or RAM devices of various speed grades that can be placed on the memory bus. It can be used to interface with an I/O device as well. The address is mapped in the 64-MB allowable boot ROM address space; care must be taken in assigning boot block protection ranges to the I/O device. If a write operation is issued to an I/O device that is erroneously mapped in the boot ROM write protection range, the cycle will terminate. When interfacing with an I/O device, memory addressing and a data bus move data with PROM control signals. When interfacing to a SRAM device, the timing can be selected per the device's cycle time. Care must be taken in mapping this device so that no writes occur in the boot ROM protected area. (Section 13.0 contains timing diagrams.)

The iochrdy signal can be used to throttle the read or write cycle timing in addition to programmable timing. The sampling of the iochrdy signal or the duration of minimum cycle times per their respective bit field values are enabled only when bit 4 is set.

The register is at offset 0xE8 in the system memory space. The CPU has read and write access.

Bits 3:0	Read timing control	*Cycle time control for read operations* 0000 = 1 clock cycle 1111 = 16 clock cycles
Bit 4	Enable	*Enables activation of the programmable read and write cycle times*
Bits 8:5	Write timing control	*Cycle time control for write operations* 0000 = 1 clock cycle 1111 = 16 clock cycles
Bit 9	I/ODevErr	Indicates that an I/O device did not respond with iochrdy within the maximum number of clocks programmed by bits 25:10 of this register. This case assumes that the iochrdy signal was driven low by the device and the controller is waiting for it to go high. This feature is intended for devices that can be unpredictably slow and the memory bus is allowed to be relinquished after a controlled number of cycles.
Bits 25:10	I/OChMaxTm	Determines the maximum number of cycles the controller will wait for the device to respond with the iochrdy signal before issuing an error and setting bit 9. The value could be programmed from 0000 = 1 to FFFF = 64 K SysClk cycles. In the event that the counter counts down to zero from its loaded value, the cycle will terminate with read data 0000,0000 h. The counter is enabled by bit 4 of the I/O Timing Control register. The default value of the counter will be 0000,000F. This indicates a cycle time of 15 clocks.
Bits 31:26	Unused	*Hardwired to zero*

6.7 Base Memory

The controller supports up to 64 MB of base memory. This memory, if installed, must include one of the following memory devices.

- 4-Mb EDO (60 ns)
- 16-Mb SDRAM (10 ns)
- 64-Mb SDRAM (10 ns)
- 128-Mb SDRAM (10 ns)

To allow pages to remain open between accesses to base memory, the controller can perform page comparisons on addresses in the base memory range.

See “Note” in Section 6.7.6 on page 34 for an anomaly in the design and its work-around.

6.7.1 Base Memory Control Register

The Base Memory Control register configures base memory. At reset, all bits in this register are set to 0, and this setting must not be changed during any other type of access (by the CPU, DMA, or PCI bus) to base memory. If base memory is enabled, software should perform a read immediately before writing to this register, because write cycles are posted in the controller’s write FIFO, and a read cycle will force the controller to write back the FIFO contents before servicing the read. For reference, a value of 0x0000803B is programmed for an SDRAM, Mux Mode 3, 8-MB memory.

The Base Memory Control register stores a word at offset 0x0 in the system memory space. It has the following fields:

Bits 1:0	Type	*Memory type* 0 = EDO 1 = Invalid, could cause controller to hang 2 = Invalid, could cause controller to hang 3 = SDRAM (16-/64-/128-Mb devices)
Bit 2	DlyEValid	When set, in response to a CPU command, this bit delays the EValid signal by one clock cycle. The default value is 0.
Bit 3	En	*Base memory enable* 1 = Enables base memory 0 = Disables base memory
Bits 5:4	MuxMode	*Address Multiplexing mode* 00 = Mux Mode 0 01 = Mux Mode 1 10 = Mux Mode 2 11 = Mux Mode 3 See Table 9 and Table 10 for descriptions of these modes. Only Mode 3 is allowed for SDRAM devices. When 64-Mb or larger devices are used, Mux Mode 4 is selected (bit 12 = 1); in this case, bits 5:4 are ignored.
Bit 6	Reserved	*Hardwired to 0*

Bit 7	PM	*Page mode* 1 = Enables Page mode When enabled, accesses to base memory leave a memory page open (MRAS# asserted) at the end of the cycle. Page mode can only be used when base memory is configured with EDO DRAM; it cannot be used with SDRAM. 0 = Disables Page mode Accessing the same memory with address bit 28 set to 0 will cause the controller to close the page at the end of the cycle. When disabled, accesses to base memory close the memory page (MRAS# negated) at the end of the cycle.
Bit 8	CASLAT	*CAS latency selection* This bit sets the CAS latency in the SDRAM mode register, where CAS latency is defined as either two or three clock cycles. CAS latency is set during power-up initialization of the SDRAM Mode register setting. The latency is programmable to optimize the performance of faster memory devices. The default value of this bit is 0. 0 = 3 cycles 1 = 2 cycles
Bits 11:9	Reserved	*Hardwired to 0*
Bit 12	MuxMD4	*Mux Mode 4*
Bit 13	Reserved	*Hardwired to 0*
Bits 17:14	Mask	*Physical address mask* Determines the size of base memory by masking off address bits from the address comparison, beginning with bit 22. Thus, bits 25:22 of the physical address may be masked, providing an address space between 4 MB (no bits masked) and 64 MB (4 bits masked). Masks must be a pattern of left-justified 1s or 0s. A 1 in the Mask field indicates that the corresponding address bit is not masked. Address bit 28 is not used in the address comparison; it is only used when Page mode is enabled. Addresses with address bit 28 cleared to 0 or set to 1 alias to one another.

Mask bit = 0: bit masked. See the example below.

Mask Bit Setting Examples

Mask Bits Set				Address Bits Decoded						BRAS# Selected	BOE# Selected	Total Memory Selected
17	16	15	14	27	26	25	24	23	22			
1	1	1	1	D	D	D	D	D	D	2 MB	2 MB	4 MB
1	1	1	0	D	D	D	D	D	X	4 MB	4 MB	8 MB
1	1	0	0	D	D	D	D	X	X	8 MB	8 MB	16 MB
1	0	0	0	D	D	D	X	X	X	16 MB	16 MB	32 MB
0	0	0	0	D	D	X	X	X	X	32 MB	32 MB	64 MB

Note: D = Decoded; X = Not decoded

Bits 21:18 Reserved — *Hardwired to 0*

Bits 27:22 BaseAdd — *Base memory address*
Compared with bits 27:22 of the physical address. A match indicates that the access is to base memory. Base memory must not be overlapped with any other resource in the system.

Bits 31:28 Reserved — *Hardwired to 0*

6.7.2 Base Memory Page Mode

The controller can maintain an open memory page (MRAS# negated) within the base memory range. Page mode is enabled by the PM bit (7) in the Base Memory Control register. Page mode can only be used when base memory is configured with EDO DRAM; it cannot be used with SDRAM.

When enabled, Page mode becomes active during accesses to base memory in which address bit 28 is set to 1. If a page is currently open from the previous access, the controller performs an address comparison to determine if the current access is within the same page. If so, access is performed using the EDO page-hit timing (MCAS# only). If the access is not within the same page (a page miss), the MRAS# signal is precharged and a normal memory access occurs. The controller then holds the page open at the end of the cycle.

When the base memory is accessed with address bit 28 cleared to 0, Page mode is not used. If a page is currently being held open, and the current access is a hit, a normal EDO page-hit cycle is performed. At the end of the cycle, the page is closed. If the access is a miss, MRAS# is precharged and the page is closed at the end of the cycle.

Because address bit 28 is not used to decode base memory addresses, base memory is aliased to two ranges, 256 MB apart. No other system resources may be placed in this address range, whether or not Page mode is enabled.

6.7.3 Base Memory Prefetching

When Page mode is enabled for base memory (bit 7 set in the Base Memory Control register), the controller automatically prefetches two words from base memory into a 2-word prefetch FIFO. That is, after each read, the controller prefetches two additional words into its internal prefetch FIFO. If the processor subsequently attempts a read from an address immediately following (sequential to) the address of the last read cycle, the first two words will be supplied from the prefetch FIFO. Table 19 shows the words that will be placed in the prefetch FIFO following various types of base memory read cycles.

Table 19. Prefetch FIFO Contents versus Read Sizes and Alignment

Start Address (Word Address)	One-Word Access		Two-Word Access		Four-Word Access		Eight-Word Access	
	Word(s) to CPU	Prefetched Word(s)	Word(s) to CPU	Prefetched Word(s)	Word(s) to CPU	Prefetched Word(s)	Words to CPU	Prefetched Word(s)
0	0	1	0, 1	2, 3	0, 1, 2, 3	4, 5	0, 7	8, 9
1	1	2, 3	Not used Note	Not usedNote	Not usedNote	Not usedNote	Not usedNote	Not usedNote
2	2	3	2, 3	4, 5	2, 3, 0, 1	4, 5	Not usedNote	Not usedNote
3	3	4, 5	Not used Note	Not usedNote	Not usedNote	Not usedNote	Not usedNote	Not usedNote
4	4	5	4, 5	6, 7	4, 5, 6, 7	8, 9	Not usedNote	Not usedNote
5	5	6, 7	Not used Note	Not usedNote	Not usedNote	Not usedNote	Not usedNote	Not usedNote
Note: Access is illegal because of the alignment.								

The controller compares the current SysAD address with the previous address to determine if the access is sequential. If sequential, the data will be available to the CPU three SysAD clocks prior to the nonsequential case. Prefetched words are retained in the prefetch FIFO if accesses to resources other than base memory are performed between base memory accesses. Writes to base memory invalidate the prefetched words and force a sequential miss.

6.7.4 SDRAM in Base Memory

The base memory space can be configured with either 16- or 64-Mb SDRAM or 4-Mb EDO modules. This section describes the SDRAM option.

6.7.5 SDRAM Device Configurations

The controller supports the following (but is not limited to) 16- or 64-Mb NEC SDRAM parts and configurations in base memory. (Figure 6 and Figure 7 show examples of SDRAM connections.)

- 512 K x 32 (16-Mb) chips, 2 banks of 2 K rows, 256 columns (NEC # µPD4516161)
- 1 M x 16 (16-Mb) chips, 2 banks of 2 K rows, 256 columns (NEC # µPD4516821)
- 2 M x 8 (16-Mb) chips, 2 banks of 2 K rows, 512 columns (NEC # µPD4516421)
- 4 M x 16 (64-Mb) chips (NEC # µPD456163)
- 2 M x 32 (64-Mb) chips (NEC # µPD454323)
- 8 M x 8 (64-Mb) chips, 4 banks of 4 K rows, 512 columns (NEC # µPD4564841)
- 16 M x 8 (128-Mb) chips, 4 banks of 4 K rows, 1 K columns (NEC # µPD45128841)

Table 20 shows some of the SDRAM configurations supported for base memory. Table 21 shows SDRAM configurations that are not supported in base memory (but may be supported in SIMM memory).

Table 20. Base Memory SDRAM Configurations Supported

Memory Size	SysAD Address Bits Required	Banks	Sides	SysAD[25:22] Mask Bits	NEC Part Number
4 MB	21:0	2, 1 M x 16	Single	1111	μPD4516821G5-A10
8 MB	22:0	4, 2 M x 8	Single	1110	μPD4516421G5-A10
16 MB	23:0	2, 4 M x 16	Single	1100	μPD4564163-A10
32 MB	25:0	4, 8 M x 8	Single	0000	μPD4564841
64 MB	25:0	8, 16 M x 4	Single	0000	μPD4564441

Table 21. Base-Memory SDRAM Configuration Not Supported

Memory Size	SysAD Address Bits Required	Banks	Sides	SysAD[25:21] Mask Bits	NEC Part Number
16 MB	23:0	8, 4 M x 4	Single	1100	Not supported in base memory

6.7.6 SDRAM Signal Connection Example

Figure 7 shows how the NEC 16-Mb SDRAMs are connected for base memory. The bank uses BRAS# and BOE# for chip selects, MCASa[3:0]# for the data I/O masks (DQMs), and MDa[31:0] for data. Banks share the same SDRAS#, SDCAS#, and BWE# signals. The bank must be programmed as *non-bank interleaved, non-page mode*. To do this, clear bit 7 in the Base Memory Control register (otherwise bit 7 has unpredictable results).

Note: There is an anomaly in the Revision 2 design for SDRAM base memory configuration.

The following two methods can be used as a workaround to access 16 MB of address space:

- Configure the control register for 32 MB of address space. In this case, BRAS# will be valid for up to 16-MB addresses.
- Configure the control register for 16 MB of address spaces; use signal BRAS# to access the lower 8 MB, and use signal BOE# to access the upper 8 MB.

Example: Control/word for 8 devices, 2 M x 8 organization = 0003_003B Hx.

Figure 6. SDRAM Connections for Base Memory (Example)

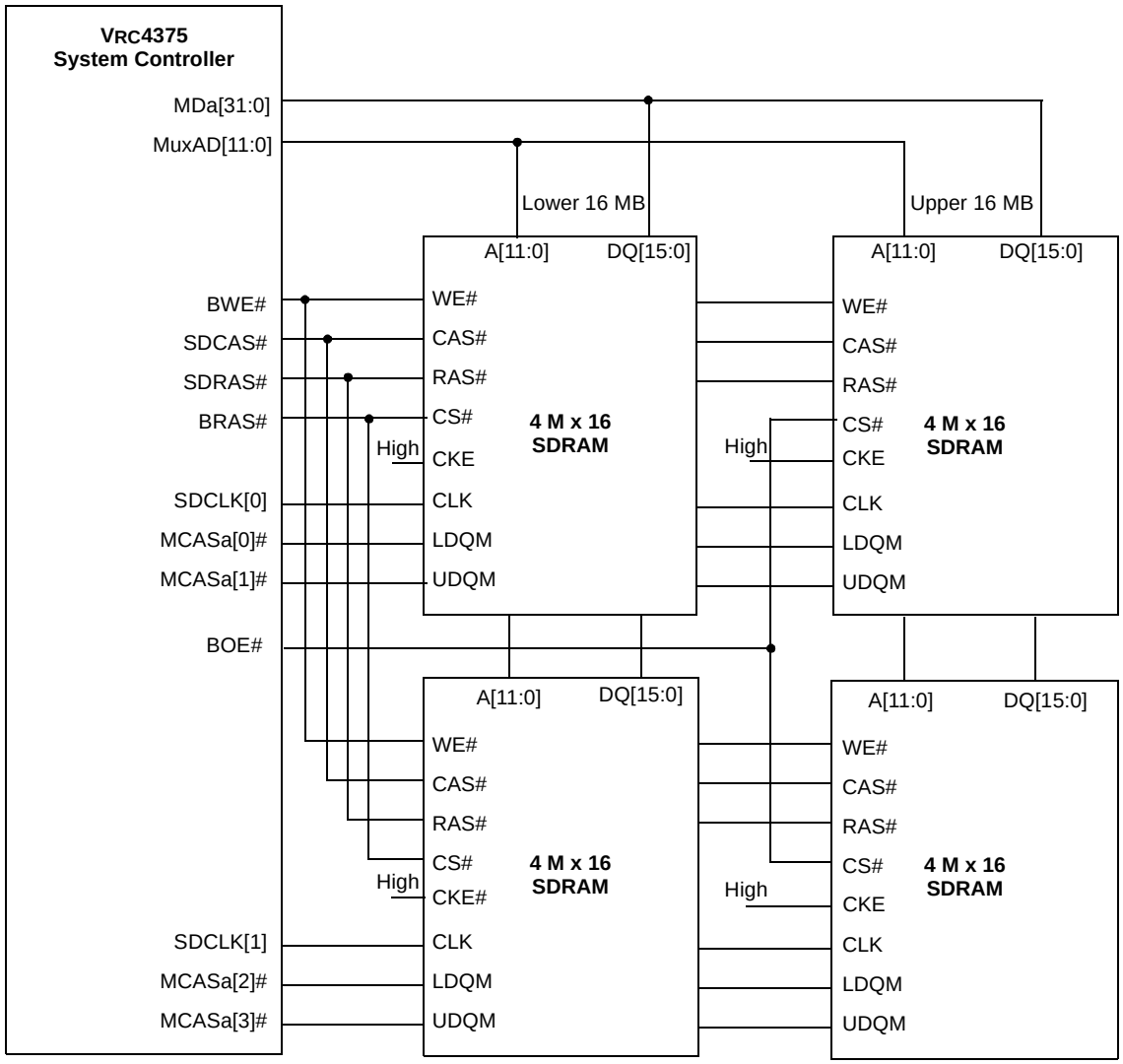
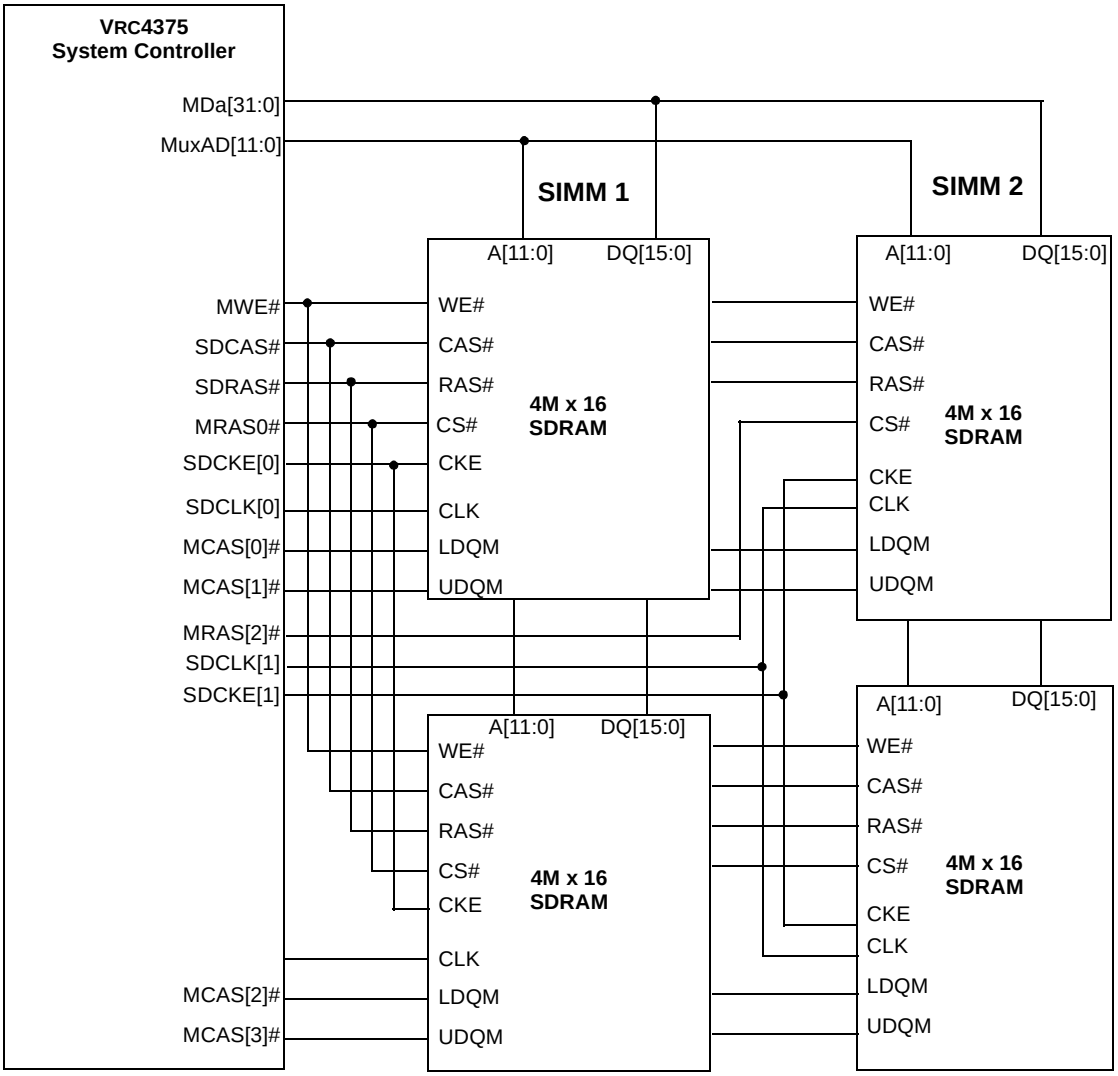


Figure 7. SDRAM Connections for SIMM 1 and SIMM 2 (Example)



6.7.7

SDRAM Banks and Burst Types

The terms *bank* and *burst type* have multiple meanings in memory design using SDRAM chips.

- Banks referenced with respect to memory modules differ from banks inside an SDRAM. Memory modules are external and two or more form a bank. For SDRAMs, MuxAD[11] serves as the bank select for all chips on a module.
- The burst type of a single SDRAM chip is programmed in the chip's mode register to be either *interleaved* or *sequential*. This concept relates only to the *word order* in which data is read into and written out of the SDRAM chip. The concept does not relate to the number of words transferred in a given clock cycle. The burst type for all SDRAM chips attached to the controller is configured during memory initialization (Section 6.11).

6.7.8 SDRAM Word Ordering

Table 22 shows the word address order for an 8-word instruction cache line fill from SDRAM. This order is determined by the SDRAM chips' burst type, which is programmed during memory initialization (Section 6.11). The controller programs the burst type and word order in the same way for all SDRAM chips connected to it (in both SIMM and base memory ranges).

Table 22. SDRAM Word Order for Instruction Cache Line Fill

Word Address A[4:2]	SDRAM Chip Burst Type	
	Sequential[Note]	Interleaved
000	Not supported	0-1-2-3-4-5-6-7
001	Not supported	1-0-3-2-5-4-7-6
010	Not supported	2-3-0-1-6-7-4-5
011	Not supported	3-2-1-0-7-6-5-4
100	Not supported	4-5-6-7-0-1-2-3
101	Not supported	5-4-7-6-1-0-3-2
110	Not supported	6-7-4-5-2-3- 0-1
111	Not supported	7- 6- 5- 4-3- 1-0

Note: The controller does not support sequential bursts for SDRAMs. It assumes that all SDRAMs are initialized to the interleaved bursts, using a burst length of 8 words.

The controller initializes any SDRAM chips in the 8-word burst length mode. Because of this, the controller always reads 8 words from SDRAM, regardless of the data width requested. However, the controller only returns the requested number of words to the CPU or PCI bus master; remaining words are either stored in the controller's internal base memory prefetch FIFO (Section 6.7.3) or discarded. The CPU or PCI bus master can write any number of bytes, and the controller automatically issues a burst stop or precharge termination command to the SDRAM in order to store only the correct number of bytes in the SDRAM.

6.8 SIMM/DIMM

The controller supports four programmable address ranges for independently controlled 72-pin SIMM or DIMM. SIMM slots 1 and 2 are connected to the MDa bus. The two SIMM slots may be configured with SDRAM, EDO DRAM, fast-page DRAM, or flash memory modules. To accommodate SIMM loading, F244 or F245 buffers must be used on the controller's MuxAD signals.

6.8.1 SIMM Memory Control Registers

The configuration of each SIMM is controlled by its own SIMM control register. The control register for SIMM slot 1 is called SIMM Memory Control Register 1, and so on for SIMM slot 2. The registers are initialized to 0 at reset. They must not be changed during any other type of access (CPU, DMA, or PCI bus) to the SIMM memory space. If SIMM memory is enabled, software should perform a read immediately before writing to this register, because write cycles are posted in the controller's write FIFO and a read cycle will force the controller to write back the FIFO contents before servicing the read cycle.

The controller can be configured to force the system to boot from SIMM slot 2 instead of from boot ROM. (Figure 7 on page 36 shows an example of SDRAM connections for SIMM. For details, and for the values in SIMM Memory Control Register 2, see Section 6.5.4 through Section 6.5.6.

The two SIMM memory control registers are each 4 bytes wide, at offsets 0x4 and 0x0C. Each contains the following fields:

Bits 1:0	Type	*Memory type* 0 = EDO DRAM 1 = Fast-page DRAM 2 = Flash 3 = SDRAM (16 Mb or 64 Mb)
Bit 2	SD	*Number of sides* 1 = Two sided. See Section 6.8.4 on page 40. 0 = Single sided
Bit 3	En	*SIMM memory enable* 1 = Enables SIMM memory 0 = Disables SIMM memory
Bits 5:4	MuxMode	*Address Multiplexing Modes 0, 1, 2, or 3* 00 = Mux Mode 0 01 = Mux Mode 1 10 = Mux Mode 2 11 = Mux Mode 3 See Table 9 and Table 10 for descriptions of these modes. Mux Mode 4 is configured by clearing bits 5:4 to 0 (as in Mux Mode 0) and setting bit 12.
Bit 6	ID	*EDO Identification mode* 1 = Places the controller into the EDO Identification mode, which is a special boot sequence. This bit is used in conjunction with bit 7.
Bit 7	D31	*Value of MDa bit 31 during EDO Identification mode* Read only. The state of bit 31 on the MDa bus, at the time bit 6 of this register (the ID bit) is set to 1 (while the EDO identification sequence is performed). At all other times, the value of this bit is 0. This bit is used only in conjunction with bit 6 (the ID bit).
Bit 8	CAS Latency	1 = 2 cycles 0 = 3 cycles For CAS latency of 2, initial latency is reduced by two more clock cycles in each case. Upon power-up reset, MuxAD[20] sets CAS latency for base and SIMM memory. This bit should be set to the same value as MuxAD[20].

Bit 9	Burst Access Cycle	Control for access time for subsequent words after the initial word access in SDRAM SIMM memory during burst access. For a read access of 8 words, the access time for CAS latency of 3 would be as follows. 0 = 10-2-2-2-2-2-2-2 (The first 10 cycles include the first word.) 1 = 9-1-1-1-1-1-1-1 (The first 9 cycles include the first word.) A corresponding performance improvement is achieved in write cycles as well (Table 11).
Bits 11:10	Reserved	*Hardwired to 0*
Bit 12	Mux Mode 4	*Address Multiplexing Mode 4* 1 = Mux Mode 4 (used for 64-Mb SDRAM only) 0 = Mux modes 0, 1, 2, or 3, as determined by bits 5:4, above. See Table 10 for a description of Mux Mode 4. Mux Mode 5 and Mux Mode 6 are determined per bits 12, 5, and 4 as follows.

Mux Mode	bit 12	bit 5	bit 4
5	1	0	1
6	1	1	0
4	1	0	0

Bits 18:13	Mask	*Physical address mask* Determines SIMM memory size by masking off address bits from the address comparison, beginning with bit 21. Thus, bits 26:21 of the physical address may be masked, providing an address space between 2 MB (no bits masked) and 128 MB (6 bits masked). Masks must be a pattern of left-justified 1s or 0s. A 1 in the Mask field indicates that the corresponding address bit is not masked.
Bits 20:19	Reserved	*Hardwired to 0*
Bits 27:21	SimmAdd	*SIMM memory base memory address* This 7-bit field (when appended with bits 31:28, which are always 0) is compared with the most-significant 11 bits of the physical address. A match indicates that the access is to the corresponding SIMM. Bit 28 is not used in the address comparison.
Bits 31:28	Reserved	*Hardwired to 0*

6.8.2 SDRAM in SIMM

The SIMM address ranges can be configured with flash, SDRAM, EDO, or fast-page SIMM or DIMM. The system can boot from flash memory in Slot 2. MuxAD bits 19:18 define the size of the flash memory per Table 30.

6.8.3 SDRAM Device Configurations

The controller supports the following (but is not limited to) 16-/64-/128-Mb NEC SDRAM parts and configurations in SIMM memory.

- 512 K x 32 (16-Mb) chips, 2 banks of 2 K rows, 256 columns (NEC # μPD4516161)
- 1 M x 16 (16-Mb) chips, 2 banks of 2 K rows, 256 columns (NEC # μPD4516161)
- 2 M x 8 (16-Mb) chips, 2 banks of 2 K rows, 512 columns (NEC # μPD4516821)
- 8 M x 8 (64-Mb) chips, 4 banks of 4 K rows, 512 columns (NEC # μPD4564841)
- 4 M x 16 (64-Mb) chips (NEC # μPD456163)
- 2 M x 32 (64-Mb) chips (NEC # μPD454323)
- 16 M x 8 (128-Mb) chips, 4 banks of 4 K rows, 1 K columns (NEC # μPD45128841)
- 16 M x 4 (64-Mb) chips, 4 banks of 4 K rows, 1 K columns (NEC # μPD4564441)

Table 23 shows some of the SDRAM configurations supported for base memory.

Table 23. SIMM Memory SDRAM Configuration Examples

Memory Size	SysAD Address Bits Required	Bank A	Sides	SysAD[25:21] Mask Bits	NEC Part Number
4 MB	21:0	2, 1 M x 16	Single	11110	μPD4516821G5-A10
8 MB	22:0	2, 1 M x 16	Double	11100	μPD4516821G5-A10
8 MB	22:0	4, 2 M x 8	Single	11100	μPD4516421G5-A10
16 MB	23:0	4, 2 M x 8	Double	11000	μPD4516421G5-A10
16 MB	23:0	8, 4 M x 4	Single	11000	Not available

6.8.4 DIMM

The 100-pin DIMM package is the only DIMM package supported. Each DIMM has four chip-select signals: S0, S1, S2, and S3. In noninterleaved (single-bank) configurations, each module uses only its two front-side chip-select inputs (S0 and S2).

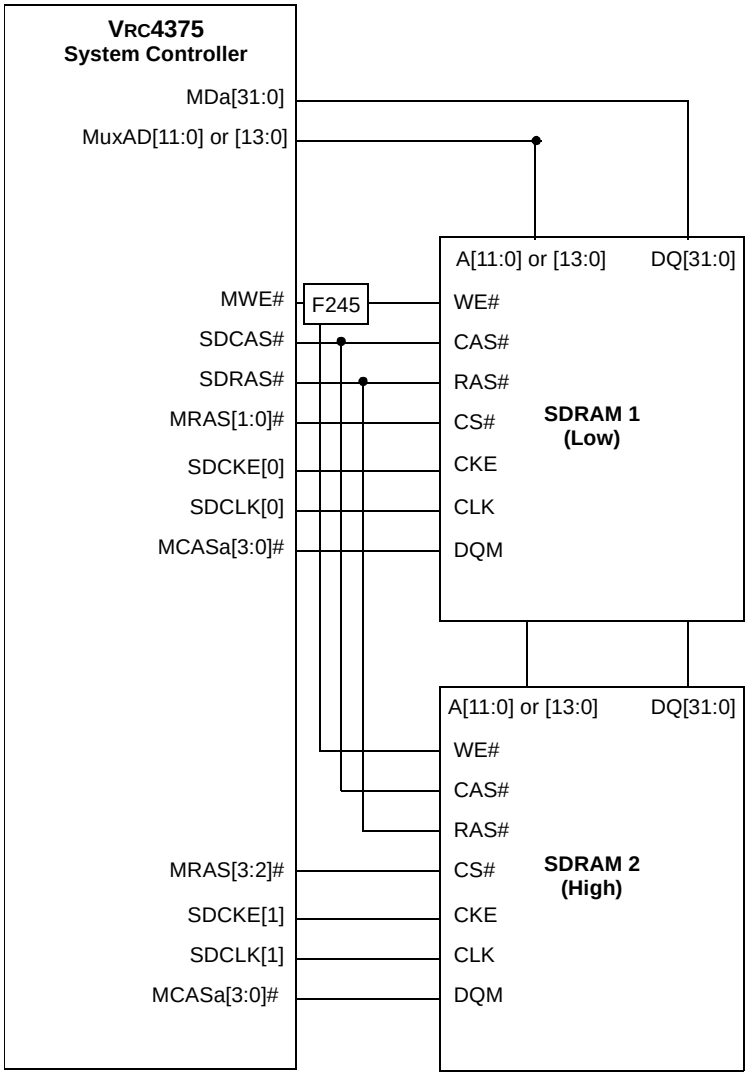
In one bank configuration, each of the controller signals used as a chip select can be connected to a maximum of four chip-select inputs on a single SDRAM chip (a DIMM module typically carries between 1 and 16 such chips).

6.8.5
SDRAM Signal Connections

Figure 8 shows how SDRAM and SIMM or DIMM packages are connected in the SIMM memory range. This example shows six controller signals used as chip selects: MRAS[3:0]# and SDCS[1:0]. Normally, eight controller signals would be used as chip selects, as shown in Table 24 (two signals for each side of each module), but two of the signals in Figure 8 (MRAS[1#] and MRAS[3]#) serve two modules each, due to the limited number of signals available on the controller. MRAS[1]# and MRAS[3]# may need buffering. The available chip selects from the controller can be connected in any way to the modules; software need not be involved in these connections.

If SIMM Memory Control Registers 1 and 2 are programmed to implement non-bank-interleaved memory, SIMM 1 and SIMM 2 contain both even and odd words.

Figure 8. SDRAM Connections for SIMM Memory (Example)



6.8.6 SDRAM Loads and Signals

Table 24 shows the number of device loads and signals used by various single-sided (SIMM) and double-sided (DIMM) configurations. In this table, one SDRAM device load equals about 8 pF.

The following DIMM/SIMM modules are also supported.

- MC-454AD644, 32 MB, 2 M x 64 x 2, 3.3 V, µPD4516821 (device)
- MC-454CB645, 32 MB, 4 M x 64 x 1, 3.3 V, µPD4564163 (Rev. E) (device)
- MC-452AB644, 16 MB, 2 M x 64 x 1, 3.3 V, µPD4516821 (device)
- MC-452AD644, 16 MB, 1 M x 64 x 2, 3.3 V, µPD4516161 (device)

Table 24. Loads and Signals for SIMM or DIMM

Signal to SIMM or DIMM	Number of Loads[1]								Number of Signals	
	x4		x8		x16		x32			
	1 Side	2 Sides	1 Side	2 Sides	1 Side	2 Sides	1 Side	2 Sides	1 Side	2 Sides
RAS#	8	N/S[2]	4	8	2	4	1	2	1	1
CAS#	8	N/S	4	8	2	4	1	2	1	1
WE#	8	N/S	4	8	2	4	4	2	1	1
DQM	2	N/S	1	2	1	2	1	2	4	4
CS#	4	N/S	2	2	1	1	1	1	2[3]	4[3]
Data	1	N/S	1	2	1	2	1	2	16	16
Address	8	N/S	4	8	2	4	1	2	12 or 14[4]	12 or 14[4]
CLK	4	N/S	4	4	2	2	1	1	2[5]	2[5]
CKE	4	N/S	4	4	2	2	1	1	2[6]	2[6]

Notes:
1. One device load equals 8 pF.
2. N/S = Not supported.
3. DIMM have four chip-select signals (S0, S1, S2, and S3). In single-bank configurations, (noninterleaved) only two are used: S0 and S2 on the front side. Each controller signal used for chip selection can connect to a maximum of four DIMMs.
4. Twelve address bits for 16-Mb SDRAM chips; 14 address bits for 64-Mb SDRAM chips.
5. Each SIMM or DIMM has two clock signals (CLK[1:0]). There can be a maximum of eight SDRAM devices per DIMM, in either single- or double-bank configurations. Each clock signal connects to four SDRAMs through two resistors.
6. Each SIMM or DIMM has two clock enable signals (CKE[1:0]). There can be a maximum of eight SDRAM devices per SIMM, in either single- or double-bank configurations. Each clock enable signal connects to four SDRAMs.

6.8.7 SDRAM Word Ordering

The word-address ordering for cache line fills from SDRAM in the SIMM memory range is the same as the word ordering in the base memory range (Section 6.7.8 on page 37).

6.9 DRAM Refresh

The controller supports CAS#-before-RAS# (CBR) DRAM refreshing to all DRAM address ranges. The refresh clock is derived from the system clock; its rate is determined by a programmable 12-bit counter in the DRAM Refresh Counter register, described below.

6.9.1 DRAM Refresh Counter Register

The DRAM Refresh Counter Register stores a word at offset 0x58. At reset, all bits in this register are set to 0. The register contains the following fields:

Bits 11:0	Cntr	*Refresh counter value* The refresh counter counts down from this value at the system clock rate. The refresh pulse is generated upon reaching 0. The reset value is 0. A value of 0x400 is equivalent to approximately 15 ms at a 66-MHz clock rate. DRAM refreshing is enabled and disabled in the Power-on Memory Initialization register (Section 6.11.1).
Bits 31:12	Reserved	*Hardwired to 0*

6.9.2 Refresh Mechanism

The refresh logic requests access to DRAM from the internal bus-arbitration logic each time the counter reaches 0. The refresh logic can accumulate up to a maximum of 8 refresh requests while it is waiting for the bus. Once the refresh logic owns the bus, all accumulated refreshes are performed to base memory and any installed SIMMs, and no other accesses (CPU, DMA, or PCI) are allowed. Refreshes are staggered by one clock (there is at least one bus clock between transitions on any pair of MRAS# signals).

Refresh automatically closes all open DRAM pages and clears the base memory prefetch FIFO. Refresh is disabled whenever the ID bit is set in any of the SIMM Memory Control registers, although refreshes will accumulate normally even when refresh is disabled. Accumulated refreshes are performed as soon as refreshing is re-enabled.

6.10 CPU to Memory Write FIFO

The controller has an 8-word CPU-to-memory write FIFO. (PCI writes to memory are buffered in the PCI target FIFO, described in Section 7.4.2.) This FIFO accepts writes at the maximum CPU speed. A single address is held for the buffered write cycle, allowing the buffering of a single write transaction. That transaction may be a word, double-word, or 4-word data cache write-back. When a word is placed in the FIFO by the CPU, the controller attempts to write the FIFO's contents to memory as quickly as possible. If the next CPU read or write cycle is addressed to memory, the controller negates EOK#, thus causing the next CPU transaction (read or write) to stall until the controller empties its FIFO. If the next CPU transaction (read or write) is addressed to a PCI bus target, the controller asserts EOK#, thus allowing the CPU transaction to complete. If, upon completion of such a CPU transaction to a PCI bus target, the controller's FIFO is still not empty, the controller will again negate EOK# to stall the next CPU write until the FIFO contents are written back to memory.

6.11 Memory Initialization

To be accessed, memory must first be initialized by software at power-on. The following sections describe the Power-on Memory Initialization register and the initialization sequence.

6.11.1 Power-on Memory Initialization Register

The Power-on Memory Initialization register configures SDRAM in both the base and SIMM memory address ranges. At reset, the register is set to 0 and it must be configured before memory is accessed after power-on. The register stores a word at offset 0x78 and has the following fields:

Bit 0	Mode	*SDRAM mode set* 1 = Enables writing to the mode registers on all SDRAM chips. When this bit is set, the controller automatically provides the data that configures all SDRAM chips on all SIMMs or DIMMs for a burst length of 8 words and CAS# latency of 3 cycles. 0 = Disables writing to mode registers on SDRAMs
Bit 1	Precharge	*SDRAM precharge* 1 = Precharge Setting this bit causes the controller to issue two sequential precharge commands to any SDRAMs in the base and SIMM memory ranges. Do not set this bit during normal system operation. It should be set only during the power-on process. 0 = No precharge This bit is cleared automatically by the controller at the end of the two precharge commands.
Bit 2	Refresh	*Refresh enable (SDRAM and DRAM)* 1 = Refresh Setting this bit causes the controller to issue eight sequential automatic refresh (CAS#-before-RAS#) commands to any SDRAMs in the base and SIMM memory ranges. This is required during SDRAM initialization. The refresh commands are issued only if the SDRAM memory type has been programmed in the base or SIMM memory ranges. This bit also enables refresh of DRAMs. 0 = No refresh (Default value at power-on or reset.) This bit is cleared automatically by the controller at the end of the eight automatic refresh commands.
Bits 31:3	Reserved	*Hardwired to 0*

6.11.2 Power-On Initialization Sequence

Follow this sequence to configure memory at power-on.

1. Initialize the 0F00_007C and 0F00_0080 registers to zero.
2. Program the Base Memory Control register (see Section 6.7.1).
3. Program the two SIMM memory control registers if these address ranges are used (see Section 6.8.1).
4. If SDRAM is installed in any address range, set the precharge bit (bit 1) in the Power-on Memory Initialization register.
5. If SDRAM devices are used, wait for eight CPU clocks. (This is required to finish the SDRAM precharge sequence initiated in step 4.)
6. Set the refresh bit (bit 2) in the Power-on Memory Initialization register. This enables refresh for all SDRAM and DRAM, and (if SDRAM is installed) it initiates 8 sequential SDRAM refresh cycles.
7. Wait for step 6 to complete. If SDRAM is installed, this is approximately 60 CPU clocks (8 SDRAM refresh cycles).
8. If SDRAM is installed in any address range, set the mode bit (bit 0) in the Power-on Memory Initialization register. This configures all SDRAM chips for a burst length of eight words and CAS# latency of 3 cycles.
9. Wait for nine CPU clock cycles and step 8 to complete.
10. Program the DRAM Refresh Counter register (Section 6.9.1).

At this point, memory is ready to use. All other Configuration registers in the controller should then be programmed before commencing normal operation.

6.12 UART Registers

The VRC4375 controller uses the NEC NY16550L megafunction macro as its internal Universal Asynchronous Receiver/Transmitter (UART). This UART is functionally identical to the National Semiconductor PC16550D® megafunction macro. Refer to the NEC *Megafunction: NY16550L User's Manual* for more information and programming details.

This section describes the available UART registers. The modem control function is not supported in this version of the UART, so the registers relating to those signals are not relevant. Refer to the signal summary (Section 3.0) for UART signals that are supported.

Note: After writing a value or a series of values in any register or set of registers, a write to the Scratch Pad register must be made.

6.12.1 UART Receiver Data Buffer Register (UARTRBR)

Bits 7:0	UDATA	*UART data (read only) when the divisor latch access bit (DLAB) = 0*
Bits 31:8	Reserved	*Hardwired to 0*
Reset Value:		*0x0000 00xx*

This register holds receive data. It is only accessed when the Divisor Latch Access bit (DLAB) is cleared in the UARTLCR.

6.12.2 UART Transmitter Data Holding Register (UARTTHR)

Bits 7:0	UDATA	*UART data (read only) when DLAB = 0*
Bits 31:8	Reserved	*Hardwired to 0*
Reset Value:		*0x0000 00xx*

This register holds transmit data. It is only accessed when the Divisor Latch Access bit (DLAB) is cleared in the UARTLCR.

6.12.3 UART Interrupt Enable Register (UARTIER)

This register is used to enable UART interrupts. It is only accessed when DLAB is set in the UARTLCR. Interrupt Control and Status register (Section 9.1.2) UARTINTEN bit 22 is a global enable for interrupt sources enabled by this register.

Bit 0	ERBFI	*UART receive data buffer full interrupt* 1 = Enables receive data available interrupts 0 = Disables receive data available interrupts Receive data buffer full state is reported to UARTLSR.
Bit 1	ERBEI	*UART transmitter buffer empty interrupt* 1 = Enables transmitter buffer empty interrupt 0 = Disables transmitter buffer empty interrupt Transmitter buffer empty state is reported to UARTLSR.
Bit 2	ERBFI	*UART line status interrupt* 1 = Enables line status error interrupt 0 = Disables line status error interrupt Line status error interrupt state is reported to UARTLSR.
Bit 3	ERBFI	*UART modem status change interrupt* 1 = Enables modem status change interrupt 0 = Disables modem status change interrupt Modem status changes are reported to UARTMSR[3:0].
Bits 7:4	Reserved	*Hardwired to 0*
Bits 31:8	Reserved	*Hardwired to 0*
Reset Value:		*0x0000 00xx*

6.12.4 UART Divisor Latch LSB Register (UARTDLL)

Bits 7:0	DIVLSB	*UART divisor latch (least-significant byte)* Only accessed when DLAB = 1 in UARTLCR
Bits 31:8	Reserved	*Hardwired to 0*
Reset Value:		*0x0000 00xx*

6.12.5 UART Divisor Latch MSB Register (UARTDLM)

Bits 7:0	DIVMSB	*UART divisor latch (most-significant byte)* Only accessed when DLAB = 1 in UARTLCR
Bits 31:8	Reserved	*Hardwired to 0*
Reset Value:		*0x0000 00xx*

6.12.6 UART Interrupt ID Register (UARTIIR)

Bit 0	INTPENDL	0 = UART interrupt pending (read only) 1 = No UART interrupt pending
Bits 3:1	UIID #	Only valid when INTPENDL = 0.

UIID[3:1]#:	**Priority:**	**Source of interrupt:**
0x3	Highest	Receiver line status: Overrun error, parity, framing error, or break interrupt
0x2	Second Highest	Received data available: Receiver data available or trigger level reached
0x6	Third Highest	Character time-out indication: No change in receiver FIFO during the last four character times and FIFO is not empty
0x1	Fourth Highest	Transmitter Holding register is empty
0x0	Fifth Highest	Modem status: CTS_L, DSR_L or DCD_L

Bits 5:4	Reserved	*Hardwired to 0*
Bits 7:6	UFIFOEN	*UART FIFO is enabled (read only)* Both bits are set to 1 when the transmit/receive FIFO is enabled and the UFIFOEN0 bit is set in the UARTSCR.
Bits 31:8	Reserved	*Hardwired to 0*
Reset Value:		*0x0000 00xx*

6.12.7 UART FIFO Control Register (UARTFCR)

Bit 0	UFIFOEN0	*UART FIFO enable (write only)* 1 = Enables receive and transmit FIFOs 0 = Disables and clears receive and transmit FIFOs
Bit 1	URFRST	*UART receiver FIFO reset (write only)* 1 = Clears receive FIFO and reset counter 0 = Does not clear receive FIFO and reset counter
Bit 2	UTFRST	*UART transmitter FIFO reset (write only)* 1 = Clears transmit FIFO and reset counter 0 = Does not clear transmit FIFO and reset counter
Bits 5:3	Reserved	*Hardwired to 0x0*
Bits 7:6	URTR	*UART receive FIFO trigger level*
Rcvr Trig Lvl Bits[7:6]		***Number of bytes in receiver FIFO***
0x0		01
0x1		04
0x2		08
0x3		14
		When the trigger level is reached, a receive buffer full interrupt is generated, if enabled by the ERBFI bit in the UARTIER.
Bits 31:8	Reserved	*Hardwired to 0x0000 00*
Reset Value:		*0x0000 00xx*

6.12.8
UART Line Control Register (UARTLCR)

Bits 1:0	WLS	*Word length select* 11 = 8 bits 10 = 7 bits 01 = 6 bits 00 = 5 bits
Bit 2	STB	*Number of stop bits enabled* 1 = Enables 2 bits, except 1.5 stop bits for 5 words 0 = Enables 1 bit
Bit 3	PEN	*Parity enable* 1 = Generates parity on writes, checks it on reads 0 = Does not generate or check parity For the UART, even or odd parity can be generated or checked, as specified in bit 4 (EPS). This is different from the parity on the CPU, memory, or PCI bus, which is considered even parity.
Bit 4	EPS	*Parity select* 1 = Selects even parity 0 = Selects odd parity
Bit 5	USP	*Stick parity* 1 = Forces UART_TxDRDY# signal output low 0 = Generates and checks parity normally This bit is only valid when the parity enable (PEN) bit is set.
Bit 6	USB	*Set break* 1= Forces UART_TxDRDY# signal output Low 0 = Generates normal operation of UART_TxDRDY signal output
Bit 7	DLAB	*Divisor latch access* 1 = Accesses baud-rate divisor at offset 0x88 0 = Accesses TxD/RxD and IE at offset 0x88 When this bit is set, UART accesses the UART Divisor Latch LSB register (UARTDLM) at offset 0x88. When cleared, the UART accesses the Receiver Data Buffer register (UARTRBR) on reads at offset 0x84, the UARTTHR on writes at offset 0x84, and UARTIER on any accesses at offset 0x88.
Bits 31:8	Reserved	*Hardwired to 0x0000 00*
Reset Value:		*0x0000 00xx*

6.12.9 UART Modem Control Register (UARTMCR)

This register controls the state of external UART_DTR# and UART_RTS# modem-control signals and of the loopback test.

Bit 0	DTR	*Data terminal ready* 1 = Negates UART_DTR# signal 0 = Asserts UART_DTR# signal
Bit 1	RTS	*Request to send* 1 = Negates UART_RTS# signal 0 = Asserts UART_RTS# signal
Bit 2	OUT1	*Out 1* 1 = OUT1# state active 0 = OUT1# state inactive (reset value) This is a user-defined bit that has no associated external signal. Software can write to the bit, but this has no effect.
Bit 3	OUT2	*Out 2* 1 = OUT2# state active 0 = OUT2# state inactive (reset value) This is a user-defined bit that has no associated external signal. Software can write to the bit, but this has no effect.
Bit 4	LOOP	*Loopback test* 1 = Enables loopback 0 = Enables normal operation This is an NEC internal test function.
Bits 7:5	mbz	*Must be zero. Hardwired to 0x0.*
Bits 31:8	mbz	*Must be zero. Hardwired to 0x0000 00.*

6.12.10 UART Line Status Register (UARTLSR)

This register reports the current state of the transmitter and receiver logic.

Bit 0	DR	*Receive data ready* 1 = Receive data buffer full 0 = Receive data buffer not full Receive data is stored in the UART Receiver Data Buffer register (UARTRBR).
Bit 1	OE	*Receive data overrun error* 1 = Overrun error on receive data 0 = No overrun error
Bit 2	PE	*Receive data parity error* 1 = Parity error on receive data 0 = No parity error
Bit 3	FE	*Receive data framing error* 1 = Framing error on receive data 0 = No framing error
Bit 4	BI	*Break interrupt* 1 = Break received on UART_RxDRDY# signal 0 = No break
Bit 5	THRE	*Transmitter Holding register empty* 1 = Transmitter Holding register empty 0 = Transmitter Holding register not empty Transmit data is stored in the UART Transmitter Data Holding register (UARTTHR).
Bit 6	TEMT	*Transmitter empty* 1 = Transmitter Holding and Shift registers empty 0 = Transmitter Holding or Shift register not empty
Bit 7	RFERR	*Receiver FIFO error* 1 = Parity, framing, or break error in receiver buffer 0 = No parity, framing, or break error
Bits 31:8	mbz	*Must be zero. Hardwired to 0x0000 00.*

6.12.11 UART Modem Status Register (UARTMSR)

This register reports the current state of and changes in various control signals.

Bit 0	DCTS	*Delta clear to send* 1 = UART_CTS# state changed since this register was last read 0 = UART_CTS# state did not change
Bit 1	DDSR	*Delta data set ready* 1 = UART_DSR# input signal changed since this register was last read 0 = UART_DSR# input signal did not change
Bit 2	TERI	*Trailing-edge ring indicator* 1 = RI# state changed since this register was last read 0 = RI# state did not change RI# is not implemented as an external signal, so this bit is never set by the controller
Bit 3	DDCD	*Delta data carrier detect* 1 = UART_DCD# state changed since this register was last read 0 = UART_DCD# state did not change
Bit 4	CTS	*Clear to send* 1 = UART_CTS# state active 0 = UART_CTS# state inactive This bit is the complement of the UART_CTS# input signal. If the LOOP bit in the UART Modem Control Register (UARTMCR) is set to 1, the CTS bit is equivalent to the RTS bit in the UARTMCR.
Bit 5	DSR	*Data set ready* 1 = UART_DSR# state active 0 = UART_DSR# state inactive This bit is the complement of the UART_DSR# input signal. If the LOOP bit in the UART Modem Control Register (UARTMCR) is set to 1, the DSR bit is equivalent to the DTR bit in the UARTMCR.
Bit 6	RI	*Ring indicator* 1 = Not valid 0 = Always reads 0 This bit has no associated external signal.
Bit 7	DCD	*Data carrier detect* 1 =UART_DCD# state active 0 = UART_DCD# state inactive This bit is the complement of the UART_DCD# input signal. If the LOOP bit in the UART Modem Control Register (UARTMCR) is set to 1, the DCD bit is equivalent to the OUT2 bit in the UARTMCR.
Reset Value:		*0x0000 00xx*

6.12.12
UART Scratch Register (UARTSCR)

This register contains a UART reset bit plus eight bits of space for any software use.

Bits 7:0	USCR	*UART Scratch register* Available to software for any purpose.
Bit 8	URESET	*UART reset* 1 = Resets UART 0 = No reset
Reset Value:		*0x0000 0xxx*

To achieve a desired baud rate, the UART divisor latch must be properly programmed. The relationship between UART clock frequency, baud rate, and divisor value is:

baud_rate = Sys_Clock_freq / (divisor_value x 16 x 8)

Example: To calculate a divisor value for a desired baud rate of 9600 and SysClk = 66 MHz:

Divisor = 66000000 Hz/(9600 x 16 x 8) = 53.7 decimal = 36 Hex

Table 25 gives divisor values for several input clock frequencies.

Note: The actual baud rate may vary significantly from the desired baud rate.

Table 25. Input Clock Frequency Divisor Values

Baud Rate	Divisor	% Error	Divisor	% Error	Divisor	% Error	Divisor	% Error
50	2304		10368		9216	7680		
75	1536		6912		6144	5120		
110	1047	0.026	4713	0.006	4189	0.002	3491	0.003
134.5	857	0.058	3854	0.007	3426	0.001	2855	0.001
150	768		3456		3072		2560	
300	384		1728		1536		1280	
600	192		864		768		640	
1200	96		432		384		320	
1800	64		288		256		213	0.156
2000	58	0.690	259	0.077	230	0.174	192	
2400	48		216		192		160	
3600	32		144		128		107	0.312
4800	24		108		96		80	
7200	16		72		64		53	0.629
9600	12		54		48		40	
19,200	6		27		24		20	
38,400	3		14	3.571	12		10	
57,600	2		9		8		7	4.762

7.0 PCI Bus Interface

The controller's PCI bus interface complies with the *PCI Local Bus Specification, Revision 2.1*. Complete master and target capabilities are supported. No external logic or buffering is necessary. The interface implements 3.3-V, PCI-compliant pads (5-V tolerant) using the NEC CMOS-9 process technology. All PCI interface electrical characteristics (loading, drive, impedance, capacitance, and so forth) comply fully with the PCI specification.

The PCI bus interface contains two separate data paths—one for CPU access and one for DMA. Each path has its own data pipeline and FIFO, and each one operates independently of the other. The FIFOs in this interface include:

- 4-word (16-byte) bidirectional PCI master FIFO (CPU is a PCI bus master)
- 8-word (32-byte) bidirectional PCI target FIFO (memory is a PCI bus target)
- 8-word (32-byte) bidirectional DMA FIFO (PCI to memory or memory to PCI)

7.1 PCI Bus Timing

The PCI bus operates at 33 MHz and supports full burst transfers; no wait states are required with adequately fast memory. Peak PCI bus bandwidth is 133 MB/s. The PCI bus is synchronized to the SysAD bus, with the SysAD bus clock running at two times the PCI clock.

7.2 PCI Commands Supported

Table 26 summarizes the PCI command codes and whether they are supported or not supported by the controller when it is functioning as a master and as a target.

Table 26. PCI Commands

CBE#[3:0]	Command	Command Supported When Controller Is Master	Command Supported When Controller Is Target
0000	Interrupt acknowledge	No	Ignored
0001	Special cycle	No	Ignored
0010	I/O read	Yes, via PCI master I/O window (see Section 7.3)	Yes; must be in Add-On Board mode and hit PCI I/O base memory address range
0011	I/O write	Yes, via PCI master I/O window (see Section 7.3)	Yes; must be in Add-On Board mode and hit PCI I/O base memory address range
010x	*Reserved*	—	Ignored
0110	Memory read	Yes, via PCI Master Address Windows (see Section 7.3)	Yes; must hit a PCI Target Address Window (see Section 7.4)
0111	Memory write	Yes, via PCI Master Address Windows (see Section 7.3)	Yes; must hit a PCI Target Address Window (see Section 7.4)
100x	*Reserved*	—	Ignored
1010	Configuration read	Yes, via PCI Configuration registers (see Section 7.5)	Yes, via PCI Configuration registers (see Section 7.5)
1011	Configuration write	Yes, via PCI Configuration registers (see Section 7.5)	Yes, via PCI Configuration registers (see Section 7.5)
1100	Memory read multiple	No	Aliased to memory read
1101	Dual address cycle	No	Ignored
1110	Memory read line	No	Aliased to memory read
1111	Memory write and invalidate	No	Aliased to memory write

7.3 PCI Master Transactions (CPU to PCI Bus)

The controller supports bidirectional data transfers between the CPU and PCI bus targets by becoming a PCI bus master. The CPU obtains access to PCI bus resources (summarized in Table 7 on page 16) by accessing a local physical address that corresponds to one of three PCI address windows:

- PCI Master Address Window 1
- PCI Master Address Window 2
- PCI Master I/O Window

These registers are at offsets 14, 18, and 24, respectively (see Table 6). They are configured through the PCI Master Address Window registers, described below.

7.3.1 PCI Master Address Window Registers

The three PCI Master Address Window registers described above have the same structure. At reset, they are set to 0. A register must not be changed while a write is posted to the PCI bus. There must be at least two CPU clocks between writing to this type of register and performing a PCI access through the window mapped by the register.

Each of the three PCI Master Address Window registers contains the following fields.

Bits 7:0	PCIAdd	*PCI address* This 8-bit field replaces the most-significant 8 bits of the address defined in the LAdd field when the address is transmitted to the PCI bus. Bits masked by the Mask field (bits 19:13) are directly transferred from the CPU's SysAD bus (rather than from the PCIAdd field).
Bits 11:8	Reserved	*Hardwired to 0*
Bit 12	E	*Enable* 1 = Enables access to the PCI bus through the address window specified in this register 0 = Disables access
Bits 19:13	Mask	*Physical address mask* This mask is used to determine the size of the PCI window. It masks 7 address bits from the address comparison, beginning with bit 24. Thus, bits 30 to 24 may be masked, providing an address block size between 16 MB (no bits masked) and 2 GB (7 bits masked). A 0 in a mask bit indicates that the corresponding address bit is masked.
Bits 23:20	Reserved	*Hardwired to 0*
Bits 31:24	LAdd	*Local base memory address* This 8-bit field is compared with the most-significant 8 bits of the physical CPU address, conditioned on the Mask field. A match indicates that the access is to the PCI bus. LAdd should not be programmed to overlap PCI space with local resources (memory, registers or boot ROM) or PCI target windows; this would result in improper operation.

7.3.2 PCI Master Transaction Details

Transfers between the CPU and PCI bus are buffered through a 4-word bidirectional PCI master FIFO. This FIFO stores data and latches the address and byte enables for one CPU-to-PCI read or write transaction. When the CPU accesses an address in the window defined by the LAdd fields (bits 31:24) of either of the two PCI Master Address Window registers or the PCI master I/O window, the data is transferred to and from the PCI bus through the FIFO. The FIFO improves performance and provides a mechanism for resolving deadlocks between the PCI and SysAD buses.

The FIFO size of 4 words allows the CPU to perform all possible write transactions. All CPU read transactions are supported, except instruction cache line fills (8-word burst transfers).

For data cache line fills from the PCI bus, the controller reads 4 words from the PCI bus, beginning with the first word in the line (word address = 0), and returns them to the CPU in the correct subblock order. For example, a data cache line fill from address 2 is read from the PCI bus as 4 words, beginning at address 0 (0, 1, 2, 3) and returned to the CPU beginning at address 2 (2, 3, 0, 1). The controller does not support the PCI Cache Line Wrap mode.

During CPU-to-PCI bus transactions, the FIFO accepts write data at the CPU rate. If the CPU is performing a data cache write-back, a burst of 4 words occurs. The address and byte-enables for the cycle are first latched in the FIFO. Then the data words are placed in the FIFO, and the PCI bus is requested. If the CPU attempts another PCI write before the FIFO is empty, the controller stalls the CPU write. If the PCI bus has not been acquired before the FIFO is filled, the controller indicates to the CPU that further write (and read) cycles will be stalled by negating the EOK# signal to the processor. Write cycles to resources other than the PCI bus are allowed to complete after the controller decodes the address.

The controller uses the PCI block transfer protocol if the CPU read is also more than one word. During block reads, the CPU is stalled by the controller until a word has been placed in the FIFO from the PCI bus. For CPU accesses to the PCI bus of less than one word, the controller reads or writes the correct number of bytes.

Until the controller is granted the PCI bus, another PCI master may have ownership of the PCI bus and may request access to the controller as a PCI target. A PCI target FIFO in the controller allows such an access to occur without causing deadlock, as described in the next section.

7.4 PCI Target Transactions (PCI to Memory)

The controller supports bidirectional data transfers between a PCI bus master and the controller's memory as the target. The PCI bus master obtains access to the controller's memory by accessing a local physical address that corresponds to one of two PCI address windows:

- PCI Target Address Window 1
- PCI Target Address Window 2

These registers are at offsets 1C and 20, respectively (see Table 6). They are configured through the PCI Target Address Window registers, as described below.

7.4.1 PCI Target Address Window Registers

The two PCI Target Address Window registers described above are set to 0 at reset. There must be at least two CPU clocks between writing to such a register and performing a PCI access through the window mapped by the register.

Each of the two PCI Target Address Window registers contains the following fields.

Bits 10:0	LAdd	*Local address* This 11-bit field replaces the most-significant 11 bits of the PCI address defined in the PCIAdd field (bits 31:24) when the address is transmitted to the memory. Bits masked by the Mask field are directly transferred from the PCI bus, rather than from the LAdd field.
Bit 11	Reserved	*Hardwired to 0*
Bit 12	E	*Enable* 1 = Enables the PCI bus to access local resources through the address window specified in this register 0 = Disables access
Bits 19:13	Mask	*PCI address mask* This mask is used to determine the size of the local window. It will mask 7 address bits from the address comparison, beginning with bit 21. Thus, bits 27:21 may be masked, providing an address block size between 2 MB (no bits masked) and 256 MB (7 bits masked). A 0 in a mask bit indicates that the corresponding address bit is masked.
Bit 20	Reserved	*Hardwired to 0*
Bits 31:21	PCIAdd	*PCI address* This 11-bit field is compared with the most-significant 11 bits of the PCI address, conditioned on the Mask field. A match indicates that the access is to the controller. Care must be taken not to overlap the two PCI target windows.

7.4.2
PCI to CPU Transaction Details

When the controller sees an address on the PCI bus that falls within one of its two PCI Target Address Window ranges, it responds by requesting access to its attached memory. This prevents the CPU from obtaining access to the memory. The controller supports full-speed burst read and write cycles from a PCI master. Accesses are performed through an 8-word bidirectional PCI target FIFO. This FIFO stores data and latches the address and byte-enables for one PCI to CPU read or write transaction. PCI target transfers have higher priority than PCI master transfers. Thus, if both CPU requests and PCI target requests for memory are present simultaneously, the PCI target transfer occurs first.

During PCI target read cycles, the controller always accesses memory in 4-word reads, using the data cache miss protocol. These words are placed in the target FIFO and sent to the PCI bus at maximum speed. If the PCI read address is not aligned to a cache line boundary, the controller stores only the required words in the target FIFO. When the PCI word address is 2 or 3, the controller transfers the word(s) and then disconnects from the target; the controller always disconnects if there are fewer than 2 words left in the FIFO for a PCI read cycle. The controller uses the CPU's subblock ordering for PCI target read cycles. Table 27 shows the read order for various access quantities.

Table 27. PCI Target Read Order and Buffering

PCI Word Address	Words Placed in PCI Target FIFO
0	0, 1, 2, 3
1	1, 2, 3
2	2, 3
3	3

If the controller detects bad parity on a PCI target *address* cycle, the controller reports the error in the PCI header, generates an interrupt on INTA# (if enabled), and performs the access (ignoring the parity error). If the controller detects bad parity on a PCI target *data* cycle, the controller reports the error in the PCI header, generates an interrupt on INTA# (if enabled), and performs the write.

7.5 PCI Configuration Space

The controller provides a PCI configuration space, as described in the *PCI Local Bus Specification*, Section 6. This space supports bus master configuration cycles of PCI devices using a mechanism similar to configuration mechanism #1 (*PCI Local Bus Specification,* Section 3.7.4.1).

Two 1-word registers are provided for software to perform configuration cycles:

- PCI Configuration Data register
- PCI Configuration Address register

These registers are at offsets 28 and 2C, respectively (see Table 6), and are cleared to 0 at reset. To perform a configuration cycle, the CPU first writes an address to the PCI Configuration Address register, then writes the transaction data to the PCI Configuration Data register. The access to the Data register causes the cycle to begin. Byte enables, read/write states, and the full 32-bit address pass through to the PCI bus without mapping. The CPU is stalled during read cycles until the PCI configuration cycle completes. This mechanism precludes the CPU from performing burst transfers to configuration space.

The PCI Configuration Address register format contains a register number field that is used to address specific PCI bus targets. Figure 3-20 of the *PCI Local Bus Specification* shows the format. Each configurable target on the PCI bus maintains a set of PCI Configuration Space registers that consist of header registers and device-dependent registers, as defined in Section 6.1 of the *PCI Local Bus Specification.*

The controller implements two sets of the PCI Configuration Space registers, depending on its mode of operation:

- *Host Bridge mode:* In this mode, the controller is located on the motherboard and acts as the PCI host bridge for the system. The PCI Configuration Space registers for this mode are described in Section 7.6.
- *Add-On Board mode:* In this mode, the controller is located on a PCI board rather than on the motherboard. The PCI Configuration Space registers for this mode are described in Section 7.7.

7.6 PCI Configuration registers (Host Bridge Mode)

The CPU uses this configuration space during system boot to configure the controller. In Host Bridge mode, the CPU accesses the controller directly through the controller's registers in the CPU's memory space (Table 6); the PCI Configuration Address register and PCI Configuration Data register are not used.

Table 28 shows the controller's PCI Configuration Space registers for this mode. The sections that follow define the fields in each register.

After changing any of these registers, at least two CPU clock cycles must elapse before a PCI access by the CPU, DMA, or an external master.

Table 28. PCI Configuration Space registers (Host Bridge Mode)[1]

Offset from Base Memory Address 0F00_0000	Size (Bytes)	Register Name	Symbol	CPU Read/ Write	Reset Value	Description	Reference
0x100	2	Vendor ID	VID	R	0x1033	Vendor ID for NEC	Section 7.6.1 on page 61
0x102	2	Device ID	DID	R	0x005B	VRC4375 controller's device ID, assigned by NEC	Section 7.6.1 on page 61
0x104	2	Command	PCICMD	R/W	0x0	Provides general control of PCI interface	Section 7.6.2 on page 62
0x106	2	Status	PCISTS	R/WC[2]	0x0280	Status for PCI events	Section 7.6.2 on page 62
0x108	1	Revision ID	RID	R	0x0	Device revision	Section 7.6.3 on page 63
0x109	3	Class Code	CLASS	R	0x06_0000	Device type	Section 7.6.3 on page 63
0x10C	1	Cache Line Size	CLSIZ	R	0x04	System cache line size (words)	Section 7.6.4 on page 64
0x10D	1	Latency Timer	MLTIM	R/W	0x0	Value of latency timer for this master, in PCI clock cycles	Section 7.6.4 on page 64
0x10E	2	*Reserved*		R	0x0		
0x110	4	Mailbox Base Address	MBADD	R/W	0x0	Base memory address for both mailboxes	Section 7.6.5 on page 64
0x13B – 0x114		*Reserved*		R	0x0		
0x13C	1	Interrupt Line	INTLIN	R	0x0	PCI interrupt signal	Section 7.6.6 on page 65
0x13D	1	Interrupt Pin	INTPIN	R	0x0	PCI interrupt pin	Section 7.6.6 on page 65
0x13F – 0x13E	2	*Reserved*		R	0x0		
0x140	1	*Reserved*		R	0x0		
0x141	1	Retry Value	RTYVAL	R/W	0x0	Number of PCI bus retries the controller performs	Section 7.6.7 on page 65
0x142	2	PCI Arbiter Priority Control and Take Away Grant	PAPC	R/W	0x0	Priority scheme used in granting access to PCI bus	Section 7.6.7 on page 65
0x1FF – 0x144		*Reserved*		R	0x0		

Notes: **1.** The row shading/no shading pattern defines 4-byte word boundaries. Some registers can be accessed along byte or 2-byte boundaries.
2. The bits can only be set by the controller hardware, and they are cleared by writing to them. Writing a 0 leaves them unaffected. For example, writing 0x8000 clears the most-significant bit.

7.6.1 Vendor and Device ID Registers

The Vendor and Device ID registers are read only. Together, they constitute a word at offset 0x100.

7.6.1.1 Vendor ID Register

The 2-byte Vendor ID register is read only and can be accessed at offset 0x100.

Bits 15:0 VID — *Vendor ID*
Hardwired to 0x1033 for NEC

7.6.1.2
Device ID Register

The 2-byte Device ID register is read only and can be accessed at offset 0x102.

Bits 31:16	DID	*Device ID* Hardwired to 0x005B for the VRC4375 controller

7.6.2
Command and Status Registers

The command and status registers, plus reserved fields, constitute a word at offset 0x104.

7.6.2.1
Command Register

The 2-byte Command register is read/write and can be accessed at offset 0x104.

Bit 0	IOEN	*I/O space enable* Cleared to 0 at reset. Software must set it to 1 to enable access to the PCI Interrupt register status in the Add-On Board mode.
Bit 1	MEMEN	*Memory space enable* Cleared to 0 at reset. Software must set it to 1 to allow the controller to respond to memory space accesses.
Bit 2	BMAS	*Bus master enable* Cleared to 0 at reset. Software must set it to 1 to allow the controller to generate PCI accesses.
Bit 3	SPC	*Special cycle enable* Hardwired to 0. The controller ignores special cycles.
Bit 4	MWI	*Memory write and invalidate enable* Hardwired to 0. The controller does not generate MWI commands.
Bit 5	VGA	*Hardwired to 0* The VRC4375 controller is not a VGA device.
Bit 6	PER	*Parity error (PERR#) enable* 1 = Enable 0 = Disable
Bit 7	WAIT_CTL	*Wait cycle control* Hardwired to 0. The controller never does address stepping.
Bit 8	SERR_EN	*System error (SERR#) enable* 1 = Enable 0 = Disable
Bit 9	FBBE	*Fast back-to-back enable* Hardwired to 0. The controller never generates back-to-back transactions.

The 1-byte location 0x105 is reserved.

Bits 15:10	Reserved	*Hardwired to 0*

7.6.2.2
Status Register

The 2-byte Status register can be accessed at offset 0x106. It uses a special read/write protocol: The bits can be set only by the controller hardware, but they can be cleared by writing 1 to them; writing 0 leaves them unaffected. For example, writing 0x8000 clears the most-significant bit.

Bits 20:16	Reserved	*Hardwired to 0*
Bit 21	66 MHz	*66-MHz capability* Hardwired to 0. The controller is a 33-MHz device.
Bit 22	UDF	*User-definable configuration (UDF) support* Hardwired to 0. The controller doesn't support UDF.
Bit 23	FBBC	*Fast back-to-back capability* Hardwired to 1. The controller will accept fast back-to-back accesses.
Bit 24	DPR	*Data parity reported* 1 = Enable 0 = Disable
Bits 26:25	DEVSEL	*Device select (DEVSEL) timing* Hardwired to 01 (medium response)
Bit 27	STA	*Signaled target abort* Set to 1 if the controller signals a target abort. Otherwise, cleared to 0.
Bit 28	RTA	*Received target abort* Set to 1 whenever the master receives a target abort. Otherwise, cleared to 0.
Bit 29	RMA	*Received master abort* Set whenever the master receives a master abort. Otherwise, cleared to 0.
Bit 30	SSE	*Signaled system error* 1 = Generates a bus error interrupt 0 = Does not generate a bus error interrupt
Bit 31	DPE	*Detected parity error* Set when the controller detects a parity error. Otherwise, cleared to 0.

7.6.3
Revision ID and Class Code Registers

The Revision ID and Class Code registers are read only. Together, they constitute a word at offset 0x108. They contain the following fields.

7.6.3.1
Revision ID Register

The 1-byte Revision ID register is read only and can be accessed at offset 0x108.

Bits 7:0	RID	*Revision ID* Hardwired to 0, indicating a gate array

7.6.3.2
Class Code Register

The 3-byte Class Code register is read only and can be accessed at offset 0x109.

Bits 15:8	Prog	*Programming interface* Hardwired to 0
Bits 23:16	SubCl	*Subclass* Hardwired to 0
Bits 31:24	BaseCl	*Base class* Hardwired to 0x06 to indicate a bridge device

7.6.4
Cache Line Size and Latency Timer

The Cache Line Size and Latency Timer registers are both 1 byte wide, followed by two reserved bytes. Together, these locations constitute a word at offset 0x10C.

7.6.4.1
Cache Line Size

The 1-byte Cache Line Size register is read only and can be accessed at offset 0x10C.

Bits 7:0	CLSIZ	*Cache line size* Hardwired to 0x04, indicating four 32-bit words in a cache line

7.6.4.2
Latency Timer

The 1-byte Latency Timer register is read/write and can be accessed at offset 0x10D.

Bits 10:8	MLTIM	*Master latency timer (low 3 bits)* Hardwired to 0
Bits 15:11	MLTIM	*Master latency timer* See the *PCI Local Bus Specification*, Sections 3.4.4.1 and 6.2.4.

The high 2 bytes in the word starting at offset 0x10E are reserved.

Bits 23:16	Reserved	*Hardwired to 0*
Bits 31:24	Reserved	*Hardwired to 0*

7.6.5
Mailbox Base Memory Addresses

The 1-word, read/write Mailbox Base Address register is accessed at offset 0x110 in the PCI configuration space header. This register must not be changed while an external agent is accessing one of the PCI mailboxes.

Bits 10:0	Reserved	*Hardwired to 0* Indicates that the controller's PCI Mailbox registers are located in a 32-bit memory space on a 4-KB boundary and are not prefetchable.
Bit 11	MBNUM	*0 = PCI Mailbox register 1*
Bits 31:12	MBADD	*Mailbox base memory address* Used to map the controller's two mailboxes into the PCI memory space on a 4-KB boundary.

7.6.6 Interrupt Line and Interrupt Pin Registers

The Interrupt Line and Interrupt Pin registers together constitute a word at offset 0x13C.

7.6.6.1 Interrupt Line Register

The 1-byte Interrupt Line register is read only and can be accessed at offset 0x13C.

Bits 7:0	INTLIN	*PCI Interrupt Line register* This field should be written by power-on self-test software to indicate which system interrupt controller input is connected to the INTA# signal.

7.6.6.2 Interrupt Pin Register

The 1-byte Interrupt Pin register is read only and can be accessed at offset 0x13D.

Bits 15:8	INTPIN	*PCI Interrupt Pin register* Reset to 1, indicating that INTA# is the controller's PCI interrupt signal.

The two high bytes in the word starting at offset 0x13D are reserved.

Bits 31:16	Reserved	*Hardwired to 0*

7.6.7 Retry Value and PCI Arbiter Priority Control Registers

The Retry Value and PCI Arbiter Priority Control registers are read/write. Together, these locations constitute the word at offset 0x140. They have the following fields.

Bits 7:0	Reserved	*Hardwired to 0*

7.6.7.1 Retry Value Register

The 1-byte Retry Value register can be accessed at offset 0x141.

Bits 15:8	RTYVAL	*Retry value* The number of retries the controller should attempt before giving up on a PCI transaction. The actual retry count is readable in the PCI retry counter (Section 7.8).
Bits 23:16	Reserved	*Hardwired to 0*

7.6.7.2
PCI Arbiter
Priority Control Register

The 2-bit PCI Arbiter Priority Control register can be accessed at offset 0x142.

Bits 25:24 PAPC — *PCI arbiter priority control*

00 = Rotating Fair. (This is the reset value.) In this scheme, the priority of each requester changes, in round-robin fashion, after each request to give each request a fair chance to acquire the bus. The rotation sequence begins with the controller's internal request, followed by requesters 0, 1, and 2 and back to an internal request. If any of the requesters is not active, the next requester in the sequence becomes the highest priority. After a requester has been granted the bus, it retains the bus, dependent on the TKYGNT bit.

01 = Rotating Alternate 0. In this scheme, REQ0# is granted the bus every other transaction, if asserted. The rotation sequence is 0, i, 0, 1, 0, 2, 0, 3, 0, i..., where i is the controller's internal request. After a requester is granted the bus, it retains the bus, dependent on the TKYGNT bit.

10 = Rotating Alternate 1. This scheme is identical to the Rotating Alternate 0 scheme, except that the controller's internal request, rather than REQ0#, has the advantage. The rotation sequence is i, 0, i, 1, i, 2, 0... After a requester is granted the bus, it retains the bus, dependent on the TKYGNT bit.

Bit 26 TKYGNT — *Take away grant*

0 = When REQn# is granted, it remains granted until the REQn# is negated. This is the reset value.

1 = When REQn# is granted, the bus loses GNTx# if it receives a higher priority request. A rotating priority scheme is used, so all requests are at a higher priority.

The high byte in the word starting at offset 0x140 is reserved.

Bits 31:27 Reserved — *Hardwired to 0*

7.7
PCI Configuration registers (Add-On Board Mode)

Table 29 shows the controller's PCI Configuration Space registers when the controller is is in Add-On Board mode (when the controller is located on a PCI bus add-on board, rather than on the system motherboard). As shown in Table 29, there are four more registers defined in the Add-On Board mode than in Host Bridge mode (see Table 28 on page 61).

Table 29. PCI Configuration Space registers (Add-On Board Mode)[1]

Offset from Base Memory Address 0F00_0000	Size (bytes)	Register Name	Symbol	CPU Bus R/W[2]	PCI bus R/W	Reset Value	Description	Reference
0x100	2	Vendor ID	VID	R	R	0x1033	Vendor ID for NEC	Section 7.6.1 on page 61
0x102	2	Device ID	DID	R	R	0x0095	VRC4375 controller's device ID, assigned by NEC	Section 7.6.1 on page 61
0x104	2	Command	PCICMD	R/W[3]	R/W[3]	0x0	Provides general control of PCI interface	Section 7.6.2 on page 62
0x106	2	Status	PCISTS	R/WC[4]	R/WC[4]	0x0280	Status for PCI events	Section 7.6.2 on page 62
0x108	1	Revision ID	RID	R	R	0x0	Device revision	Section 7.6.3 on page 63
0x109	3	Class Code	CLASS	R	R	0x06_0000	Device type	Section 7.6.3 on page 63
0x10C	1	Cache Line Size	CLSIZ	—	R	0x04	System cache line size (words)	Section 7.6.4 on page 64
0x10D	1	Latency Timer	MLTIM	R/W[3]	R/W[3]	0x0	Value of latency timer for this master, in PCI clock cycles	Section 7.6.4 on page 64
0x10E	2	*Reserved*		—	R	0x0		
0x110	4	Mailbox Base Address	MBADD	R/W	R/W	0x0	Base memory address for both mailboxes	Section 7.6.5 on page 64
0x114	4	Base Address Register 1	BAR1	—	R/W	0x8	Base Address Register 1, for target memory	Section 7.7.1 on page 68
0x118	4	Base Address Register 2	BAR2	—	R/W	0x8	Base Address Register 2, for target memory	Section 7.7.1 on page 68
0x11C	4	Base Address Register 3	BAR3	—	R/W	0x1	Base Address Register 3, for Add-On Board Interrupt Register	Section 7.7.2 on page 68
0x120	4	Base Address Register 4	BAR4	—	R/W	0x0	Base Address Register 4, for Add-On Board Interrupt Register	Section 7.7.3 on page 68
0x138 – 0x124		*Reserved*		—	—	0x0		
0x13C	1	Interrupt Line	INTLIN	—	R	0x0	PCI interrupt signal	Section 7.6.6 on page 65
0x13D	1	Interrupt Pin	INTPIN	—	R	0x0	PCI interrupt pin	Section 7.6.6 on page 65
0x13 – 0x13E	2	*Reserved*		—	R	0x0		
0x140	1	*Reserved*		—	R	0x0		
0x141	1	Retry Value	RTYVAL	—	R	0x0	Number of PCI bus retries the controller performs before giving up	Section 7.6.7 on page 65
0x142	2	PCI Arbiter Priority Control and Take Away Grant	PAPC	—	R	0x0	Priority scheme used in granting access to PCI bus	Section 7.6.7 on page 65
0x1FF – 0x144		*Reserved*		—	R	0x0		

Notes:
1. The shading/no shading row pattern defines 4-byte word boundaries.
2. — means not used.
3. Writable by the CPU if the PCI Enable register bit 1 is set to 1. Writable by the PCI host if this bit is set to 0. See Section 7.10 on page 70.
4. The bits can only be set by the controller hardware, and they are cleared by writing 1 to them. Writing 0 leaves them unaffected. For example, writing 0x8000 clears the most-significant bit.

7.7.1 BAR1 and BAR2 Registers

In the Add-On Board mode, after the controller's CONFIG_DONE bit in the PCI Enable register is set to 1, a PCI master can program the BAR1 and BAR2 registers and the controller will use them for the PCI Target Address Window ranges.

At reset, both registers are set to 0.

Bits 20:0	Pref	*Prefetchable* Hardwired to 0x8, indicating prefetchable, relocatable memory (see *PCI Specification*, Section 6.2.5.1). This field is not used in the Target Address Window address comparison.
Bits 31:21	Base	*PCI base memory address* This field is compared with the most-significant 11 bits of the PCI address, after masking bits 27:21 of this field with the corresponding PCI Target Address Window register mask (bits 19:13 of PCI Target Address Window Register 1 for BAR1, or PCI Target Address Window Register 2 for BAR2). The memory range can vary between 2 MB (no bits masked) and 256 MB (all bits masked). A match indicates that the access is to the controller. If the address is all 0s, this register is treated as disabled and memory is not allocated.

7.7.2 BAR3 Register

The BAR3 register contains the I/O address of the Add-On Board Interrupt register.

Bits 1:0	Space	*Space indicator* Hardwired to 01, indicating that the address is to I/O space.
Bits 31:2	IOAddr	*I/O address* The I/O address of the Add-On Board Interrupt register (see Section 7.7.4).

7.7.3 BAR4 Register

The BAR4 register is shown in Table 29. It contains the memory-mapped address of the Add-On Board Interrupt register.

Bits 3:0	Space	*Space indicator* Hardwired to 0, indicating that the address is to memory space.
Bits 31:4	MAddr	*Memory-mapped address* The memory-mapped address for the BAR3 I/O address.

7.7.4 Add-On Board Interrupt Register

The Add-On Board Interrupt register is located at the address specified in the BAR3 register (see Section 7.7.2). The register specifies the state of the INTA# signal.

Bit 0	PCI_INT	*PCI interrupt pending* 1 = PCI interrupt on INTA# pending 0 = PCI interrupt on INTA# not pending
Bits 31:1	Reserved	*Hardwired to 0*

When the CPU reads the SET_PCI_INT bit (bit 3) of the PCI Enable register (Section 7.10), the controller returns the value of bit 0 in the Add-On Board Interrupt register, which is the state of the INTA# signal. A PCI master causes the controller to negate INTA# by writing any value (1 or 0) to the PCI_INT bit (bit 0) of the Add-On Board Interrupt register.

7.8 PCI Retry Counter

The PCI Retry Counter is a read-only word at offset 0x70. It has only one status field.

Bits 4:0	RTRYCNT	*Retry count* The number of PCI bus transactions that the controller has retried. The maximum value for retries is set in the Retry Value register (Section 7.6.7.1 on page 65).
Bits 31:5	Reserved	*Hardwired to 0*

7.9 PCI Arbiter

The PCI bus arbiter determines access to the PCI bus for the controller and four other PCI bus masters. Four request/grant signal pairs are provided for the other masters; the controller has a fifth, internal request/grant function for its own requests. The arbiter implements three priority schemes, which are programmable in the PCI Arbiter Priority Control register (PAPC), described in Section 7.6.7.2 on page 66.

7.10 PCI Enable Register

The PCI Enable register is accessed at base memory address offset 0x74, as shown in Table 6. This register enables the PCI arbiter and the Add-On Board mode, specifies the completion of controller configuration, and sets and clears interrupts. At reset, all bits are set to 0.

Bit 0	ARB_ENABLE	*Enable arbiter* Enables the built-in PCI arbiter.
Bit 1	ADD_ON_BOARD	*Enable Add-On Board mode* In this mode, the VRC4375 controller is located on a PCI Add-On Board rather than on the motherboard.
Bit 2	CONFIG_DONE	*PCI configuration done* Software should set this to 1 after configuring the controller's other PCI registers. When set to 1, the controller responds normally to PCI operations. When cleared to 0, the controller responds to PCI target cycles with a retry.
Bit 3	SET_PCI_INT	*Assert PCI interrupt* Used only in Add-On Board mode. When set to 1 by the CPU, the controller sets bit 0 of the Add-On Board Interrupt register (Section 7.7.4) to 1 and asserts the PCI interrupt signal (INTA#). When the CPU reads SET_PCI_INT, the controller returns the value of bit 0 in the Add-On Board Interrupt register, which is the state of INTA#. The controller automatically clears SET_PCI_INT to 0 one clock after software sets it to 1, so there is no need for software to clear it. A PCI master causes the controller to negate INTA# by writing any value (1 or 0) to bit 0 of the Add-On Board Interrupt register.
Bit 4	RST_NMI	*Negate NMI#* Used only in Add-On Board mode. When set to 1, the controller negates its NMI# signal. The controller asserts NMI# whenever it detects that SERR# (PCI system error) is asserted. The NMI# service routine can read this bit to determine its state or set it to 1, which clears the interrupt. The controller automatically clears the bit to 0 one clock after software sets it to 1.
Bits 31:5	Reserved	*Hardwired to 0*

7.11 PCI Mailbox Registers

The controller has two PCI Mailbox registers for passing messages between the CPU and PCI bus masters:

- PCI Mailbox Register 1
- PCI Mailbox Register 2

Both registers are 1 word wide and may be read and written by either the CPU or a PCI bus master. From the CPU side, the addresses of the PCI Mailbox registers are at offsets 30 and 34, respectively (see Table 6). From the PCI bus side, the addresses are software configurable, as described in Section 7.5 and Section 7.6. The PCI Mailbox registers are mapped into PCI memory space and respond only to PCI memory cycles.

Both PCI Mailbox registers are cleared to 0 at reset. The registers respond as soon as the memory space enable (MEMEN) bit is set in the PCI Command register (Section 7.6.2); there is no enable function specific to these registers. If the Mailbox Base Address register (offset 0x110 in Table 28) is not initialized before the MEMEN bit (bit 1) is set in the Command register (offset 0x104 in Table 28), the base memory addresses for the two PCI Mailbox registers will be mapped to offsets 0x0 and 0x800, respectively, and may collide with other PCI devices.

When a PCI mailbox register is accessed from the PCI bus (either read or write), it causes a mailbox interrupt bit (MB1 or MB2) to be set in the controller's Interrupt Control and Status register (Section 9.1.2 on page 84). The interrupt is automatically cleared when the CPU reads or writes the corresponding PCI mailbox register.

7.12 Exclusive Access to PCI Bus Resources

As shown in Table 28 and Table 29, the controller provides a mechanism for obtaining exclusive (locked) access to PCI targets, as defined in the *PCI Local Bus Specification*, Section 3.6. As a master on the PCI bus, the controller can lock a specific target on the PCI bus, using the LOCK# signal.

To request exclusive access to a target, software sets bit 0 of the PCI exclusive access register (described immediately below) to 1. When this bit is set, the next PCI access uses the exclusive protocol, if possible, allowing the addressed resource to become locked to the requester through the controller. To release the target, software clears bit 0 prior to the last exclusive access; the current access remains exclusive until completed, at which time the target resource is released.

When the PCI bus target is locked, transactions are allowed only between the controller and the locked target. Transactions that do not complete are retried until they successfully complete. If the retry limit set in the Retry Value register (Section 7.6.7.1 on page 65) is reached, the controller sets bit 2 of the PCI Exclusive Access register to 1. If the controller receives an abort during a locked transaction, it sets bit 3 of the PCI Exclusive Access register to 1.

The controller can also perform exclusive accesses as a target. To configure this, software sets bit 1 of the PCI Exclusive Access register. When the controller senses that it is the target of a locked PCI bus cycle, it enters Locked mode. While in Locked mode, no other accesses to the controller, either from the PCI bus or from the CPU bus, are allowed until the master negates both FRAME# and LOCK#. However, refresh cycles are permitted to the DRAM system even while the memory is locked.

7.12.1 PCI Exclusive Access Register

The Exclusive Access register stores a read/write word at offset 0x60. It contains the following fields.

Bit 0	EAREQ	*Exclusive access request* 1 = Exclusive access request. In response, the controller asserts LOCK#, if conditions on the PCI bus permit (see Section 3.6 of the *PCI Local Bus Specification* for details). 0 = Releases the target; the controller negates LOCK# after completing the current access.
Bit 1	UNLOCK	*Controller is not a locked target* 1 = Disables controller as target of exclusive access 0 = Enables controller as target of exclusive access
Bit 2	RTRYREACHED	*Retry limit reached* 1 = Retry limit has been reached. The limit is set in the Retry Value register (Section 7.6.7.1 on page 65), and the retry count can be read in the PCI Retry Counter register (Section 7.8 on page 69). 0 = Retry limit has not been reached.
Bit 3	ABORT	*Abort received* 1 = Either a master abort or target abort has been received while the controller was asserting LOCK#. These aborts are described in Figures 3-4 and 3-10 of the *PCI Local Bus Specification*. 0 = No abort has been received.
Bits 31:4	Reserved	*Hardwired to 0*

8.0 DMA Transfers

The controller supports CPU-initiated DMA transfers between memory and the PCI bus. These transfers can be unaligned reads or writes at the full PCI rate of 133 MB/s. Four sets of CPU-programmed registers configure DMA transfers; during transfers, other sets of registers can be read or written to. The DMA FIFO, an 8-word (32-byte) bidirectional FIFO, temporarily stores PCI-to-memory or memory-to-PCI transfers inside the controller.

To initiate a DMA transfer, the CPU configures the controller's DMA registers (Section 8.3) with the memory address, PCI bus address, read/write transfer direction, boundary crossing points, end-of-transfer interrupt enable, and transfer enable. Once configured, the controller arbitrates for the memory and PCI bus, then performs the transfer independently of the CPU.

PCI bus masters cannot initiate DMA transfers. Instead, such masters gain access to the controller's memory through PCI Target Address Window registers, described in Section 7.4.

8.1 DMA Operations

DMA transfers can be from the PCI bus to memory (called a PCI read), or from memory to the PCI bus (called a PCI write). The direction is set in the R/W bit (bit 29) of the DMA Control registers.

8.1.1 PCI to Memory Transfers (PCI Read)

For a PCI bus read (from the PCI bus to memory), the controller begins by requesting access to the PCI bus. When granted, the controller reads words from the PCI bus and stores them in its 8-word DMA FIFO. When the FIFO contains 4 words (the FIFO is half full), the controller requests access to memory, which is granted as soon as any current CPU memory operation completes. Then the controller empties data from the FIFO to memory at the fastest rate supported by memory.

If the controller's DMA FIFO becomes full during a transfer, the controller releases the resource responsible for filling the FIFO until the FIFO is emptied to 4 words (the FIFO is half full). Then the controller reacquires the resource and continues filling the FIFO.

8.1.2 Memory to PCI Transfers (PCI Write)

For a PCI bus write (from memory to the PCI bus), the controller begins by requesting access to memory. When granted, the controller reads the first 8 words into its DMA FIFO at the fastest rate supported by memory.

The controller accumulates 4 words in its FIFO before requesting the PCI bus. The controller attempts to empty the FIFO as quickly as the PCI target can accept the data. Meanwhile, the controller attempts to keep its FIFO full. If the FIFO becomes full, the controller releases memory until the FIFO reaches 4 words (the FIFO is half full), at which time it again accesses memory and begins filling the FIFO.

If the FIFO becomes empty, the controller issues a disconnect command to the PCI bus. If there is more data to transfer in the same DMA operation, the controller continues filling its FIFO from memory and accesses the PCI bus when either one word or four words have been accumulated in the FIFO, depending on the memory type described above. When the correct number of words has been read from memory, the controller stops filling its FIFO but continues emptying the FIFO until the last transfer completes.

8.1.3 Transfer Completion

When a DMA transfer completes, the controller interrupts the CPU (if INT# interrupts are enabled), with bit 1, 2, 16, or 17 set in the controller's Interrupt Control and Status register (Section 9.1.2). The controller then checks the other set of DMA Control registers to determine if another DMA transfer is pending. If another is pending, the controller allows one pending CPU-to-memory and one pending CPU-to-PCI transaction to complete before beginning the next DMA transfer. If there is a next record address already in place in the register and a chaining-enable bit is set (DMA/PCI/CPU-Memory Arbitration Priority register at offset 0x80, bits 11:8), then the DMA controller will automatically fetch the next record from the memory, load it in the controller, and start a new DMA transaction. It will continue the transaction until the Next Record Address register is zero or the chaining-enable bit is reset before starting a new transaction.

When chaining and interrupt are both enabled, the DMA transfer complete interrupt will occur only after completion of the last DMA structure in the chain.

If the controller receives a PCI master-abort or target-abort termination, the controller resets the DMA channel, indicates the error type by setting bits 1:0 of the Bus Error Status register (Section 9.1.1), and interrupts the CPU (if INT# interrupts are enabled). If a DMA bus error occurs, the controller interrupts the CPU (if INT# interrupts are enabled), with bit 5 set in the Interrupt Control and Status register (Section 9.1.2). If the other DMA channels are enabled to begin a transfer (bit 28 is set in the other DMA Control registers), the controller begins the pending transfer.

8.2 CPU Access During DMA Transfers

After a DMA transfer starts, the CPU cannot access memory until the DMA reaches a boundary crossing point in the memory address space. The boundary crossing point is programmed by the CPU, as defined in the DMA Control registers. The controller allows the CPU to perform one memory transaction at each boundary crossing point. Thus, a CPU memory read stalls between boundary crossing points, but a CPU memory write will be buffered in the CPU-to-memory write FIFO. When the write FIFO contains a posted write, all other CPU-to-memory transactions stall (EOK# negated) until a boundary crossing point is reached or the DMA transaction completes.

If the CPU attempts to read an address mapped to the PCI bus during a DMA transfer, the CPU read stalls until the DMA transfer completes. If the CPU attempts to write to an address mapped to the PCI bus address during a DMA transfer, the CPU write is posted in the PCI master FIFO until the DMA transfer completes. When the PCI master FIFO contains a posted write, all other CPU transactions stall (EOK# is deasserted) until the FIFO is empty.

8.3 DMA Registers

The controller has four sets of DMA Configuration registers, each of which controls a DMA transfer. One set of registers may be read from or written to while the other set is controlling a DMA transfer. The Configuration registers are:

- DMA Control registers 1, 2, 3, and 4
- DMA Memory Address Registers 1, 2, 3, and 4
- DMA PCI Address Registers 1, 2, 3, and 4
- DMA Next Record Address Register

In addition to these Configuration registers, the controller also has the following DMA Status registers:

- DMA Words Remaining register
- DMA Current Memory Address register
- DMA Current PCI Address register

The registers are located at offsets 0x38 through 0x4C, offsets 0xA4 through 0xB8, and offsets Ox64 through 0x6C, as shown in Table 6 on page 14. The following sections describe the contents of these registers.

8.3.1 DMA Control Registers 1, 2, 3, and 4

The CPU uses these registers to configure DMA transfers. One register can be read from or written to while the other is controlling a DMA transfer. When a DMA transfer has started, remaining bits in the DMA Control registers, except the DRST and SU bits (bits 24 and 27), have no effect. The CPU should not try to write to other bits unless a DMA operation is in progress.

The registers are 4 bytes wide, at offsets 0x38, 0x44, 0xA4, and 0xB0. They are set to 0 at reset and contain the following fields.

Bits 19:0 BlkSize

Block size
The number of bytes (up to 1 MB) to be transferred.
0 = 1 MB

Bits 23:20 BoundPnt

Boundary crossing point
The address boundary at which CPU memory transactions may be performed during a DMA transaction. When the current DMA memory address matches this boundary, as defined in this field, the controller allows the CPU to perform one memory transaction. Boundaries are defined in the table below.

BoundPnt Field	Byte Address Boundary
0000	None
0001	32
0010	64
0011	128
0100	256
0101	512
0110	1 K
0111	2 K
1000	4 K
1001	8 K
1010	16 K
1011	32 K
1100	64 K
1101	128 K
1110	256 K
1111	512 K

Bit 24 DRST

DMA reset
1 = Terminates an in-progress DMA transfer and resets the DMA logic, after completion of the current bus cycle. This bit takes precedence over all other bits in the DMA Command register. The value written to the other bits of this register when DRST is 1 is irrelevant: This bit takes precedence.
0 = The controller clears this bit automatically after the DMA channel has been reset.

Bit 25 MIO

PCI memory or I/O
1 = Transfers data to or from PCI memory space
0 = Transfers data to or from PCI I/O space

Bit 26	INC	*Increment PCI address* 1 = Increments the PCI address as the DMA transfer is performed. 0 = Restarts from the original starting address for any condition that causes the DMA to restart a PCI burst. The starting address is programmed in the DMA Memory Address register or the DMA PCI Address register.
Bit 27	SU	*Suspend DMA* 1 = Suspends the current DMA transfer after completion of the current PCI cycle. All register values are preserved. 0 = Restarts the suspended DMA transfer. This bit may be set and cleared without consideration of the other bits in the DMA Control registers, except DRST (bit 24). That is, when the DMA transfer has started, changing bits other than SU and DRST has no effect.
Bit 28	GO	*Begin transfer* 1 = Starts the DMA transfer as soon as the PCI and memory buses are available. 0 = The controller automatically clears this bit after the transfer completes. Software-clearing this bit has no effect; the DMA transfer will continue. This bit must not be set if the bus master enable bit (bit 2) in the PCI Command register (Section 7.6.2.1) has not been previously set.
Bit 29	R/W	*PCI read/write direction* 1 = Reads data from the PCI bus and writes it to local memory. 0 = Reads data from local memory and writes it to the PCI bus.
Bit 30	IE	*Interrupt enable* 1 = When a DMA transfer completes and if INT# interrupts are enabled, interrupts the CPU with bit 1 or 2 set in the controller's Interrupt Control and Status register (see Section 9.1.2). 0 = The controller does not interrupt the CPU on completion of the DMA transfer.
Bit 31	BZ	*Busy (read only)* 1 = The DMA transfer controlled by this register is currently in process. This bit may be polled. 0 = No DMA transfer controlled by this register is in process.

8.3.2 DMA/CPU/PCI to Memory Arbitration Priority Selection Register

This register is programmed by the CPU at offset 0x80. It selects priority schemes from among four DMA channels for memory accesses. Bits 13:8 select priority and link list chaining enable/disable, as described below. The default value is 0x0009 at reset, which selects the arbiter mode giving priority to the PCI slave for two consecutive accesses to the CPU.

Bits 7:0	Reserved	*Hardwired to 0*
Bits 11:8	DMA chaining En	*Link list chaining* Chaining of the link list stored in local memory associated with each DMA channel can be enabled or disabled. The DMA transfer takes place per one control structure stored in their respective registers in the device. If the chaining-enable bit is reset, transfer is considered complete and no DMA control structure is fetched from the memory; thus, no additional DMA transfers take place. These enable bits can be used to control DMA operations in conjunction with the Next Record Address register (Section 8.3.5). The value of the Next Record Address register stored in memory needs to be 4-word aligned only during chaining operations. For fastest DMA operations, set the values in the DMA/PCI Address register, DMA/Memory Address register, and Next Record Pointer register to be word aligned. Link list chaining is automatically stopped when the contents of this register are zero. bit 8 = 1, DMA channel 1 chaining enabled bit 8 = 0, DMA channel 1 chaining disabled bit 9= 1, DMA channel 2 chaining enabled bit 9 = 0, DMA channel 2 chaining disabled bit 10 = 1, DMA channel 3 chaining enabled bit 10 = 0, DMA channel 3 chaining disabled bit 11 = 1, DMA channel 4 chaining enabled bit 11 = 0, DMA channel 4 chaining disabled

Bits 13:12	DMA arbit priority	*DMA arbitration priority* Four DMA channels comprise groups A and B: Group A includes channels 1 and 2; Group B includes channels 3 and 4. The DMA arbiter working between these two groups issuing DMA requests for memory accesses (DMA memory requests to be arbitrated by the DMA/PCI/CPU Memory Arbitration register) can be programmed with different priority schemes. 00 = Arbitration priority sequence: 1, 2, 3, 4, 1, 2, 3, 4, ... 01 = Arbitration priority sequence: 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 4, ... 10 = Arbitration priority sequence: 3, 4, 3, 4, 1, 2, 3, 4, 3, 4, 1, 2, ... **Example:** Assuming DMA channel requests are prioritized as the arbitration sequence: 1, 2, 3, 4, 1, 2, 3, 4, ... and bits 13:12 = 01, arbitration is performed as follows. Channel 1 is allowed to complete one DMA transaction, then channel 2, then channel 1 again, and then channel 2, then 3 and 4 and so on. After each DMA transaction per a control structure (either written to the registers by the CPU or fetched by the DMA controller from memory when chaining is enabled) is complete, DMA requests from other channels are arbitrated and granted per the arbitration scheme.
Bits 31:14	Reserved	*Hardwired to 0*

8.3.3 DMA Memory Address Registers 1, 2, 3, and 4

These registers are programmed by the CPU with the starting memory address for the transfer. The registers are at offsets 0x3C, 0x48, 0xA8, and 0xB4. At reset, they are set to 0x0.

Bits 31:0	Local Address	*Memory starting address* The starting address to be used when accessing the controller's memory. This field remains static throughout the DMA transfer. The current memory address being accessed can be read from the DMA Current Memory Address register (Section 8.3.7).

8.3.4 DMA PCI Address Registers 1, 2, 3, and 4

These registers are programmed by the CPU with the starting PCI bus address for the transfer. The registers are at offsets 0x40, 0x4C, 0xAC, and 0xB8. At reset, these registers are set to 0.

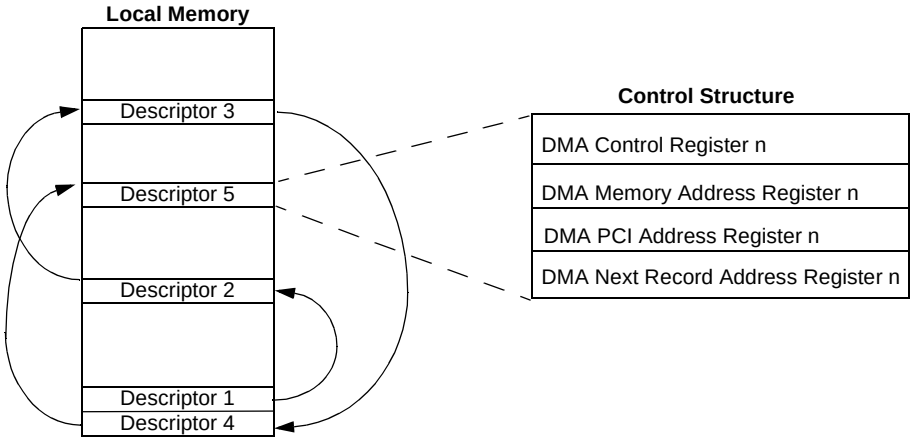
Bits 31:0 LocalAddr — *PCI bus starting address*
The starting address to be used when accessing the PCI bus. This field remains static throughout the DMA transfer. The current PCI bus address being accessed can be read from the DMA Current PCI Address register (Section 8.3.8).

8.3.5 DMA Next Record Address Registers 1, 2, 3, and 4

These registers are programmed by the CPU. The registers are at offsets 0xBC, 0xC0, 0xC4 and 0xC8, and are set to 0x0 at reset.

Bits 31:0 NextAddr — *Next record pointer*
Address where the subsequent list of control structures or descriptors (list is 4 contiguous words each) reside. A value of 0 indicates the end of the structure chain. This field remains static throughout the DMA transfer. The list gets fetched directly by the DMA controller from the local memory. The CPU can write new structure data in the local memory prior to the new structure data being fetched by the DMA controller. The CPU can also write to all structures in the local memory prior to starting a DMA operation. The control structures may or may not be contiguously located in the memory. The start address should be on a word boundary only when the Next Record Pointer bits are used for link list operations (Figure 9).

Figure 9. DMA Scatter/Gather Using Link List



8.3.6 DMA Words Remaining Register

The CPU reads this register to determine the number of words remaining in the current DMA transfer. The register is at offset 0x64; at reset it is set to 0x0.

Bits 31:0	WordCnt	*Words remaining* (read only) The number of words remaining to be transferred during DMA operation

8.3.7 DMA Current Memory Address Register

The CPU reads this register to determine the memory address accessed during a DMA transfer. The register is at offset 0x68; at reset it is set to 0x0.

Bits 31:0	CrntAddr	*Current memory address* (read only) The current memory address during DMA operation

8.3.8 DMA Current PCI Address Register

The CPU reads this register to determine the PCI address accessed during a DMA transfer. The register is at offset 0x6C; at reset it is set to 0x0.

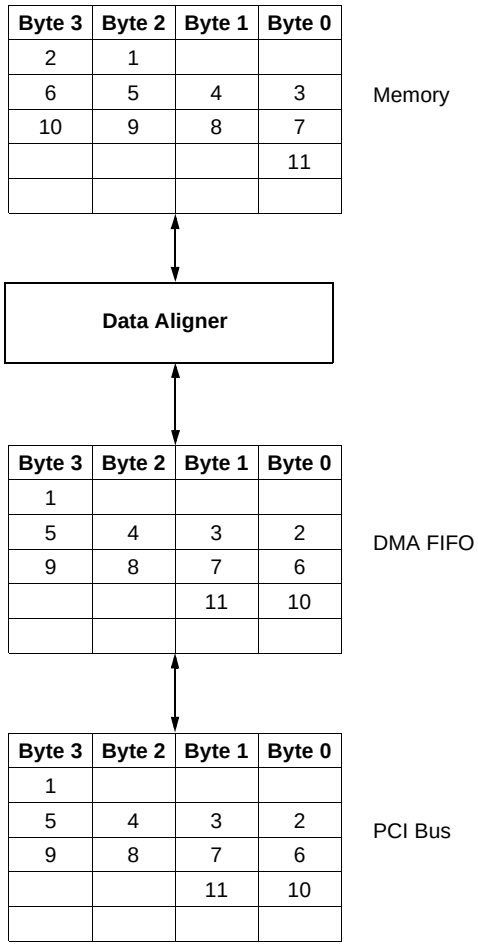
Bits 31:0	CrntAddr	*Current PCI address* (read only) The current PCI address during DMA operation

8.4 Data Aligner

The controller automatically handles unaligned bidirectional transfers between the PCI bus and memory. The aligner shifts byte data into the DMA FIFO or memory, in the alignment required or supplied by the PCI bus (Figure 10). The aligner permits the controller to use high-speed burst protocols for transfers, even when both the source and destination addresses are not aligned on word/address boundaries or with each other.

Figure 10 shows the operation of the aligner for a DMA transfer from byte address 0002 in memory to byte address 0003 on the PCI bus (or in the opposite direction).

Figure 10. DMA Transfer Alignment Example



9.0 Interrupts

The controller supports maskable interrupts using the INT# input signal to the CPU, and nonmaskable interrupts using the NMI# input signal to the CPU.

9.1 Maskable Interrupts (INT#)

The controller can be enabled to interrupt the CPU when the following types of memory or PCI bus errors occur.

- *Illegal address errors:* Memory accesses by the CPU to physical addresses outside of one of the five memory ranges, one of the three PCI windows, or one of the controller's internal registers
- *Target abort, master abort, and retry limit errors:* PCI bus accesses by the CPU that result in a target abort, master abort, or more retries than specified by the Retry Value register

The controller reports errors to the CPU by asserting the INT# signal, if enabled by bit 0 in the Interrupt Control and Status register (Section 9.1.2). The CPU's interrupt service routine can then read the Bus Error Status register (Section 9.1.1) to determine the type of error. The INT# signal is a level-sensitive output to the CPU and may not be shared with other interrupt generators. The controller does not prioritize the various interrupt sources.

During CPU reads, any detected errors cause the controller to return the correct number of data words. However, the bus error bit is set in SysCmd[0] for those words that are returned after the word that caused the bus error.

For DMA accesses to controller memory that miss the configured memory ranges, the Bus Error Status register contains the error information, just as for errors during CPU accesses. DMA accesses to the PCI bus that result in a target abort, master abort, or more retries than specified set the error type field in the Bus Error Status register but not the error address field. Bus errors generated by the DMA cause the DBE interrupt to be generated, if enabled.

External PCI accesses that hit either target window, but miss all internal controller resources, will set the ET code to 0 and the error address. The error address will be the translated address. External PCI accesses can never set an ET code other than 00. This bus error sets the PBE interrupt, if enabled.

9.1.1 Bus Error Status Register

The CPU reads this register when the CPU detects a bus error interrupt from the controller to determine the cause of the error. The contents remain constant after the error, until read by the CPU. The register is at offset 0x50; at reset, it is set to 0x50. This register contains the following fields.

Bits 1:0	ET	*Error type* (read only) 00 = Illegal address 01 = Target abort received 10 = Master abort signaled 11 = Retry limit reached. The value specified in the Retry Value register (Section 7.6.7.1) has been reached.
Bits 31:2	EA	*Error address* (read only) The most-significant 30 bits of the local (controller side) physical address that caused the error. This field is valid for CPU and DMA unit accesses to the controller's memory. It is not valid for DMA unit accesses to the PCI bus memory space.

9.1.2 Interrupt Control and Status Register

The Interrupt Control and Status register is read only in its lower byte, read/write in its middle two bytes, and write only in the high byte. The low byte should be read by the CPU, along with the Bus Error Status register, when the CPU detects an INT# interrupt from the controller. The contents of the Interrupt Control and Status register remain constant after the error, until read by the CPU. The register is at offset 0x50; it is set to 0 at reset.

Bit 0	CBE	*CPU bus error* (read only) 1 = CPU bus error 0 = No such error
Bit 1	DMA1	*DMA channel 1 complete* (read only) 1 = Transfer specified in DMA Control Register 1 is complete 0 = Transfer 1 is not complete
Bit 2	DMA2	*DMA channel 2 complete* (read only) 1 = Transfer specified in DMA Control Register 2 is complete 0 = Transfer 2 is not complete
Bit 3	MB1	*PCI Mailbox 1 accessed* (read only) 1 = Mailbox 1 accessed from the PCI bus 0 = Mailbox 1 not accessed
Bit 4	MB2	*PCI Mailbox 2 accessed* (read only) 1 = Mailbox 2 accessed from the PCI bus 0 = Mailbox 2 not accessed
Bit 5	DBE	*DMA bus error* (read only) 1 = A bus error occurred during a DMA transfer 0 = No bus error

Bit 6	PBE	*PCI bus error* (read only) 1 = A bus error occurred during a PCI target access 0 = No bus error
Bit 7	PAR	*PCI parity error* (read only) 1 = Parity error. The error can be on an address parity during a target cycle, a data parity during a target write cycle, or a data parity during a master read cycle. 0 = No parity error
Bit 8	CBEmsk	*CPU bus error enable* (read/write) 1 = Enables CPU bus error interrupts 0 = Disables CPU bus error interrupts
Bit 9	DMA1msk	*DMA channel 1 complete enable* (read/write) 1 = Enables DMA channel 1 complete interrupts 0 = Disables DMA channel 1 complete interrupts
Bit 10	DMA2msk	*DMA channel 2 complete enable* (read/write) 1 = Enables DMA channel 2 complete interrupts 0 = Disables DMA channel 2 complete interrupts
Bit 11	MB1msk	*PCI mailbox 1 access enable* (read/write) 1 = Enables PCI mailbox 1 accessed interrupts 0 = Disables PCI mailbox 1 accessed interrupts
Bit 12	MB2msk	*PCI mailbox 2 access enable* (read/write) 1 = Enables PCI mailbox 2 accessed interrupts 0 = Disables PCI mailbox 2 accessed interrupts
Bit 13	DBEmsk	*DMA bus error enable* (read/write) 1 = Enables DMA bus error interrupts 0 = Disables DMA bus error interrupts
Bit 14	PBEmsk	*PCI bus error enable* (read/write) 1 = Enables PCI bus error interrupts 0 = Disables PCI bus error interrupts
Bit 15	PARmsk	*PCI parity error enable* (read/write) 1 = Enables PCI parity error interrupts 0 = Disables PCI parity error interrupts
Bit 16	DMA3	*DMA channel 3 complete* (read only) 1 = Transfer specified in DMA Control Register 3 is complete 0 = Transfer 3 is not complete
Bit 17	DMA4	*DMA channel 4 complete* (read only) 1 = Transfer specified in DMA Control Register 4 is complete 0 = Transfer 4 is not complete
Bit 18	DMA3msk	*DMA channel 3 complete enable* (read/write) 1 = Enables DMA channel 3 complete interrupts 0 = Disables DMA channel 3 complete interrupts

Bit 19	DMA4msk	*DMA channel 4 complete enable* (read/write) 1 = Enables DMA channel 4 complete interrupts 0 = Disables DMA channel 4 complete interrupts
Bit 20	DMA3clr	*DMA channel 3 complete clear* (write only) 1 = Clears the DMA channel 3 complete interrupt. Always returns 0 when read.
Bit 21	DMA4clr	*DMA channel 4 complete clear* (write only) 1 = Clears the DMA channel 4 complete interrupt. Always returns 0 when read.
Bit 22	UARTINT	*UART interrupt to the CPU* (read only) 1 = A UART interrupt to the CPU is active (always returns 0 when read). 0 = No UART interrupt The source of this interrupt is any of the four interrupt sources described in Section 6.12.
Bit 23	UARTINTEN	*UART interrupt to the CPU* (read/write) 1 = Enables all UART interrupt sources (always returns 0 when read) 0 = Disables all UART interrupt sources This bit is a global enable for all UART interrupt sources that are individually enabled in the UART Interrupt Enable register (UARTIER), Section 6.12. Clearing all bits in UARTIER would clear the UART interrupts, thus clearing UARTINT (bit 22).
Bit 24	CBEclr	*CPU bus error clear* (write only) 1 = Clears the CPU bus error interrupt. Always returns 0 when read.
Bit 25	DMA1clr	*DMA channel 1 complete clear* (write only) 1 = Clears the DMA channel 1 complete interrupt. Always returns 0 when read.
Bit 26	DMA2clr	*DMA channel 2 complete clear* (write only) 1 = Clears the DMA channel 2 complete interrupt. Always returns 0 when read.
Bit 27	MB1clr	*PCI Mailbox 1 access clear* (write only) 1 = Clears the PCI mailbox 1 accessed interrupt. Always returns 0 when read.
Bit 28	MB2clr	*PCI Mailbox 2 access clear* (write only) 1 = Clears the PCI mailbox 2 accessed interrupt. Always returns 0 when read.
Bit 29	DBEclr	*DMA bus error clear* (write only) 1 = Clears the DMA bus error interrupt. Always returns 0 when read.
Bit 30	PBEclr	*PCI bus error clear* (write only) 1 = Clears the PCI bus error interrupt. Always returns 0 when read.
Bit 31	PARclr	*PCI parity error clear* (write only) 1 = Clears the PCI parity error interrupt. Always returns 0 when read.

9.1.3 Timers/PCI INTA# Interrupt Control and Status Register 2

The Timers/PCI INTA# Interrupt Control/Status Register 2 has three timers: the Set NMI Timer, Set Timer Counter, and Read Timer Counter. These timers are clocked at the CPU bus clock rate. The frequency of the timer clock is programmable; it can be slowed down to 1/8 of the CPU clock so that slower events can be timed. All three timers are readable and writable by the CPU. Timers can be read by the CPU while they are counting. They can be automatically reloaded with the previously loaded value and restarted or can be stopped while in progress. All three timers issue interrupts (which can be enabled/disabled) to the CPU upon reaching their maximum value.

The Timers/PCI INTA# Interrupt Control/Status Register 2 resides at offset 0xE4. Bits 10:8 indicate the end of the timer count; when set, the bits indicate there is a timer event that completed, causing an interrupt pending bit to set. This register also contains PCI INTA# control bits. This input into the PCI interface allows external PCI devices to interrupt the CPU. Upon receiving an interrupt from the controller, they are read by the CPU to ascertain the interrupting device. The contents of the Interrupt Control and Status register remain constant after the interrupt, until read and cleared by the CPU. The timers are clocked at the CPU bus clock rate or selectable clock frequencies (bits 29:24), so timing calculation must be made accordingly. All timers count up. The write registers (Set Timer registers) have a different offset from the read registers (Read Timer Counter registers), so write registers are not affected while a value is read from the read registers, which indicate a running count of the timer/counter. Once a value is loaded in the Set Timer registers, it stays there until the timer's interrupts are cleared. In the Timer Interrupt Control and Status register, the original value can be reloaded in the counter to restart it from that count, if the Tn reload enable bit is set.

Interrupts are automatically cleared when the CPU reads this register following assertion of an interrupt pending bit.

9.1.3.1 Set NMI Timer Register

This register is read/writable by the CPU. The CPU loads a value in it and the counter starts counting up. When it reaches 0xFFFF FFFF, it generates an interrupt to the CPU, provided appropriate control bits are set. See the bit description below for the Timer Interrupt Control and Status register.

9.1.3.2 Set Timer Counter Register 1

This register is read/writable by the CPU. The CPU loads a value in it and the counter starts counting up. When it reaches 0xFFFF FFFF, it generates an interrupt to the CPU, provided appropriate control bits are set. See the bit description below for the Timer Interrupt Control and Status register.

9.1.3.3 Set Timer Counter Register 2

This register is read/writable by the CPU. The CPU loads a value in it and the counter starts counting up. When it reaches 0xFFFF FFFF, it generates an interrupt to the CPU, provided appropriate control bits are set. See the bit description below for the Timer Interrupt Control and Status register.

9.1.3.4 Read NMI Timer Counter Register

This register is read only by the CPU. The CPU can read its value to get the timer count.

9.1.3.5
Read Timer Counter Register 1

This register is read only by the CPU. The CPU can read its value to get the timer count.

9.1.3.6
Read Timer Counter Register 2

This register is read only by the CPU. The CPU can read its value to get the timer count.

The Timer Interrupt Control and Status register is set to 0 at reset and contains the following fields.

Bit 0	NMITEN	*NMI timer enable* (read/write) 1 = Enables, starts the timer 0 = Disables, stops the timer
Bit 1	T1EN	*Timer 1 enable* (read/write) 1 = Enables, starts the timer 0 = Disables, stops the timer
Bit 2	T2EN	*Timer 2 enable* (read/write) 1 = Enables, starts the timer 0 = Disables, stops the timer
Bit 3	PCIINTAEN	*PCI interrupt (PCI INTA#) enable* (read/write) 1 = Enables PCI INTA# input for interrupting the CPU 0 = Disables PCI INTA# input from interrupting the CPU; the interrupt will be pending. In the event the interrupt enable bit is not set, the INTA# signal is latched internally and will not be released until cleared by the CPU. The CPU is interrupted only after this bit is set.
Bit 4	PCIINTCLR	*PCI interrupt A# clear* (write only) 1 = Clears the pending interrupt 0 = Do not care; no action
Bit 5	PCIINT	*PCI interrupt A#* (read only) 1 = PCI interrupt A# pending 0 = No PCI interrupt A# pending
Bits 7:6	Reserved	*Hardwired to 0x00*
Bit 8	NMIINT	*Nonmaskable interrupt* (read only) 1 = Nonmaskable interrupt pending 0 = No nonmaskable interrupt pending If this bit is set while the CPU reads this register, a clear pulse clears it. If the clock edge where this pulse is generated coincides with the clock edge where a new interrupt is to be recorded, then setting of this bit takes priority (this bit is not cleared). It is cleared in a subsequent read by the CPU. This also applies to bits 9 and 10.

Bit 9	T1INT	*T1 interrupt* (read only) 1 = T1 interrupt pending 0 = No T1 interrupt pending
Bit 10	T2INT	*T2 interrupt* (read only) 1 = T2 interrupt pending 0 = No T2 interrupt pending
Bit 11	Reserved	*Hardwired to 0*
Bit 12	NMIINTEN	*NMI interrupt enable* (read/write) 1 = Enables NMI interrupt 0 = Disables NMI interrupt
Bit 13	T1INTEN	*T1 interrupt enable* (read/write) 1 = Enables T1 interrupt 0 = Disables T1 interrupt
Bit 14	T2INTEN	*T2 interrupt enable* (read/write) 1 = Enables T2 interrupt 0 = Disables T2 interrupt
Bit 15	Reserved	*Hardwired to 0*
Bit 16	NMITRELOAD	*NMI timer reload* (read/write) 1 = Automatically reloads the original timer value and starts if NMITEN is set 0 = Does not reload
Bit 17	T1RELOAD	*T1 reload* (read/write) 1 = Automatically reloads the original timer value and starts if T1EN is set 0 = Does not reload
Bit 18	T2RELOAD	*T2 reload* (read/write) 1 = Automatically reloads the original timer value and starts if T2EN is set 0 = Does not reload
Bits 23:19	Reserved	*Hardwired to 0*
Bits 25:24	NMITFR	*NMI timer clock frequency selection* 00 = CPU bus clock rate 01 = CPU bus clock rate divided by 2 10 = CPU bus clock rate divided by 4 11 = CPU bus clock rate divided by 8
Bits 27:26	T1FR	*Timer 1 clock frequency selection* 00 = CPU bus clock rate 01 = CPU bus clock rate divided by 2 10 = CPU bus clock rate divided by 4 11 = CPU bus clock rate divided by 8
Bits 29:28	T2FR	*Timer 2 clock frequency selection* 00 = CPU bus clock rate 01 = CPU bus clock rate divided by 2 10 = CPU bus clock rate divided by 4 11 = CPU bus clock rate divided by 8

Bits 31:30 Reserved *Hardwired to 0*

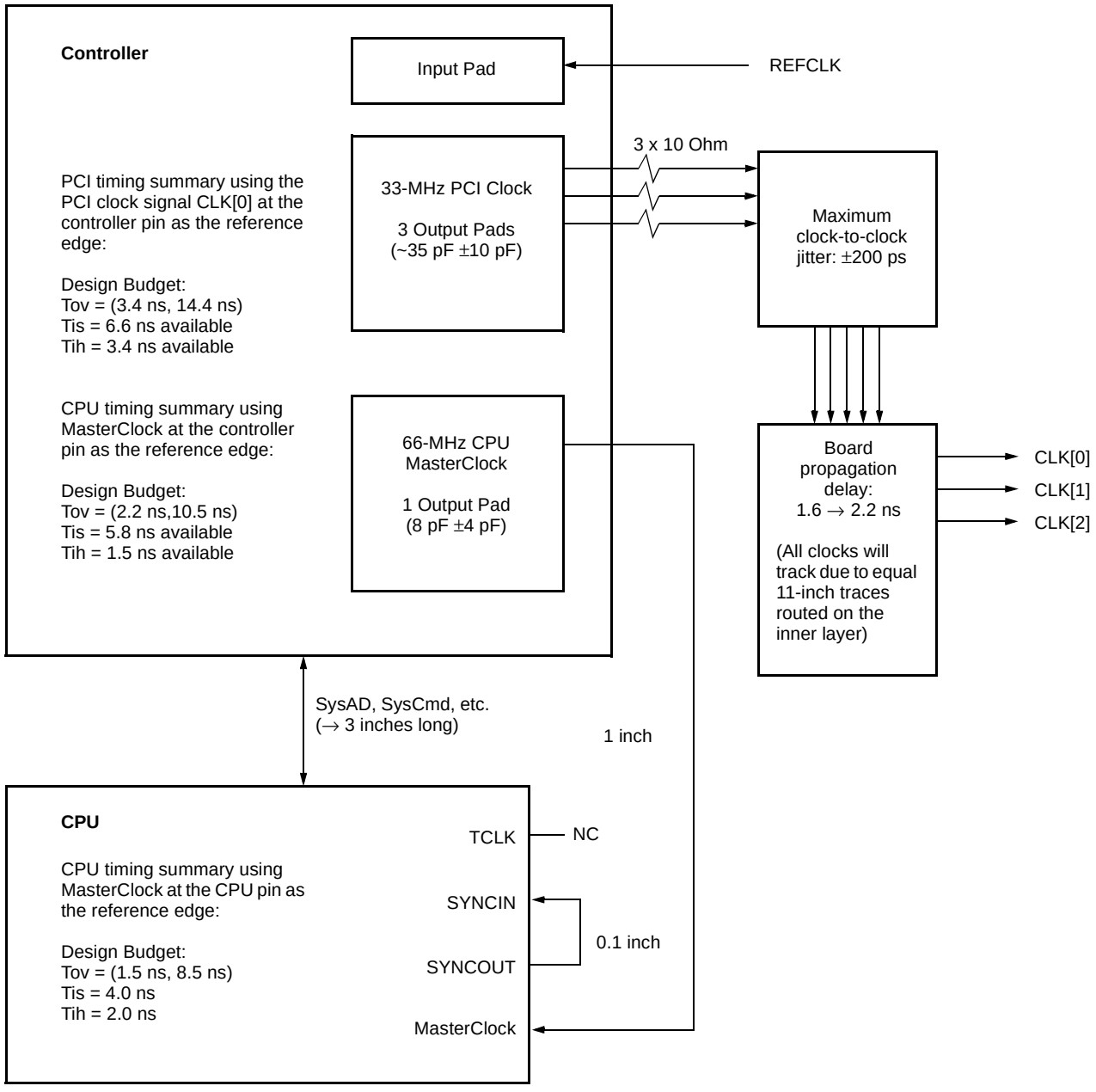
9.2 Nonmaskable Interrupts (NMI#)

The controller asserts NMI# when a PCI device asserts SERR#. It also asserts NMI# when the NMI interrupt timer times out.

10.0 Clocking

The controller receives a 66-MHz oscillator reference clock (REFCLK) signal and distributes the 66-MHz MasterClock signal to the CPU. The controller also generates and distributes four copies of the 33-MHz PCI clock (CLK[3:0]). Figure 11 shows the controller's clock connections with the system.

Figure 11. Clock Connections



11.0 Reset Configuration Signals

The rising edge of the PCI bus reset signal (RST#) serves as the controller's reset. Table 30 lists the configuration signals that the controller samples for one REFCLK edge while RST# is active.

Table 30. Reset Configuration Signals

MuxAD Signals	Function	Description
MuxAD[2:0]	Boot ROM size	Table 14 on page 24
MuxAD[7:3]	Boot ROM write protect	Section 6.5.3 on page 26
MuxAD[8]	Flash memory boot enable 1 = Enable 0 = Disable	Section 6.5.4 on page 26
MuxAD[9]	Not used	
MuxAD[10] (Legacy)	Processor interface (legacy) Endian byte order 1 = Big endian 0 = Little endian	
MuxAD[11]	PCI interface endian byte order 1 = Big endian 0 = Little endian	
MuxAD[12]	Processor interface (new) Endian byte order 1 = Big endian 0 = Little endian Legal combinations allowed: Bits 12:10 = 001, 100, 110, 010	
MuxAD[14:13]	Boot ROM width 00 = 8 bits, BBE0# asserted 01 = 16 bits, BBE0#,1# asserted 10 = 32 bits, BBE[3:0]# asserted 11 = not used Writes to flash memory should be done in word format. Reads from flash memory can be on any of the above boundaries.	
MuxAD[15]	Reserved	
MuxAD[19:18]	Flash/boot ROM size in SIMM slot 2 00 = 8 MB 01 = 16 MB 10 = 32 MB 11 = 64 MB	
MuxAD[20]	Base and SIMM memory device access time for subsequent words after initial word (this is in addition to CAS latency selection as defined in the control registers in Section 6.0). 1 = Two cycles 0 = Three cycles	

12.0 Endian Mode Software Issues

12.1 Overview

Endian mode refers to a device's word-addressing method and byte order. *Big-endian* devices address data items at the *big end* (most-significant bit number). The most-significant byte (MSB) in an addressed data item is at the *lowest* address. *Little-endian* devices address data items at the *little end* (least-significant bit number). For a little-endian device, the most-significant byte (MSB) in an addressed data item is at the *highest* address.

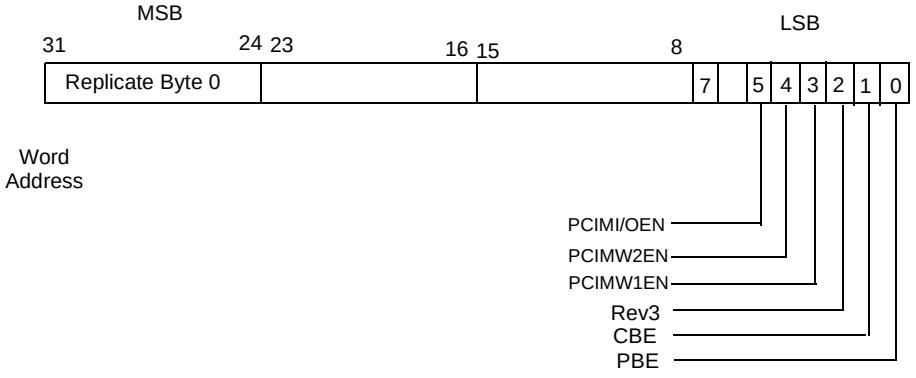
The native endian mode for MIPS processors, such as Motorola® and IBM® 370® processors, is Big-Endian mode. However, the native mode for Intel® (which developed the PCI standard) and VAX® processors is Little-Endian mode. For PCI-compatibility reasons, most PCI-peripheral chips, including the VRC4375 controller, operate natively in Little-Endian mode.

While the VRC4375 controller is natively little endian, it supports either Big- or Little-Endian mode in the CPU interface. The state of the MuxAD[12:11] signal at reset determines this endian mode. However, there are important considerations when using the controller in a mixed-endian design. The most important aspect of the endian issue is which byte lanes of the SysAD bus are activated for a particular address.

If Big-Endian mode is implemented for the CPU interface, the controller swaps bytes within words and halfwords that are coming in and going out on the SysAD bus. All of the controller's other interfaces operate in Little-Endian mode. The following implications are associated with this:

- MuxAD[11] is sampled upon power-on reset and the corresponding bit 0 (PBE) in the Endian mode (EM) register is set/reset. If tied high, bit 0 is set to 1. The CPU has read/write privileges.
- MuxAD[12] is sampled upon power-on reset and the corresponding bit 1 (CBE) in the EM register is set/reset. If tied high, bit 1 is set to 1. The CPU has read/write privileges.

Figure 12. Endian Mode (EM) Register



The Endian mode register contains the following bits.

Bit 0	PCI Big Endian	*PCI bus is big endian* 0 = Do not swap bytes 1 = Swap bytes
Bit 1	CPU Big Endian	*CPU bus is big endian* 0 = Do not swap bytes 1 = Swap bytes
Bit 2	Rev3	*VRC4375 revision* 0 = Revision 2 or earlier 1 = Revision 3
Bit 3	PCIMW1EN	*PCI access through Master Window 1* Byte steering will be enabled if this bit is set with the BE bit. See the truth table in Table 31.
Bit 4	PCIMW2EN	*PCI access through Master Window 2* Byte steering will be enabled if this bit is set with the BE bit. See the truth table in Table 31.
Bit 5	PCII/OEN	*PCI access through I/O window* Byte steering will be enabled if this bit is set with the BE bit. See the truth table in Table 31.

Note: At power-on reset, do not pull MuxAD[10] and MuxAD[11] high at the same time, as this may cause a malfunction.

The sections below view the endian issue from a programmer's perspective. They describe how to implement mixed-endian designs and how to make code endian independent.

- Data in memory is always ordered in Little-Endian mode, even with a big-endian CPU or PCI interface.
- Data in all internal registers and FIFOs is considered little endian, regardless of CPU or PCI endianness.

12.2
Endian Mode Operation

The controller provides steer-byte and swap-byte mechanisms from/to a big-endian CPU to/from internal registers and PCI registers/devices. Byte steering occurs during register access only if steering is enabled. Data swapping is always enabled for the CPU and system controller in different endian modes. Data steering has higher priority than data swapping.

Table 31 displays the truth table. Byte steering occurs as shown in Figure 13.

After power-on reset, byte swapping is controlled by the PBE and CBE signals, which are set if there is a pull-up resistor at the signal pins MuxAD[11] and MuxAD[12]. (See Figure 14.)

Figure 15 shows the bit and byte order of the two endian modes, as it applies to bytes within word-sized data items. The bit order within bytes is the same for both modes. The big (most-significant) bit is on the left side, and the little (least-significant) bit is on the right side. Only the bit order of subitems is reversed within a larger addressable data item (halfword, word, doubleword, or quadword) when crossing between the two endian modes. The subitem order of significance within the larger data item remains the same. For example, the least-significant halfword (LSHW) in a word is always to the right and the most-significant halfword (MSHW) is to the left.

When Endian mode register bit 1 is set, the data bytes are steered as shown in Figure 13.

Figure 13. Byte Steering Mechanism

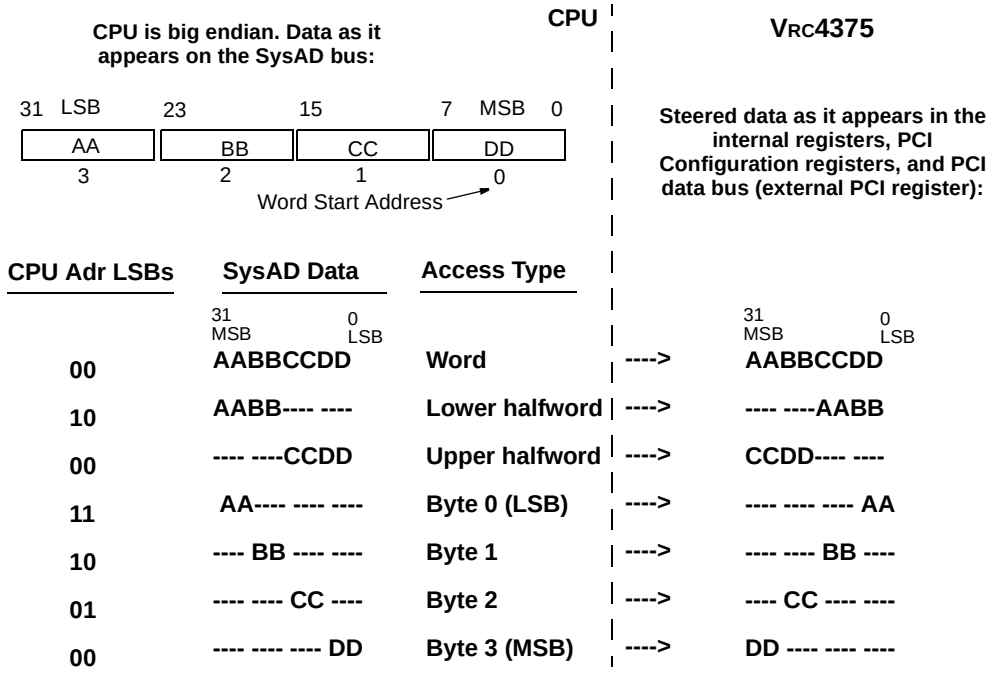


Table 31. Byte Steer/Swap Truth Table

EM Reg : REV3	Reg. Access[Note]	BE		Swap Byte or	Steer Byte
0	0	0	---->	0	0
0	0	1	---->	1	0
0	1	0	---->	0	0
0	1	1	---->	1	0
1	0	0	---->	0	0
1	0	1	---->	1	0
1	1	0	---->	0	0
1	1	1	---->	0	1

Note: Register access occurs when PCIMW1EN, PCIMW2EN, or PCII/OEN = 1.

When Endian mode register bit 1 is set, the data bytes are swapped as shown in Figure 14.

Figure 14. Byte Swap Mechanism

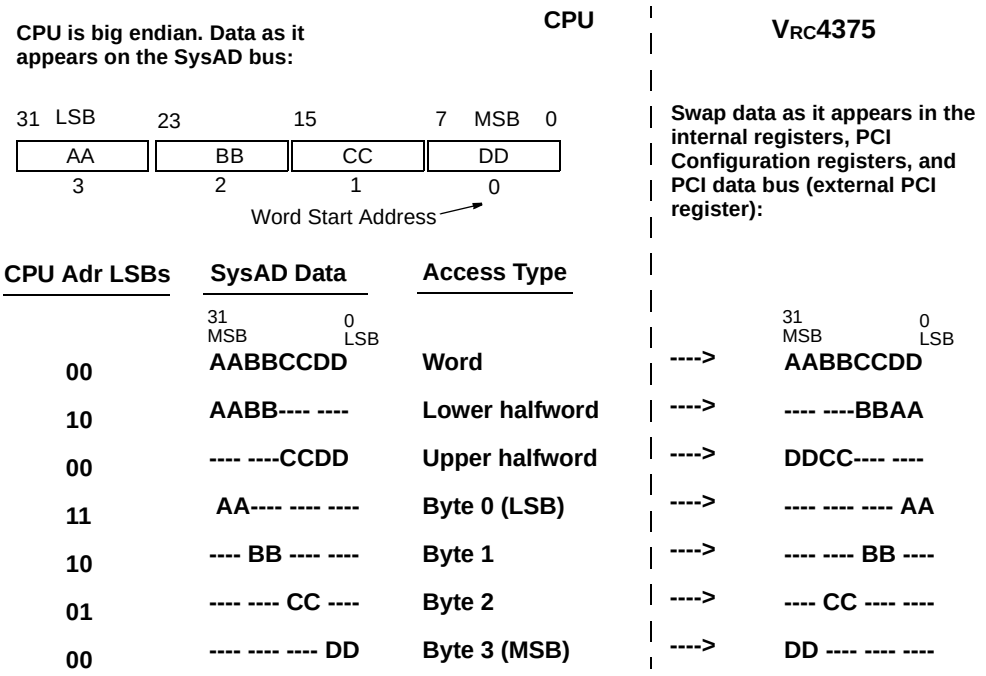
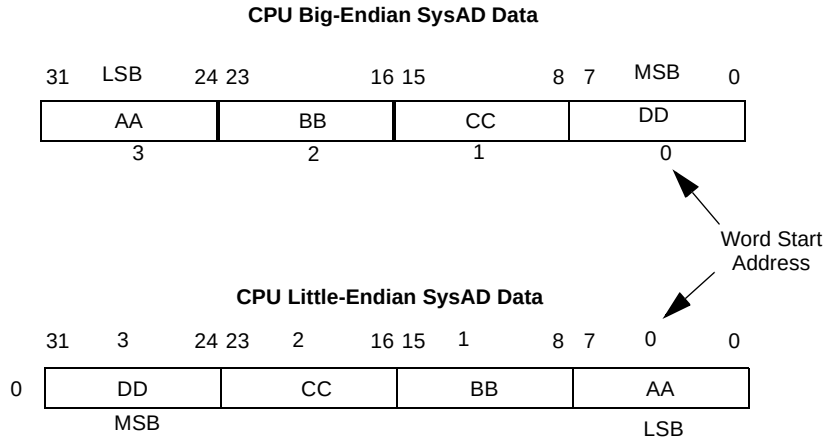


Figure 15. Bit and Byte Order of Endian Modes (Word Accesses)



MSB = Most-significant byte
LSB = Least-significant byte

If the access type matches the data item type, subitem data swapping is unnecessary. Thus, when making halfword accesses into a data array consisting of halfword data, no byte swapping takes place. In this case, data item bit order is retained between the two endian modes. The code that sequentially accesses the halfword data array would be identical, regardless of the endianness of its CPU. The code would be endian independent.

However, when making halfword accesses into a data array consisting of word data, access to the *more significant* halfword requires the address corresponding to the *less significant* halfword (and vice versa). Such code is not endian independent. A supergroup access (for example, accessing two halfwords simultaneously as a word from a halfword data array) causes the same problem. Such problems also arise when a halfword access is made into a 32-bit I/O register, whereas a word access into a 32-bit register creates no problem.

12.3 LAN Controller Example

The AMD® Am79C971® LAN controller is one example of a PCI bus device that is natively little endian but adapts to mixed-endian environments. This LAN controller supports big-endian system interfaces with two data types: a 32-bit word corresponding to the width of I/O registers, and an 8-bit byte corresponding to the width of the Ethernet DMA FIFO.

12.3.1 DMA Accesses From Ethernet FIFO

Ethernet data packets consist of bytes. To maximize bus bandwidth, these bytes are transferred using 32-bit word DMA accesses into memory. The mismatch in the mixed-endian environment means that a byte swap must be performed to allow the little-endian LAN controller to access the big-endian memory. The LAN controller provides its own internal hardware for this byte swap.

12.3.2 Word Accesses Into I/O Registers

The 32-bit internal I/O registers of the LAN controller are assumed to be accessed by 32-bit word transfers. In that case, the access type and data type match, and no swapping of bytes or halfwords is needed because the order of significance is the same for both endian modes. For such word transfers, the I/O register model is endian independent; internal swapping hardware for nonword accesses into the I/O registers is not provided.

Word accesses offer the advantage that the register address values documented in the *Am79C971 Technical Manual* can be used without change (although offsets for individual register fields such as the PCI latency timer must be ignored). The position of individual register fields, as well as byte position within these fields, would also remain as documented in the *Am79C971 Technical Manual*.

12.3.3 Byte or Halfword Accesses Into I/O Registers

Word accesses can cause some inconvenience (for example, shadow registers) when modifying only one or two fields within a 32-bit PCI register. In this case, byte or halfword access to the 32-bit register may be simpler. This type of transfer is analogous to the halfword access into a data array consisting of word data types shown in Figure 13 and Figure 14. Such accesses are mismatched to the defined data type and must be cross-addressed to get the byte or halfword of interest. The Am79C971 LAN controller does not provide big-endian hardware support to deal with byte or halfword transfers into the I/O registers. Code written to perform byte or halfword accesses into the 32-bit I/O registers will not be endian independent.

The I/O register field addresses documented in the *Am79C971 Technical Manual* are based on a register model derived from a little-endian perspective. The number order of these addresses progresses from right (least significant) to left. However, a big-endian system will respond to all addresses as if the number order progresses from left (most significant) to right. To access the desired byte or halfword, the address order documented in the *Am79C971 Technical Manual* must be reversed.

The PCI Status register and PCI Command register fields are examples of frequently used I/O register fields. The address offsets documented in the technical manual are 0x06 and 0x04, respectively. The PCI Command register field is located in the less significant halfword of the 32-bit I/O register that is also located at offset 0x04. The PCI Command register field shares the same offset with its 32-bit register because of the little-endian number order. In a big-endian system, the more significant halfword (PCI Status register field) would share the same offset value with its 32-bit register. So, if the offset 0x04 is used to access the PCI Command register field, a big-endian system would actually access the PCI Status register field. To access the proper halfword, the offsets must be exchanged between the two 16-bit register fields. In other words, there must be a reversal (or swapping) of number order, relative to the information documented in the *Am79C971 Technical Manual*.

These special addressing considerations are completely independent of the operand pointers associated with the CPU register used as the source or destination. The source or destination within the CPU's register file can be at any location, size, or alignment without altering the transfer results. A common error is to byte-swap CPU register data when transferring a halfword to or from a 32-bit register. The order of significance is the same for both endian modes, so no byte swapping is needed. This is purely an addressing problem.

Table 32 and Table 33 show how the offsets in the *Am79C971 Technical Manual* are swapped with other offsets to produce the proper cross-addressed offset required by big-endian systems. The determining factors for the swap are the values of the two least-significant bits of the offsets. According to the *Am79C971 Technical Manual*, the PCI Command register field has the offset 0x04. Table 33 shows that the offset 0x06 is needed to access the PCI Command register field. The two least-significant bits of 0x04 are b00, which convert to b10 to give the result of 0x06h.

Table 32. Cross Addressing for Byte Accesses Into a 32-Bit I/O Register

Least-Significant Bits of Offset from AM79C971 Technical Manual	Least-Significant Bits of Offset Required by Big-Endian System
b0 0	b1 1
b0 1	b1 0
b1 0	b0 1
b1 1	b0 0

Table 33. Cross Addressing for Halfword Accesses Into a 32-Bit I/O Register

Least-Significant Bits of Offset from AM79C971 Technical Manual	Least-Significant Bits of Offset Required by Big-Endian System
b0 0	b1 0
b1 0	b0 0

12.4 GUI Controller Example

The Cirrus Logic® CL-GD5465® GUI controller is another example of a PCI bus device that offers some mixed-endian support. The designers of this GUI controller assumed three data types: 32-bit word, 16-bit halfword, and 8-bit byte. Unlike the LAN controller, which could make certain assumptions as to data type (for I/O register or DMA FIFO accesses), the GUI hardware cannot determine what data type will be used during any particular data transfer; any data type might be involved in any I/O register or Rambus® DRAM (RDRAM®) access.

The data type must be known for a given bus transfer so that the appropriate byte or halfword swap can be performed. The data types may change from one bus cycle to the next; one software task may be operating in parallel with and independently of another software task. One of the easiest methods by which to accommodate such an environment, without semaphores and such, is to provide address apertures into the memory space.

The aperture scheme calls for GUI hardware resources to be mirrored into three address ranges. Depending on which address range is selected, a specific data type and data swap is used. Chapter 13 of the *CL-GD5465 Technical Reference Manual* gives details of these three apertures.

12.4.1 Word Accesses Into I/O Registers

The GUI controller's internal 32-bit I/O registers can be accessed with 32-bit word transfers. In this case, the access and data types match; no swapping of bytes or halfwords is required because the order of significance is the same for both endian modes. With such word transfers, the I/O register model is endian independent, so the first address aperture described in the *CL-GD5465 Technical Reference Manual* is used.

Word accesses have the advantage that the register address values documented in the technical manual can be used without change (although offsets for individual register fields such as the PC latency timer must be ignored). Individual register field positions and byte positions within these fields also remain the same, as shown in the *CL-GD5465 Technical Reference Manual*.

12.4.2 Accessing Byte or Halfword I/O Registers

As in the LAN controller example, byte or halfword access may be simpler than word accesses when modifying only one or two fields within a 32-bit I/O register. This type of transfer is analogous to the halfword access into a data array consisting of word data types, as shown in Figure 13 and Figure 14. Such accesses are mismatched to the defined data type and must be swapped to get the byte or halfword of interest. Code written to perform byte or halfword accesses into the 32-bit word I/O registers will not be endian independent.

There are two methods for performing byte or halfword accesses into the GUI controller. The first method uses the apertures for halfword-swap (second aperture) and byte-swap (third aperture) operations. This method has the advantage that the little-endian addresses documented in the *CL-GD5465 Technical Reference Manual* are the same as those used by big-endian code, except for the addition of the offset required to select the appropriate aperture. (As of this printing, the second aperture remains unverified.)

The second method of performing byte or halfword accesses is to cross-address the transfer. Care must be taken, however, when referencing the *CL-GD5465 Technical Reference Manual*. The I/O register field addresses documented in the technical manual are based on a little-endian register model. The address number order progresses from right (least significant) to left. However, big-endian systems respond to addresses as if the number order progresses from left (most-significant) to right. To access the desired byte or halfword, the address order documented in the *CL-GD5465 Technical Reference Manual* must be reversed.

12.4.3 Accessing RDRAM

The CL-GD5465 GUI controller's internal pixel and video engines constrain RDRAM to little-endian state. Here again, big-endian systems have a few problems accessing data subgroups, such as a single-byte access into a 32-bit data type. Subitem accesses are also a factor for RDRAM. Cross-addressing and address aperture solutions are the same as those described in Section 12.4.2. Supergroup accesses are also encountered with RDRAM. This situation is mentioned in Section 12.2. A specific GUI-oriented example of this would be an 8-bit data type, such as a pixel, which is transferred four at a time to maximize PCI bus bandwidth.

There are two methods for dealing with supergroup transfers. One is the address aperture method, used in the subitem scenario of Section 12.4.2. The third aperture, byte swap, is used to provide the proper data swap for the four 8-bit pixel case. The second aperture, halfword swap, is used to transfer such things as two 16-bit pixels simultaneously.

The second aperture method requires that the data order in the CPU register be swapped prior to an RDRAM write access, or immediately after an RDRAM read access. To continue with the previous four-pixel transfer example, the byte number order of the four pixels in the CPU register would be reversed. Now the pixel number order increases, starting from the right side of the register (the first pixel originally on the left is now on the right). Then the four pixels are written into the RDRAM with a standard 32-bit word transfer (first aperture). The case of two 16-bit pixels requires the two halfwords to be swapped, but not the order of the two bytes inside the halfwords. This second method is probably more time-consuming and is not recommended.

13.0 Timing Diagrams

This section provides the timing diagrams for VRC4375 system controller operations, as follows.

13.1

Memory Timing

Figure 16. Flash ROM Write

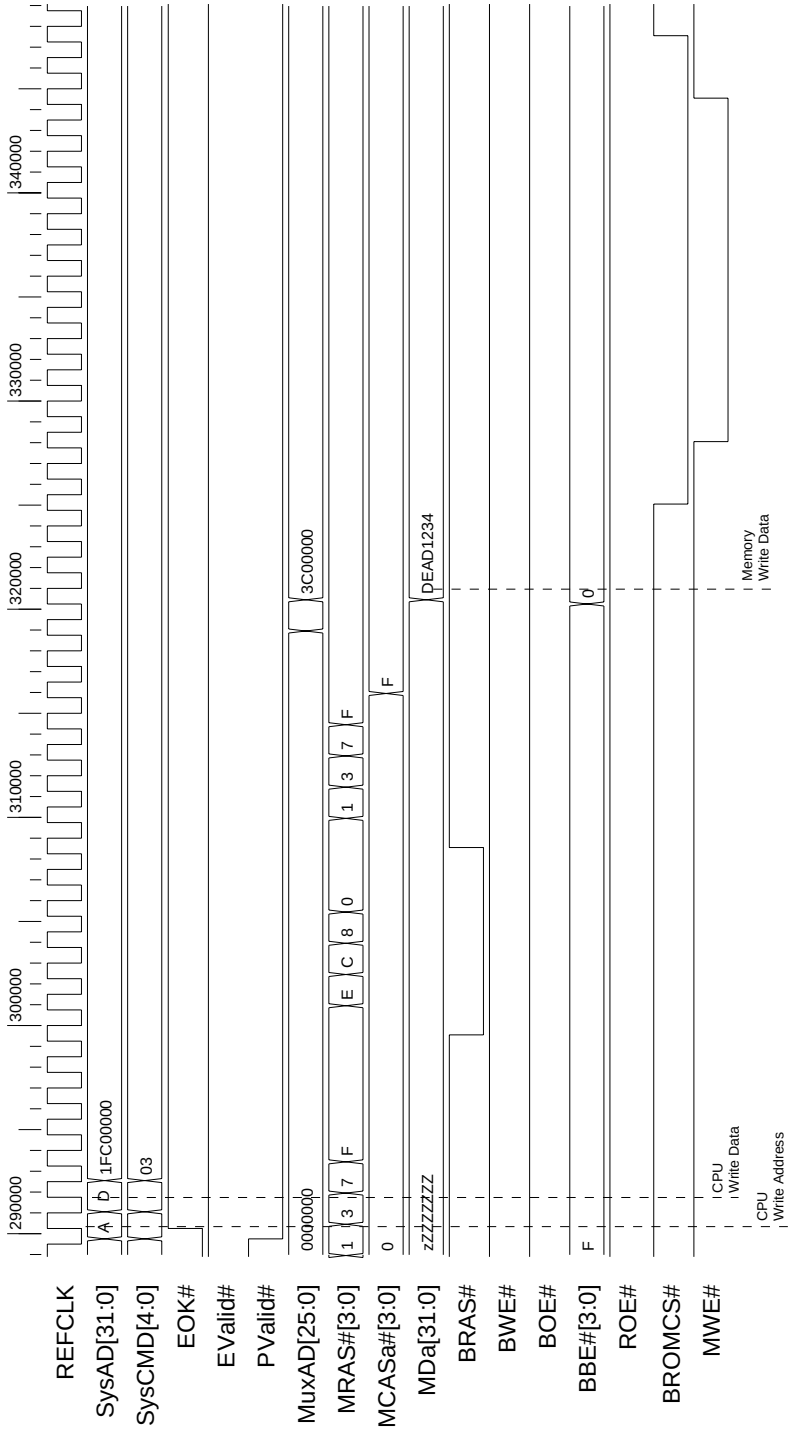


Figure 17. CPU Word Read from SDRAM Base Memory

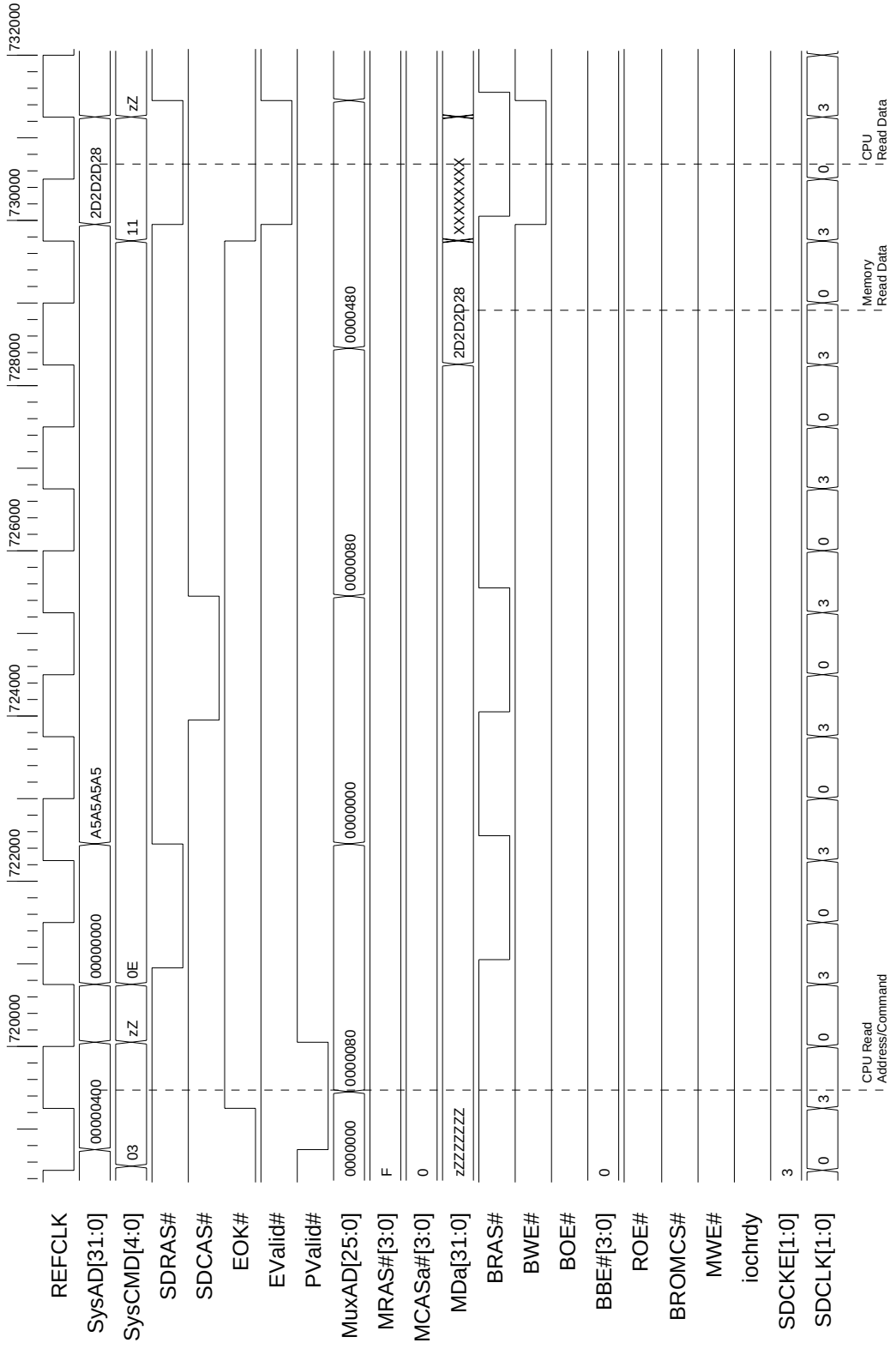


Figure 18. CPU Word Write to SDRAM Base Memory

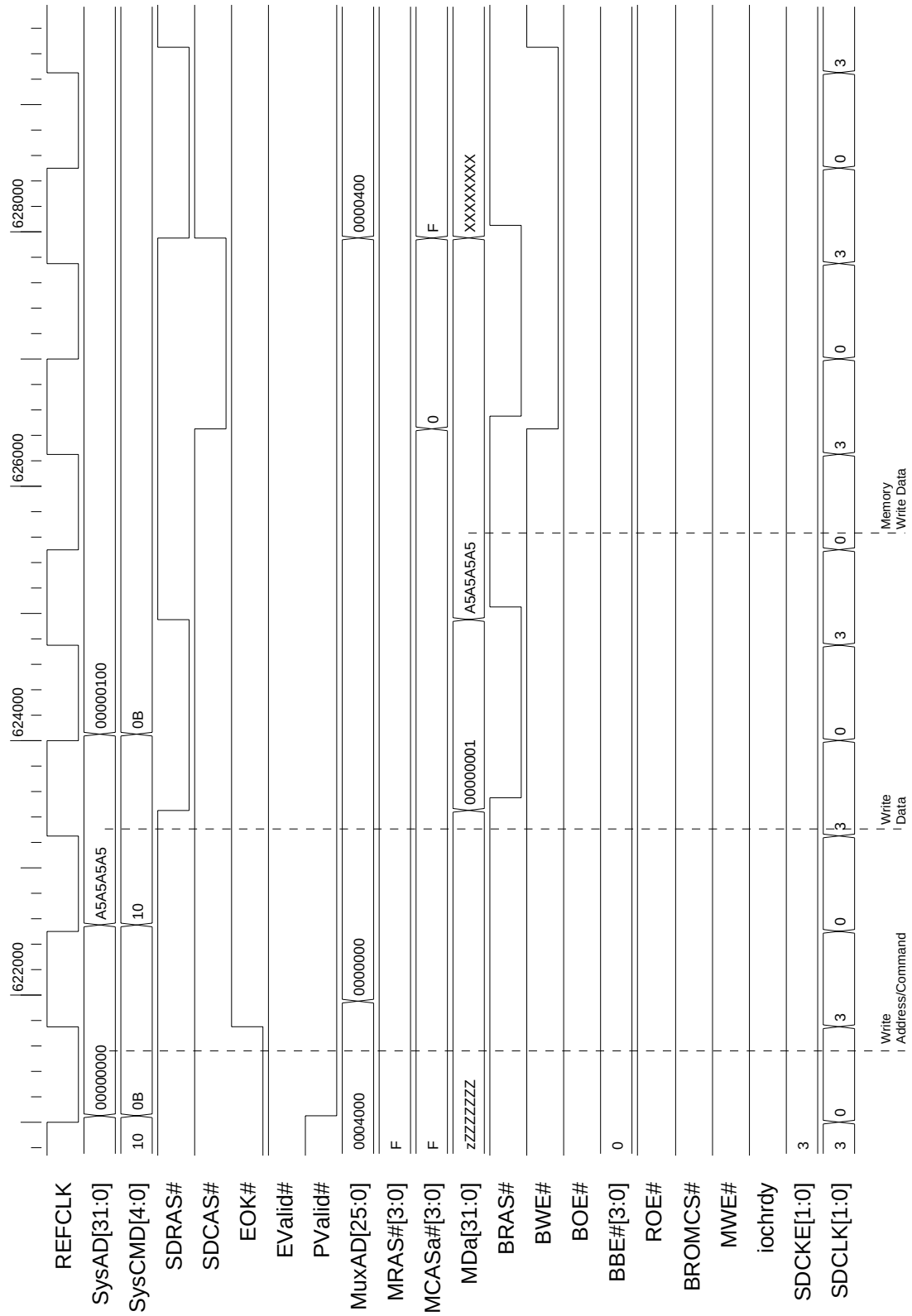


Figure 19. CPU Quad Read from SDRAM Base Memory

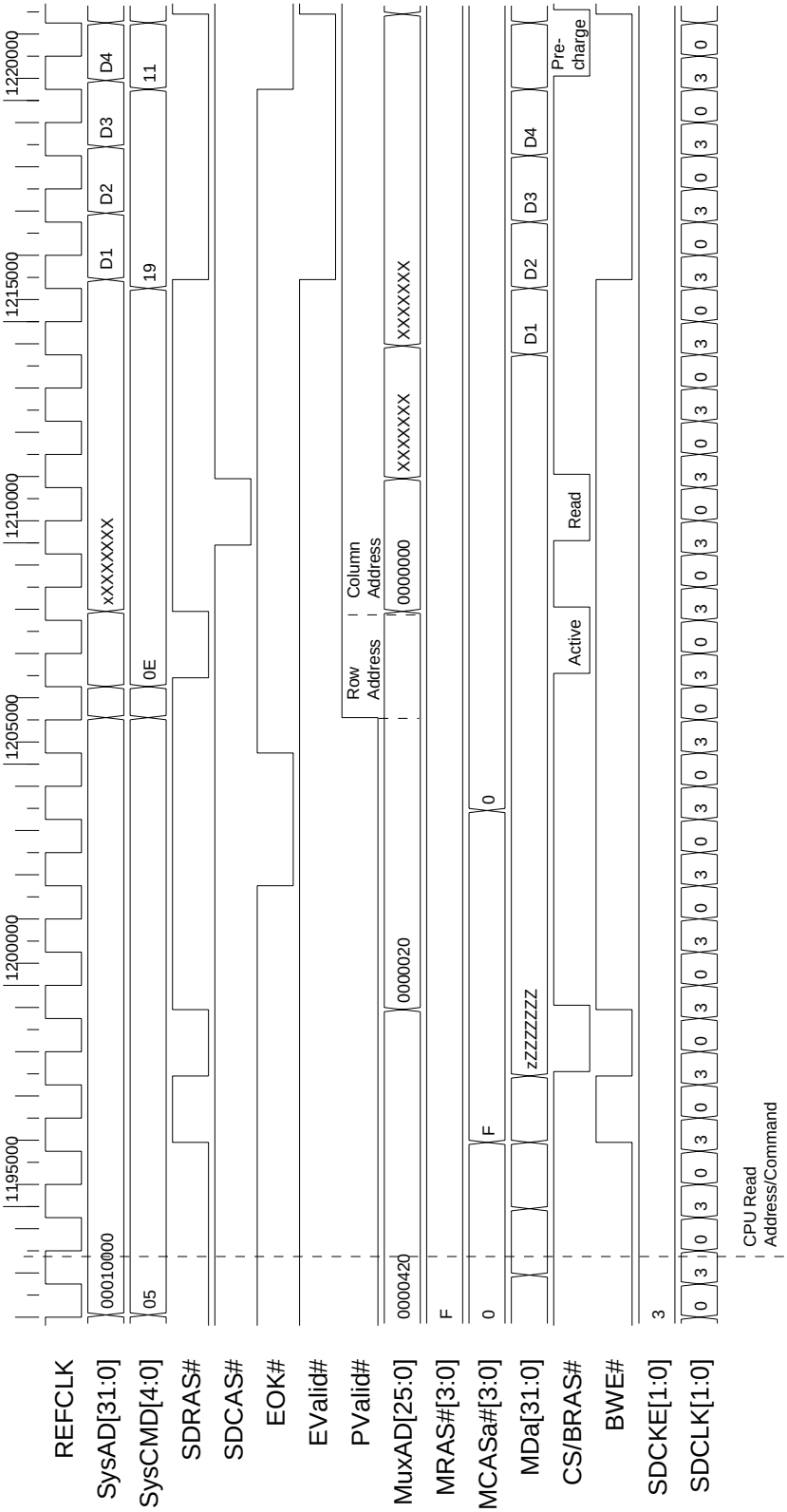


Figure 20. CPU Quad Write to SDRAM Base Memory

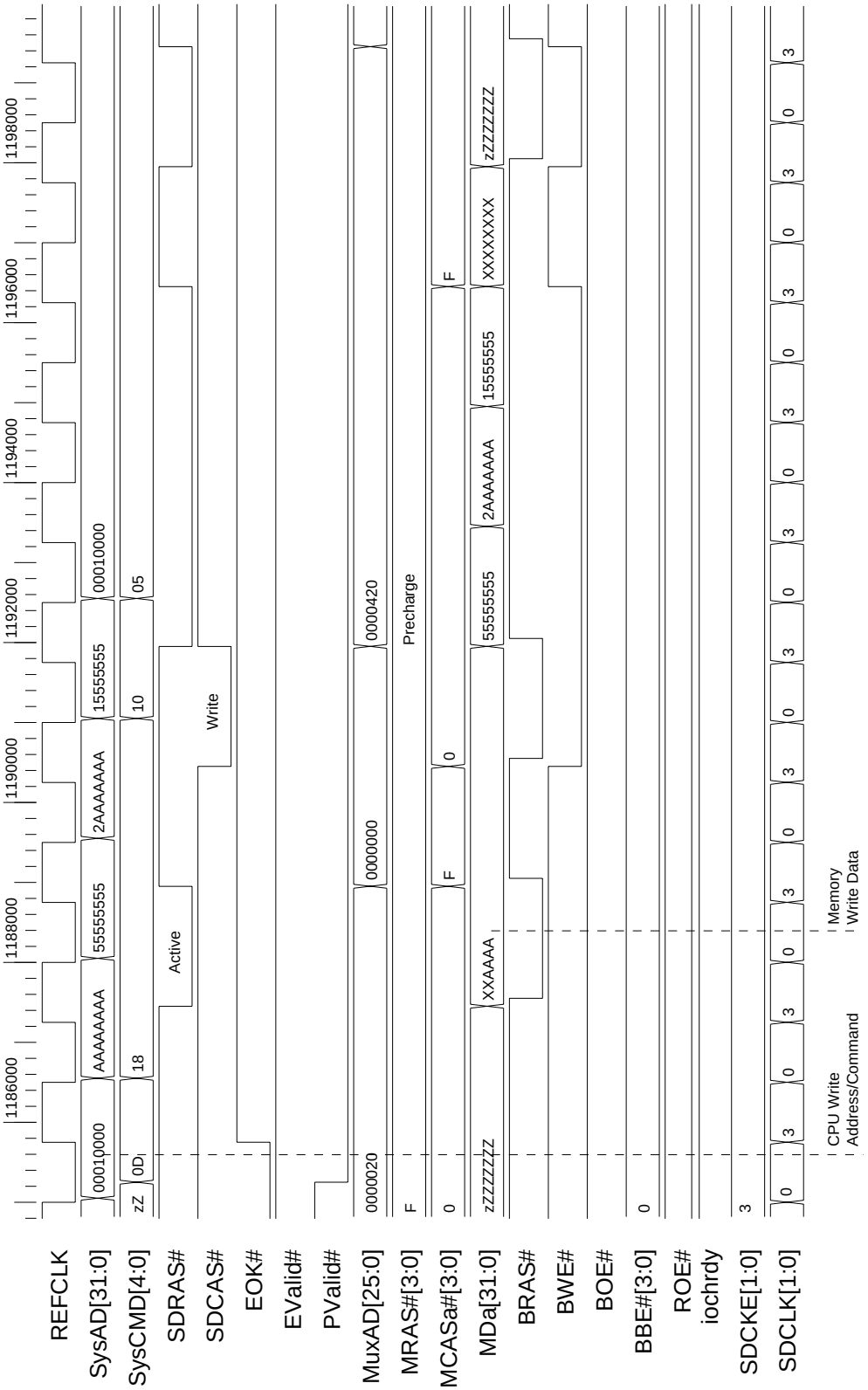


Figure 21. CPU Octet Read from SDRAM Base Memory

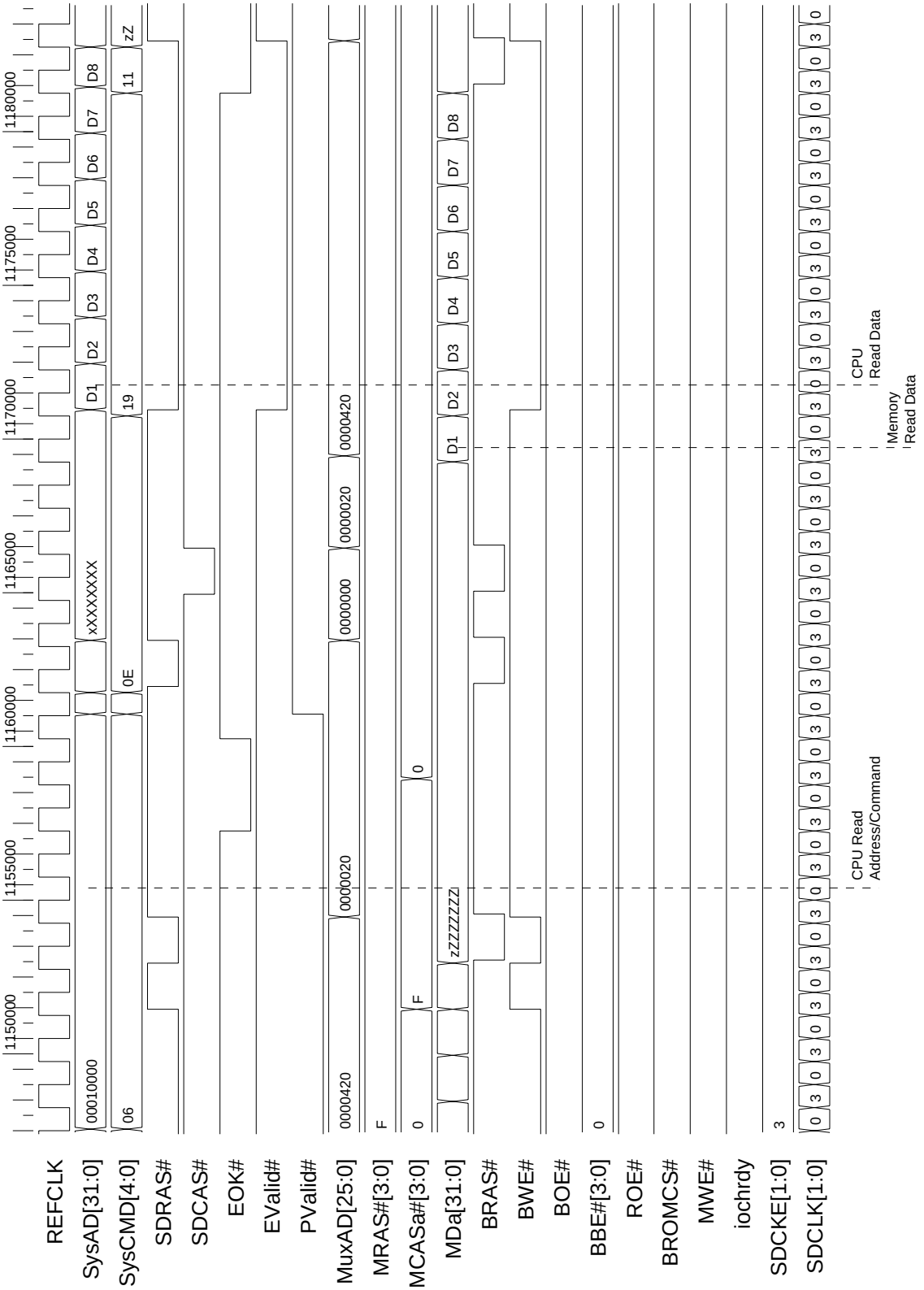


Figure 22. CPU Octet Write to SDRAM Base Memory

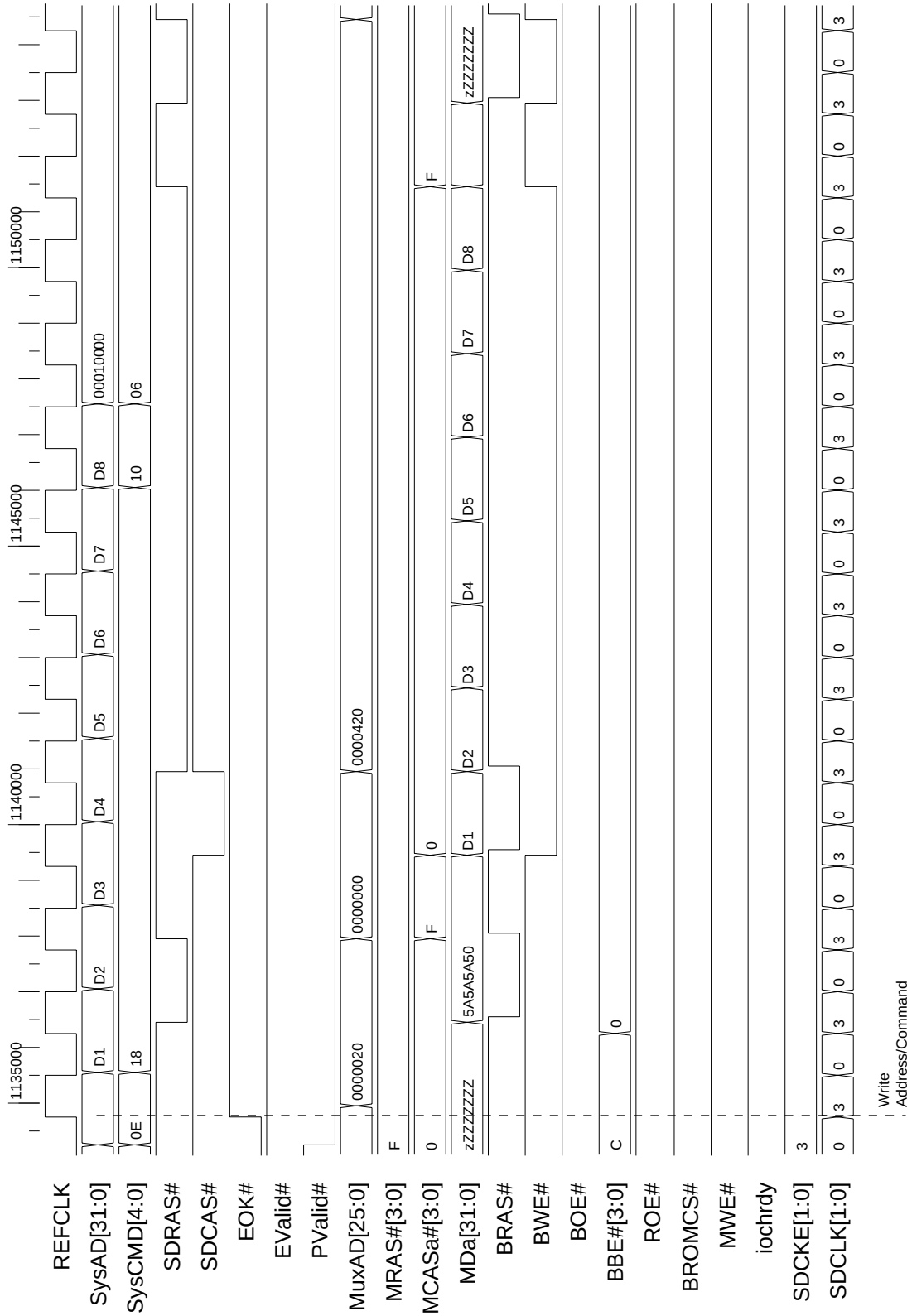


Figure 23. CPU Quad Read from SDRAM SIMM

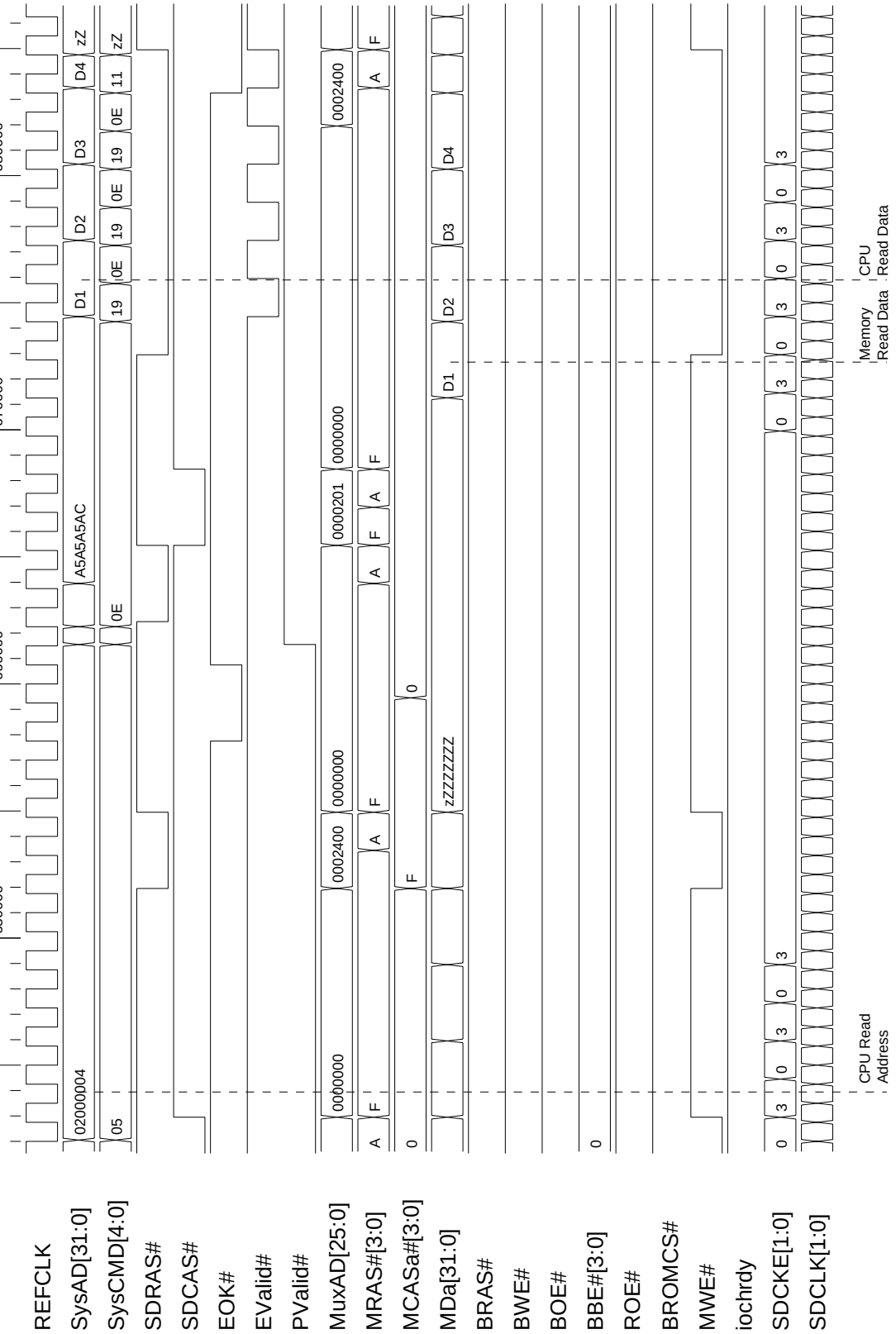


Figure 24. CPU Quad Write to SDRAM SIMM

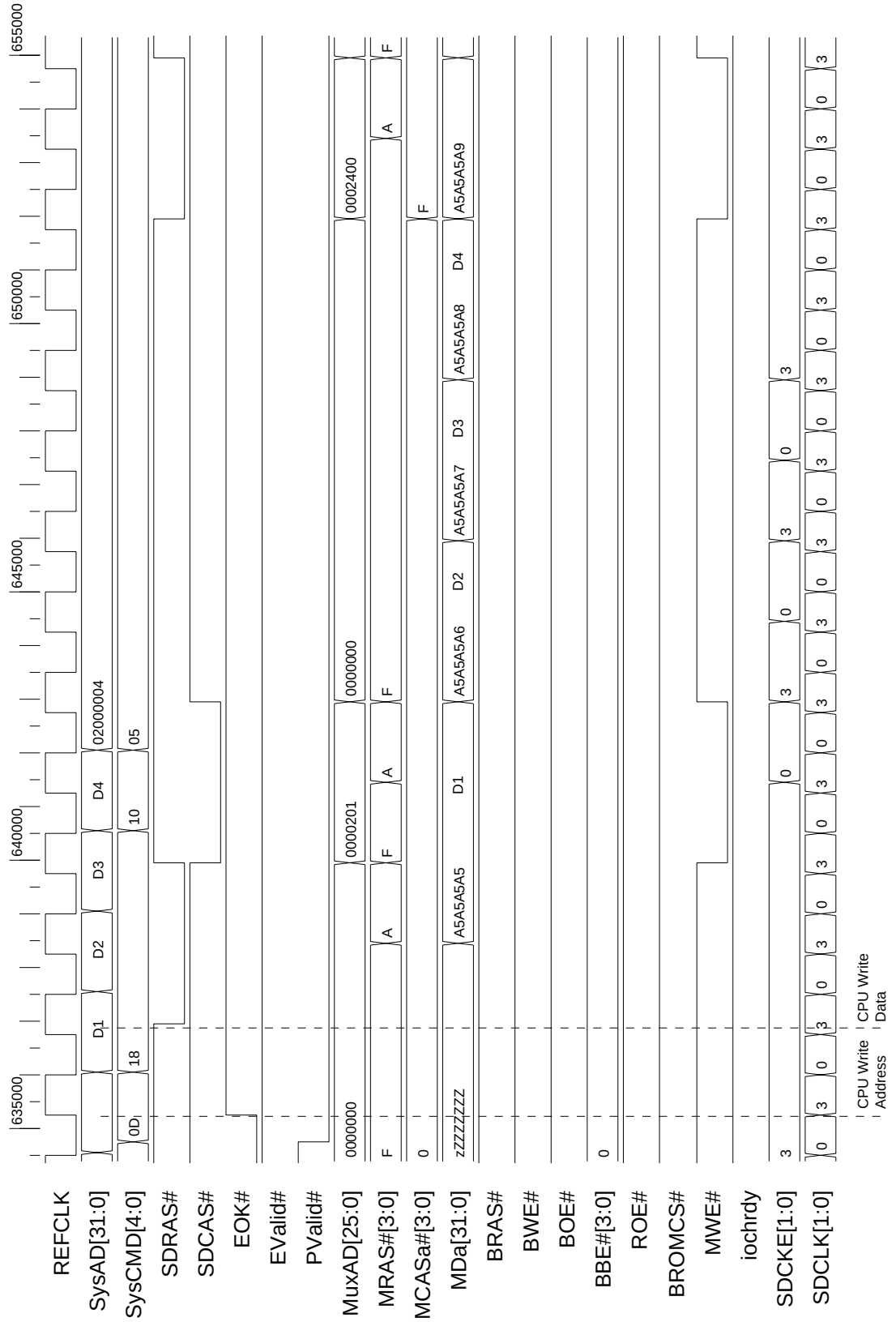


Figure 25. CPU Octet Read from SDRAM SIMM

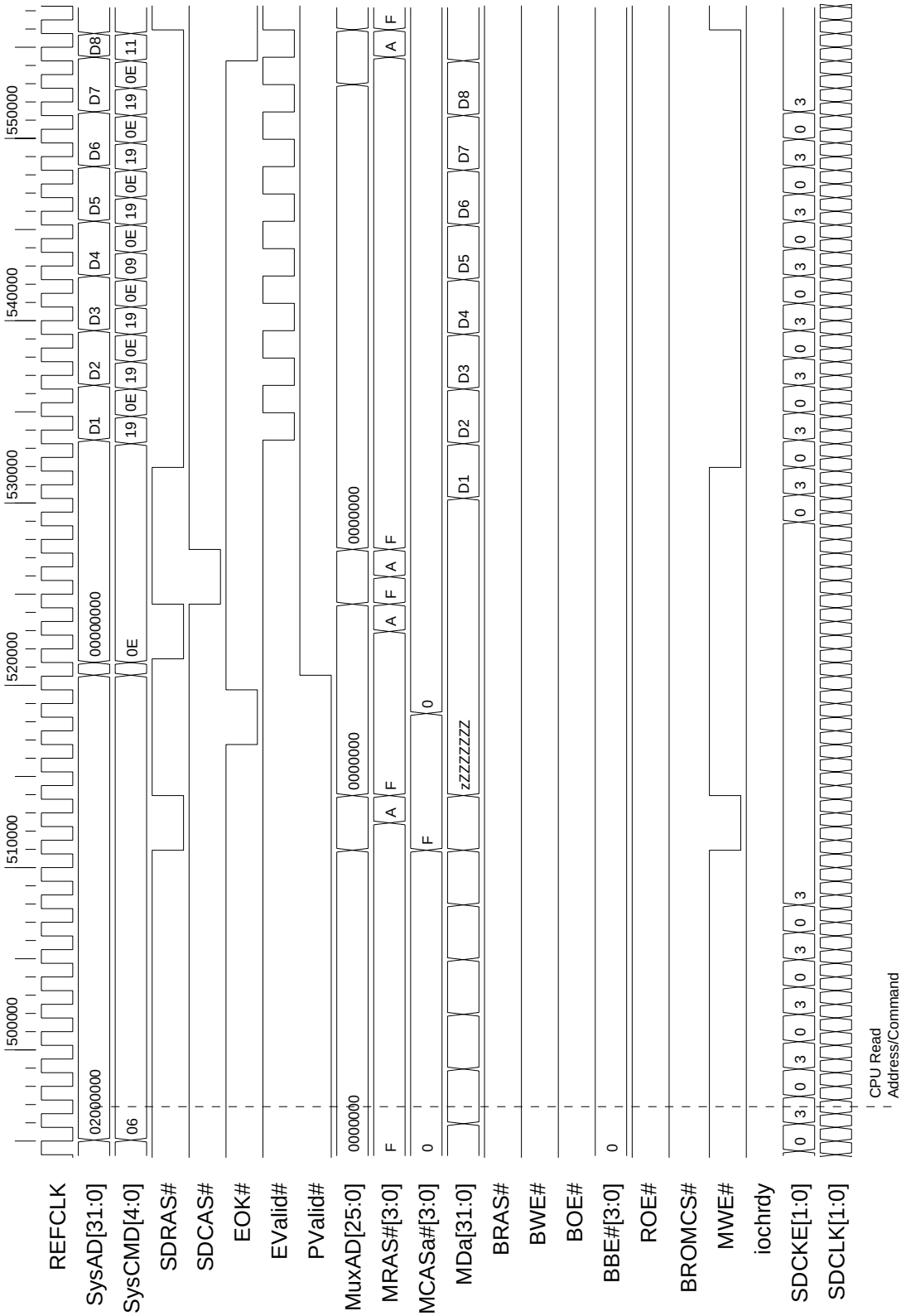


Figure 26. CPU Octet Write to SDRAM SIMM

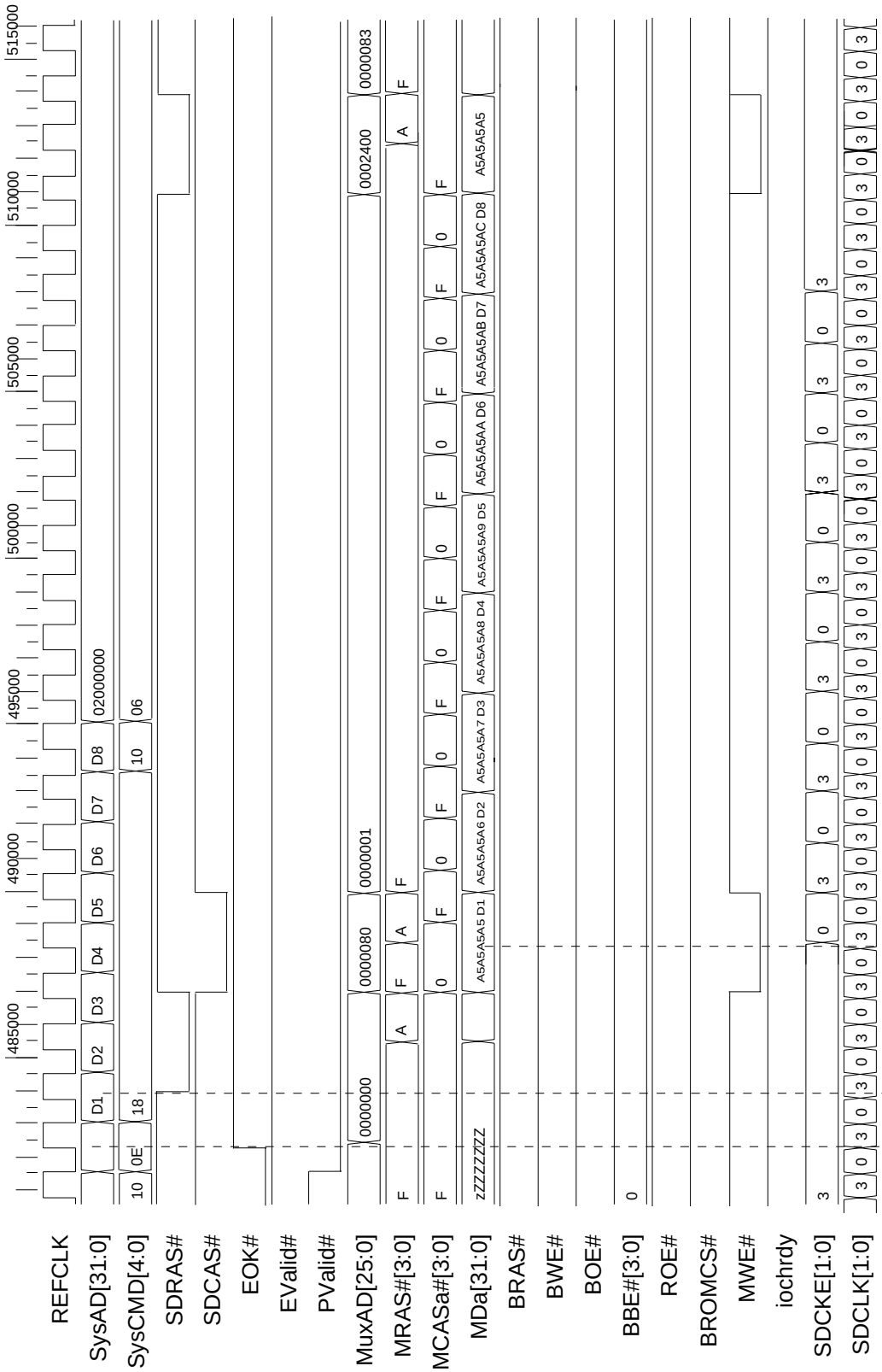


Figure 27. CPU Word Write to SDRAM SIMM

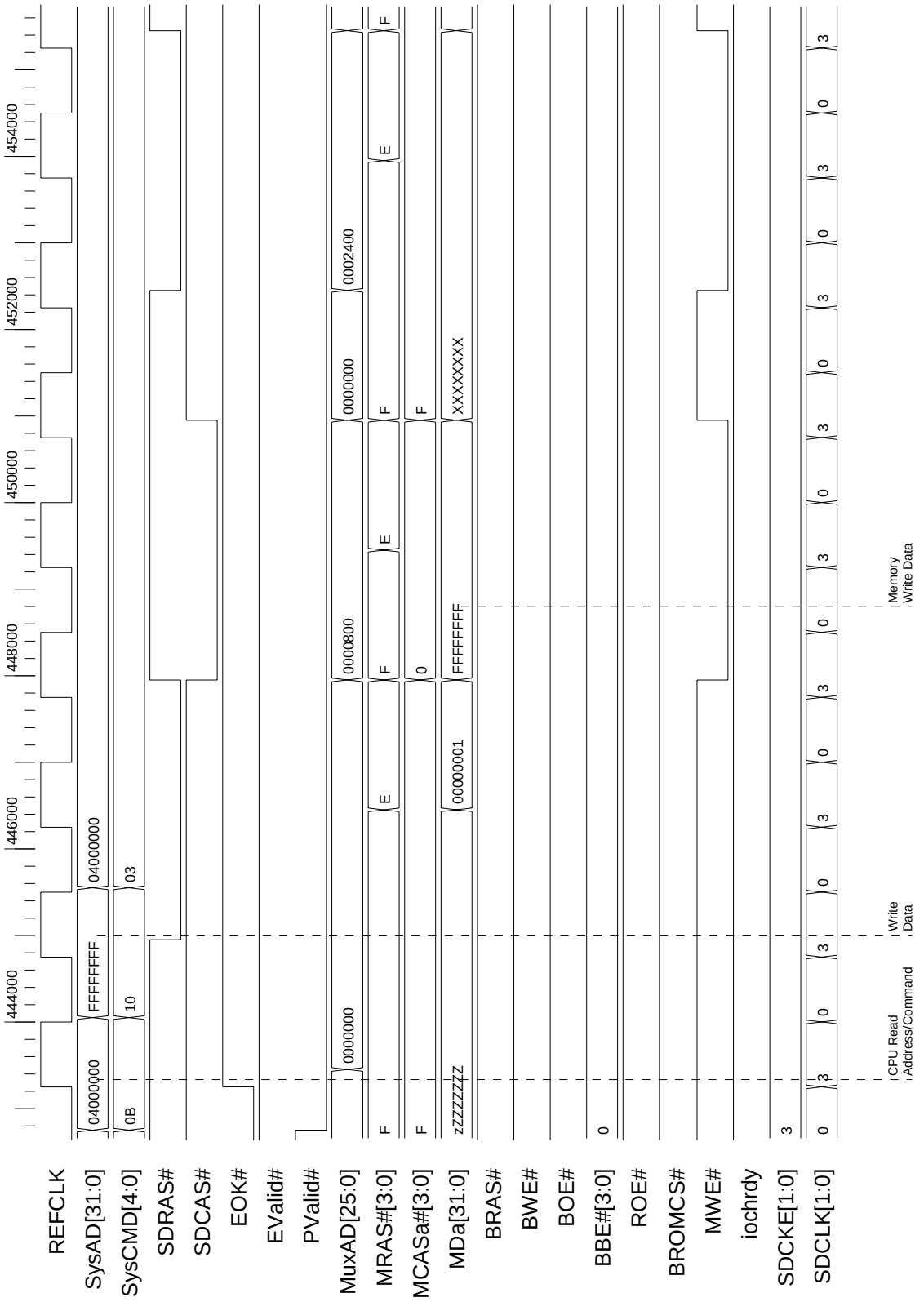


Figure 28. CPU Word Read from SDRAM SIMM

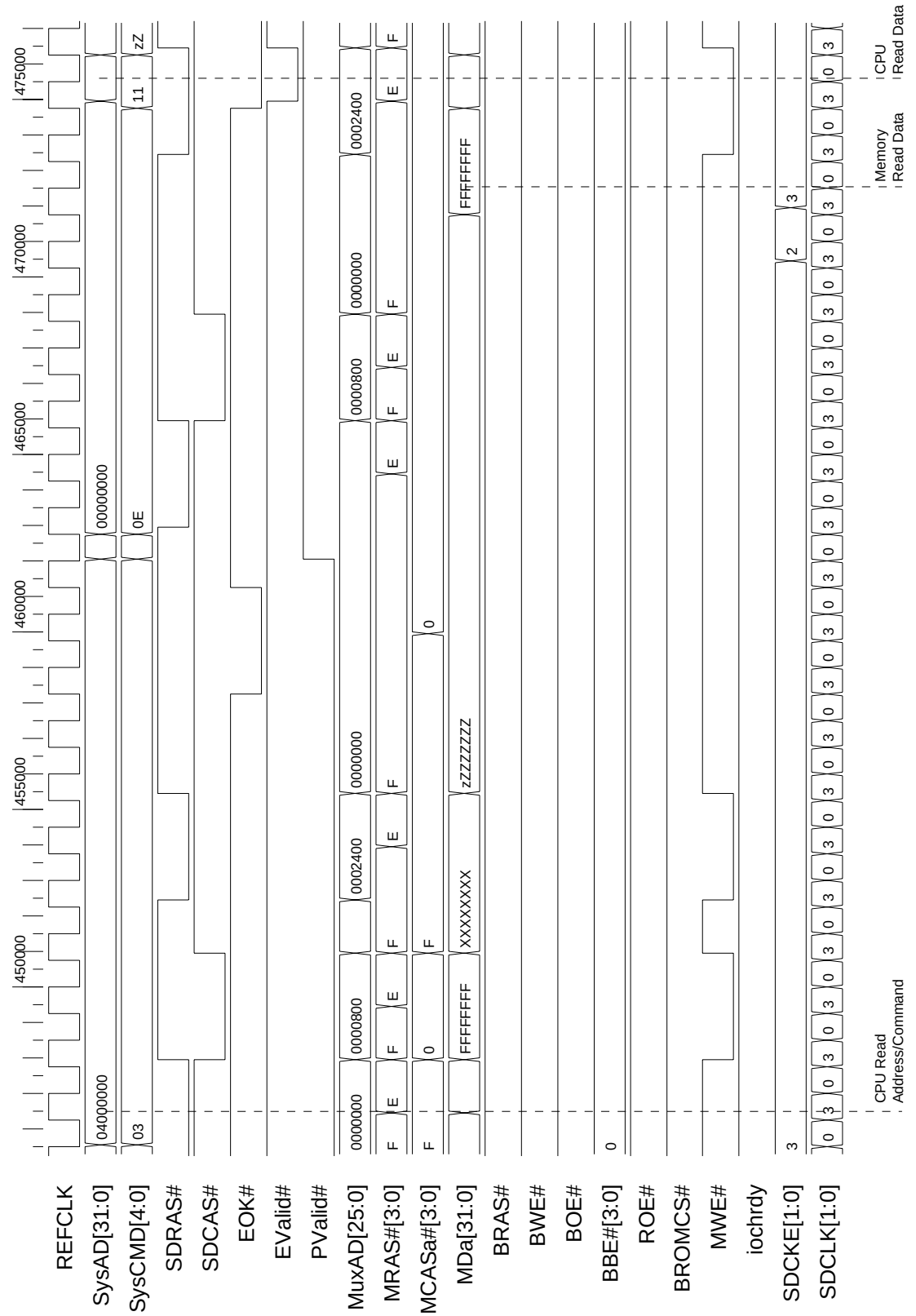
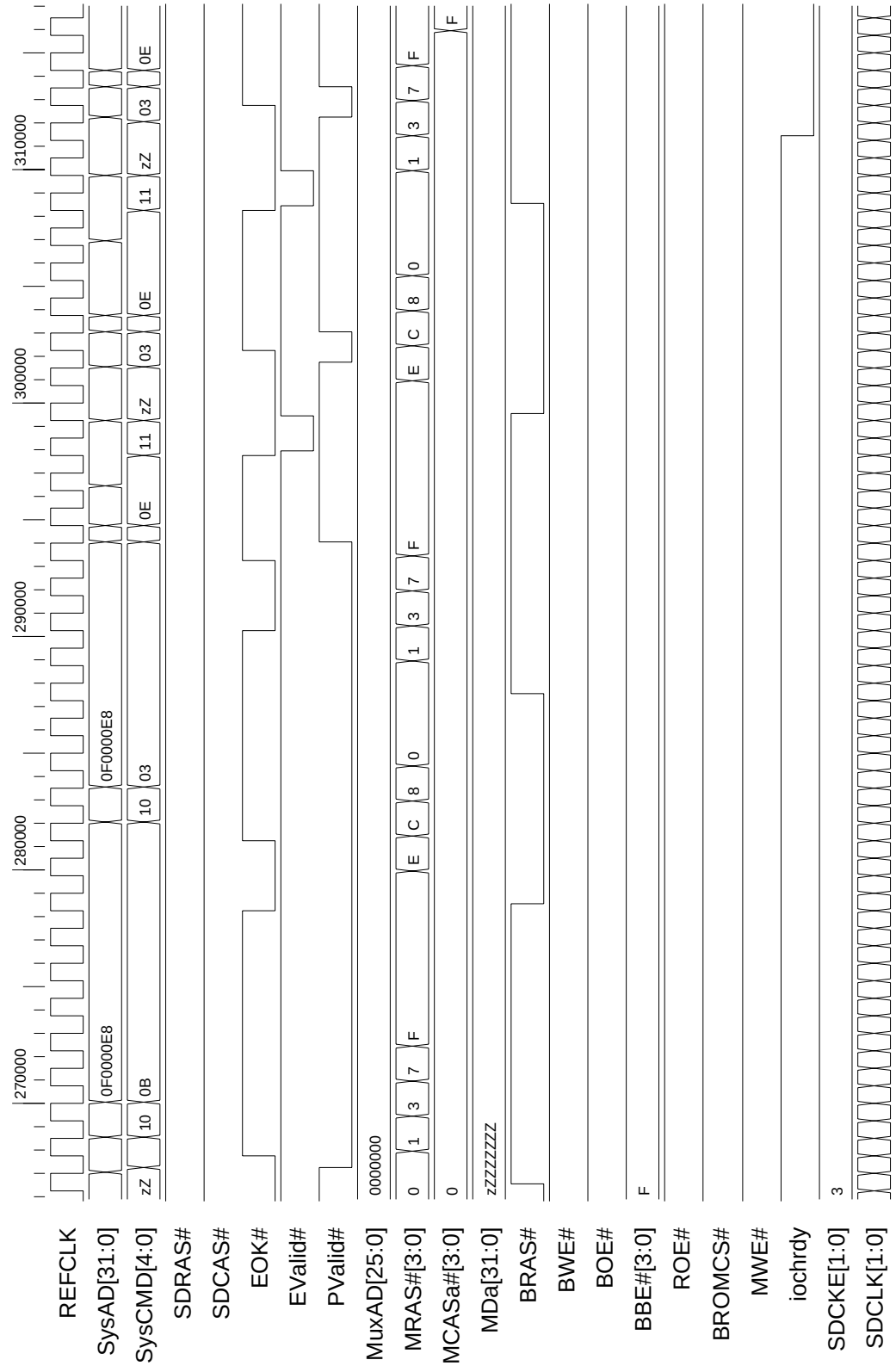


Figure 29. iochrdy Signal Timing



General-Purpose I/O Timing Control Register 00000C00 (Hex)

Figure 30. CPU Word Read from EDO Base Memory

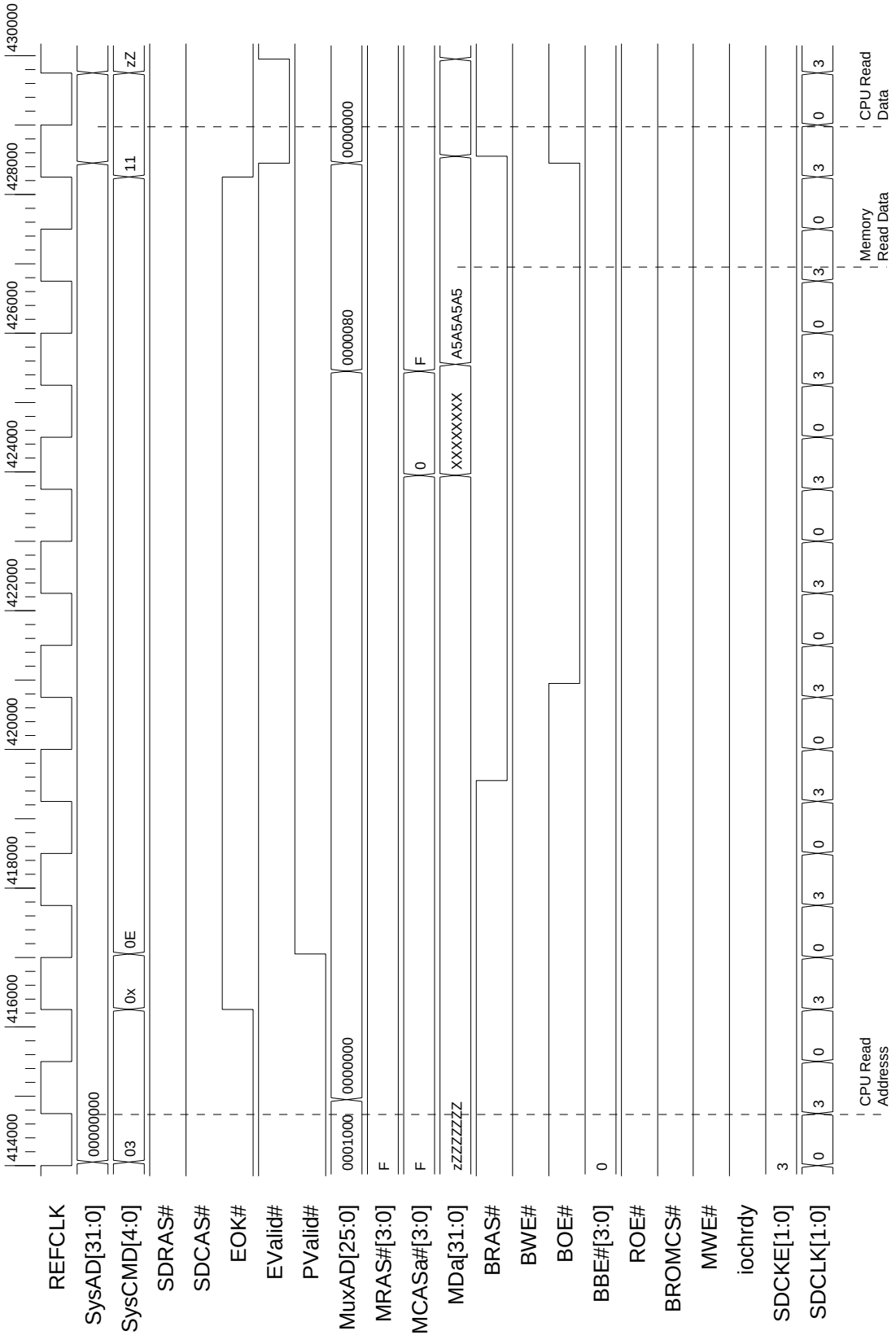


Figure 31. CPU Word Write to EDO Base Memory

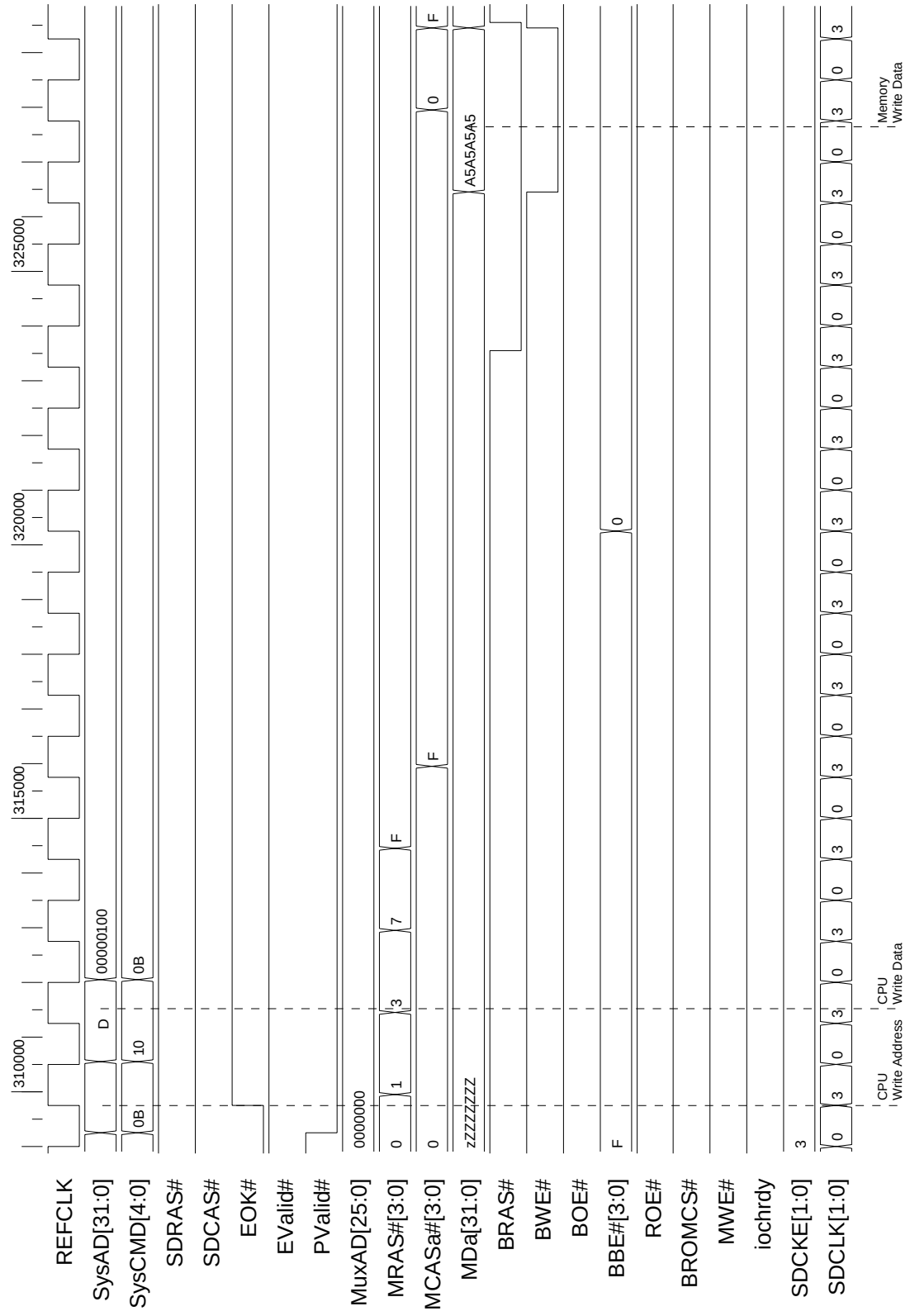


Figure 32. CPU Quad Read from EDO Base Memory

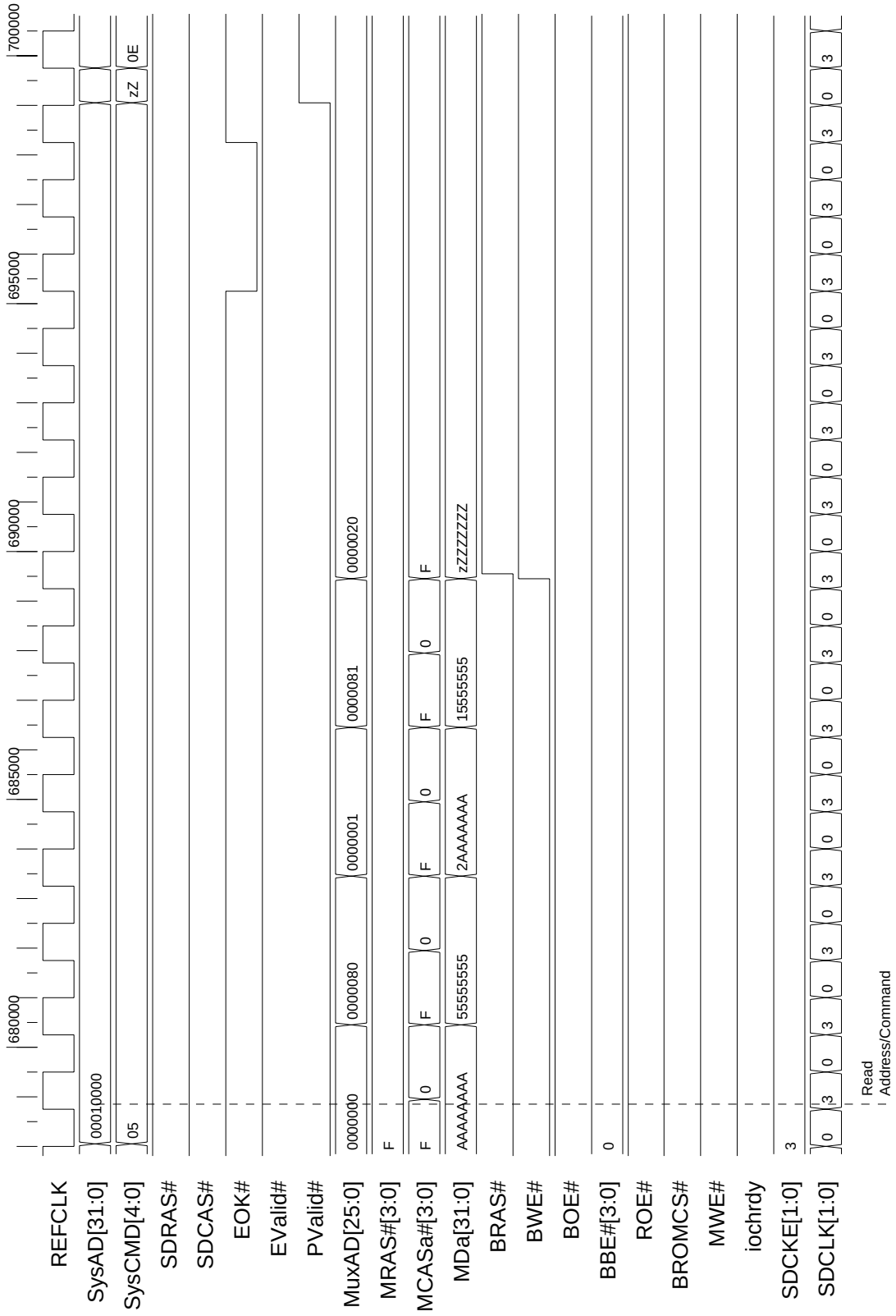


Figure 33. CPU Quad Write to EDO Base Memory

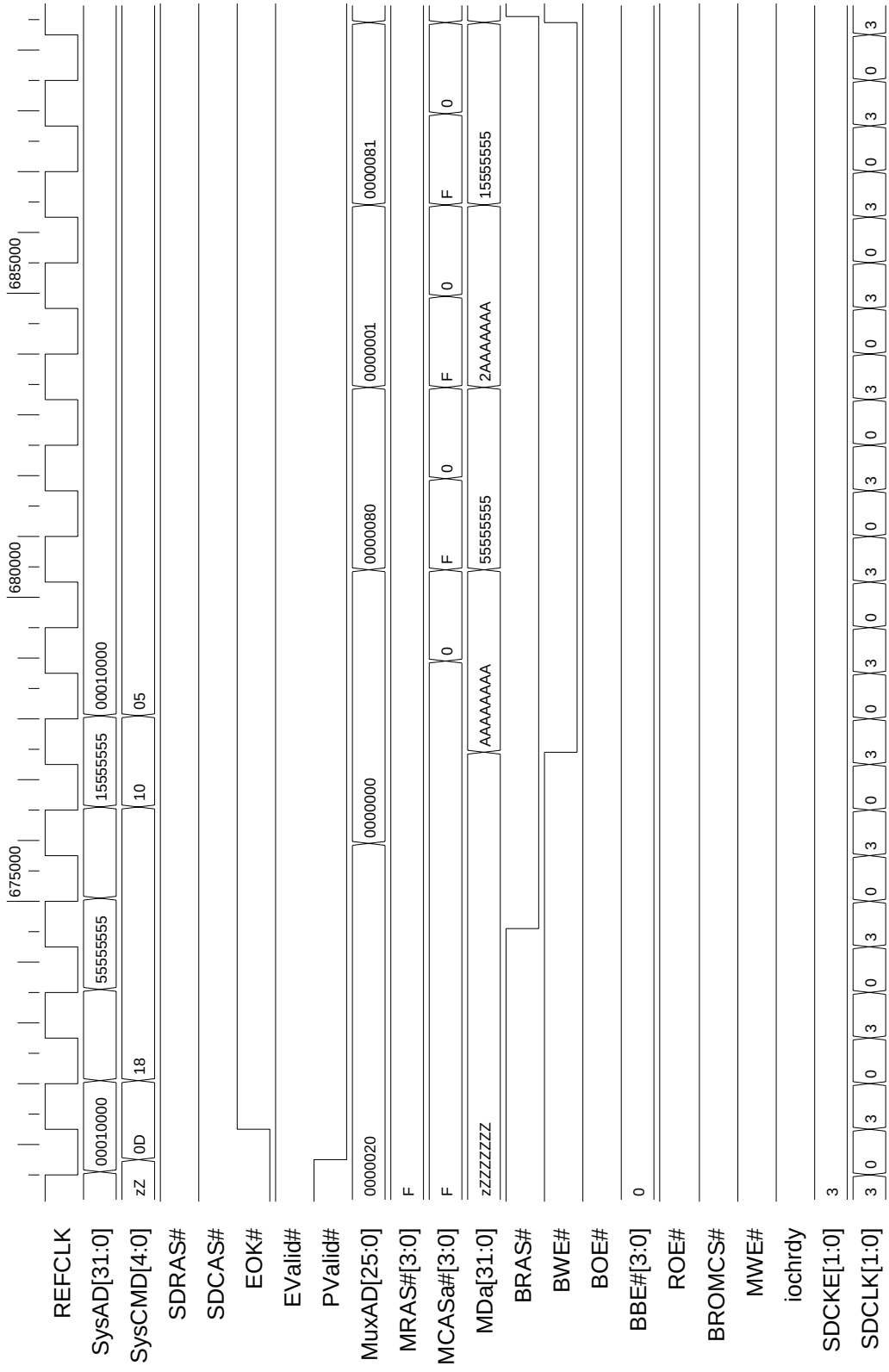


Figure 34. CPU Octet Write to EDO Base Memory

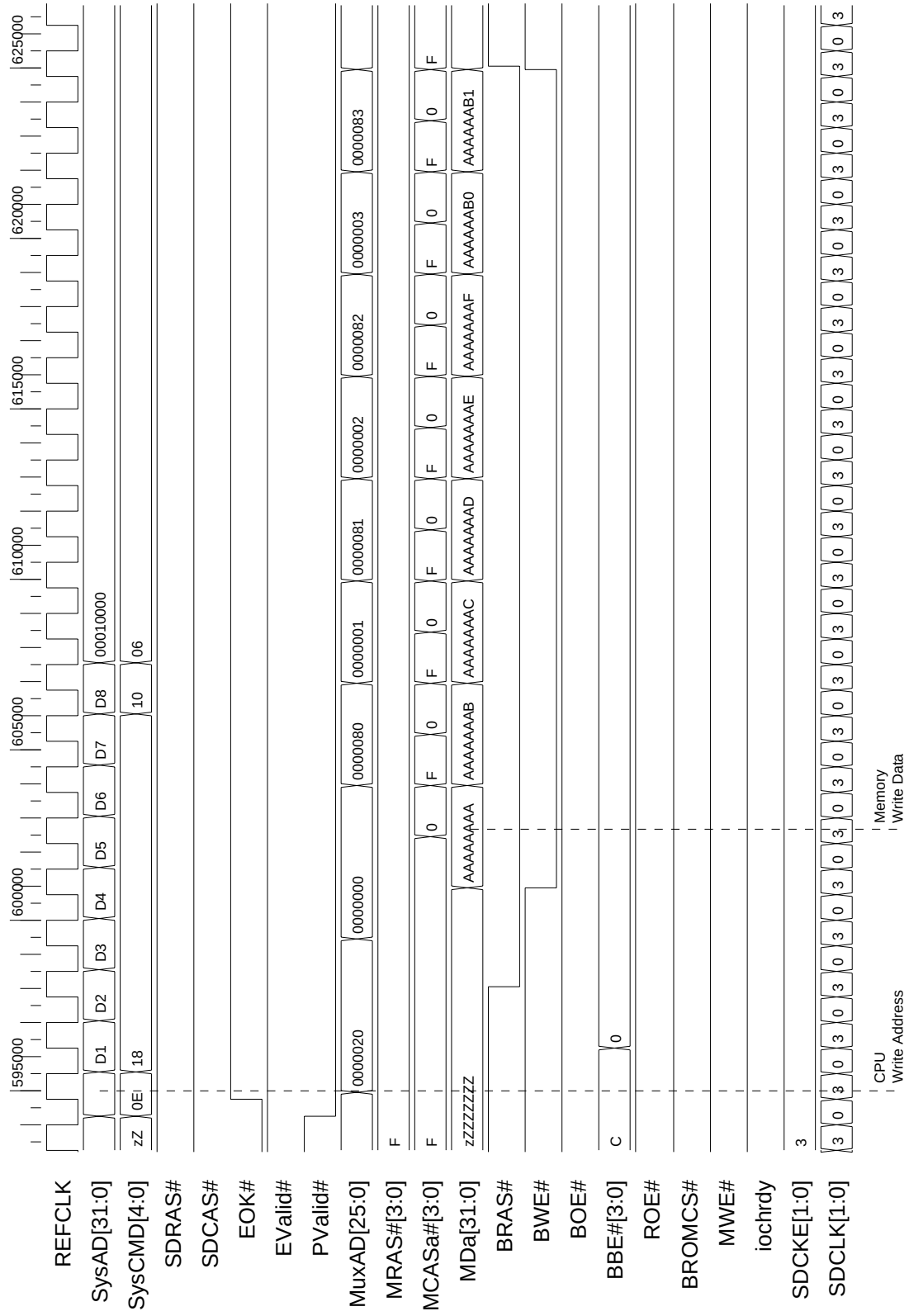


Figure 35. CPU Octet Read from EDO Base Memory (1 of 3)

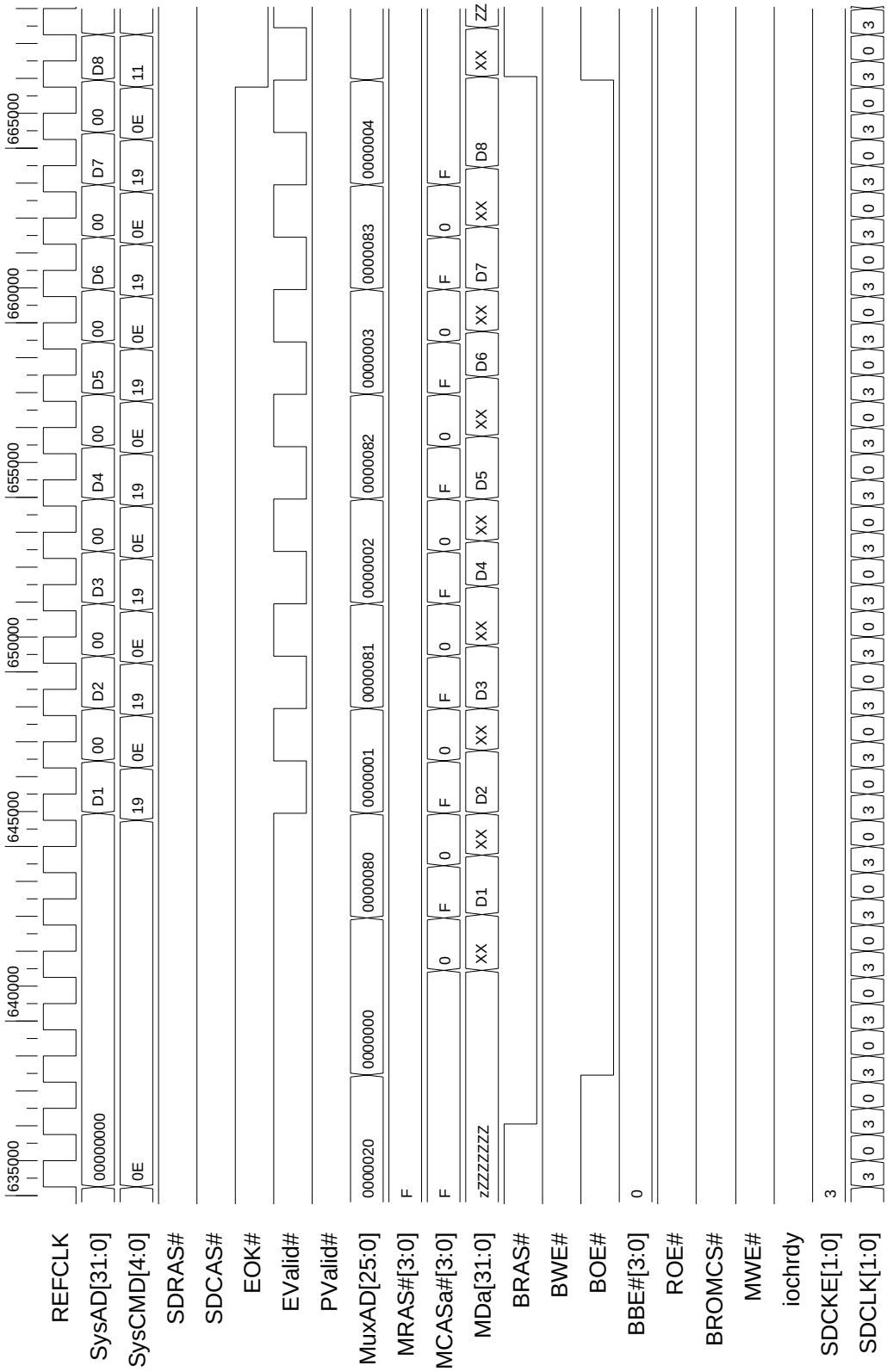


Figure 36. CPU Octet Read from EDO Base Memory (2 of 3)

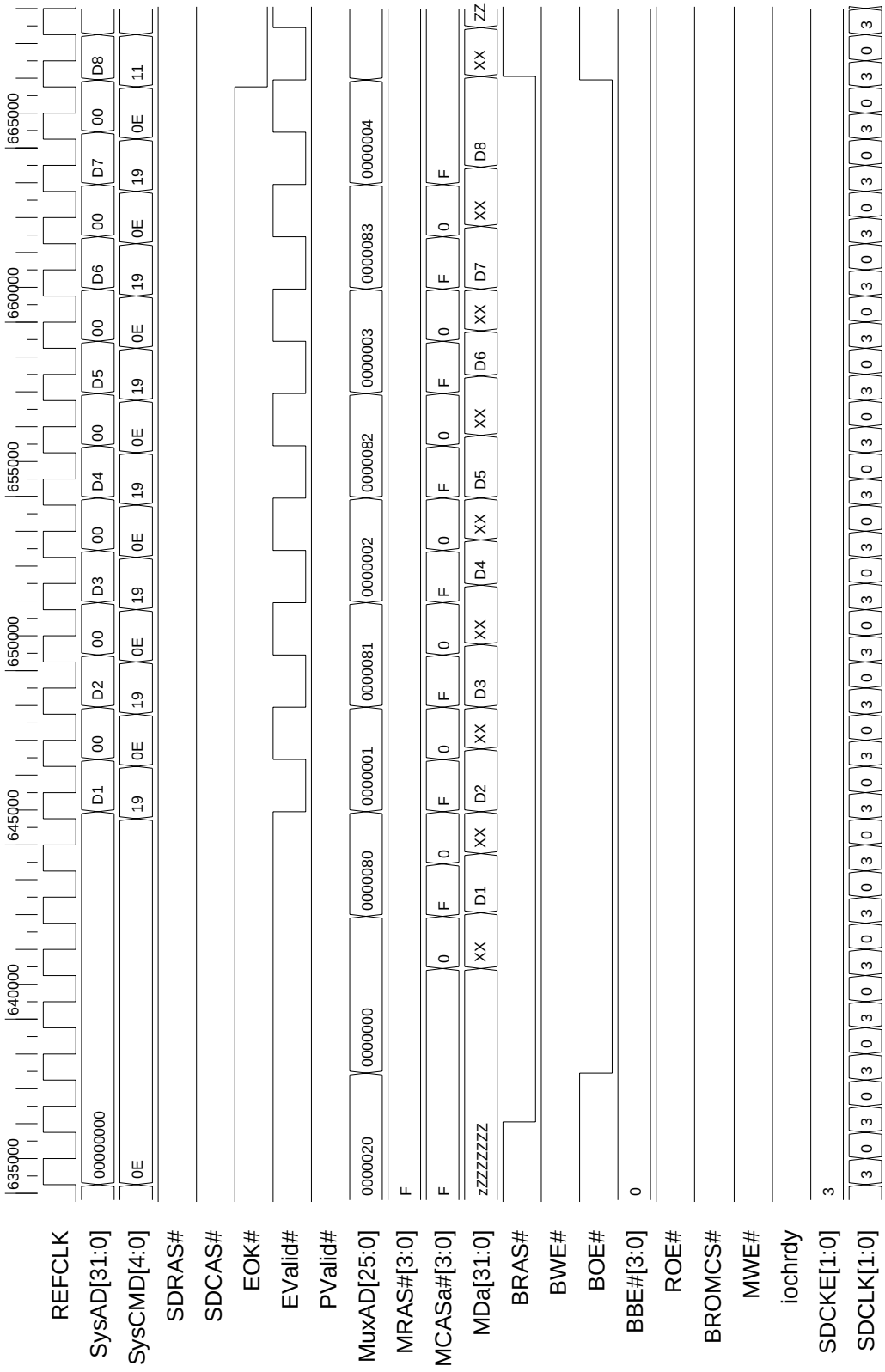
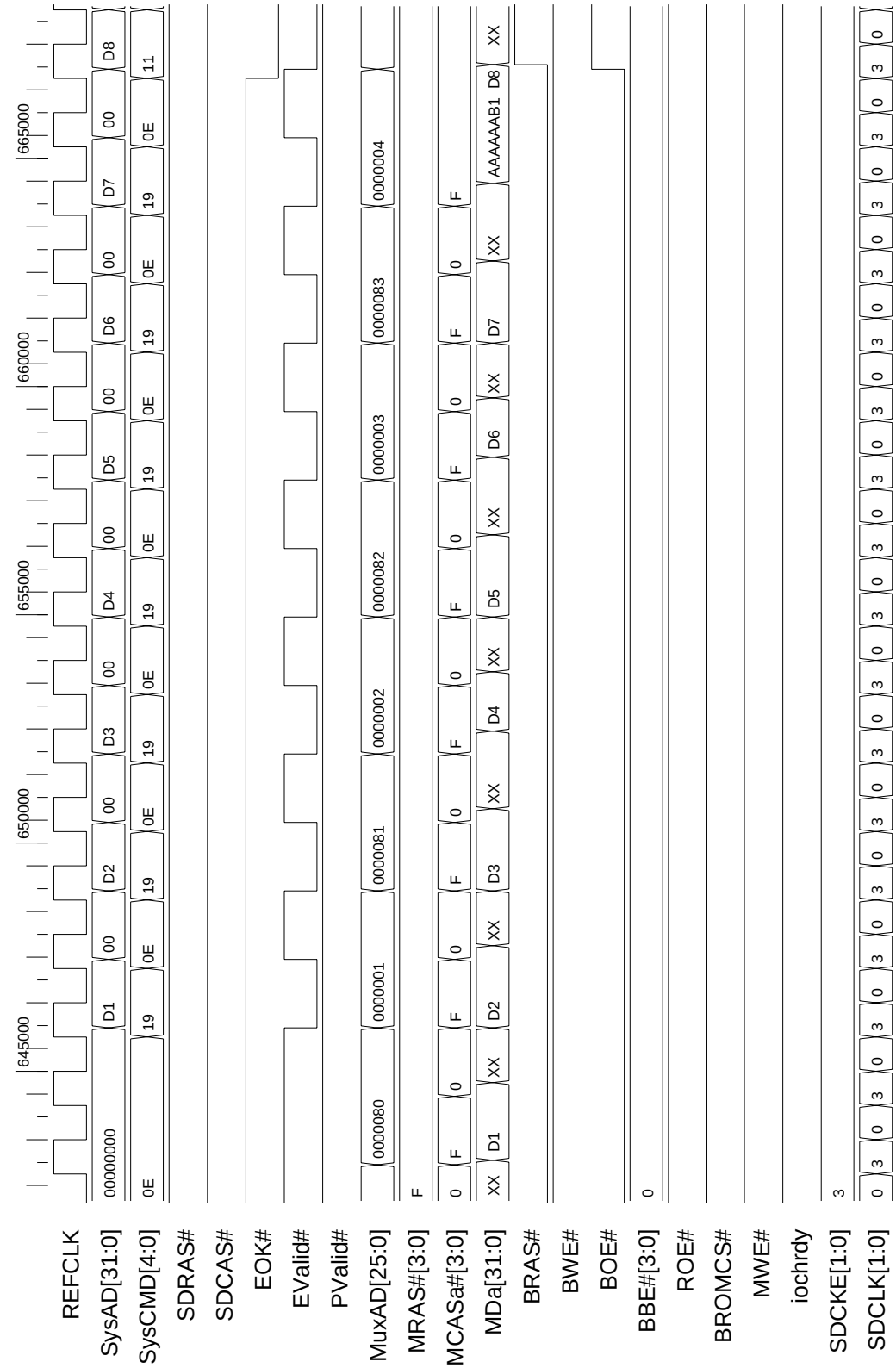


Figure 37. CPU Octet Read from EDO Base Memory (3 of 3)



13.2

DMA Timing

Figure 38. DMA Transfer

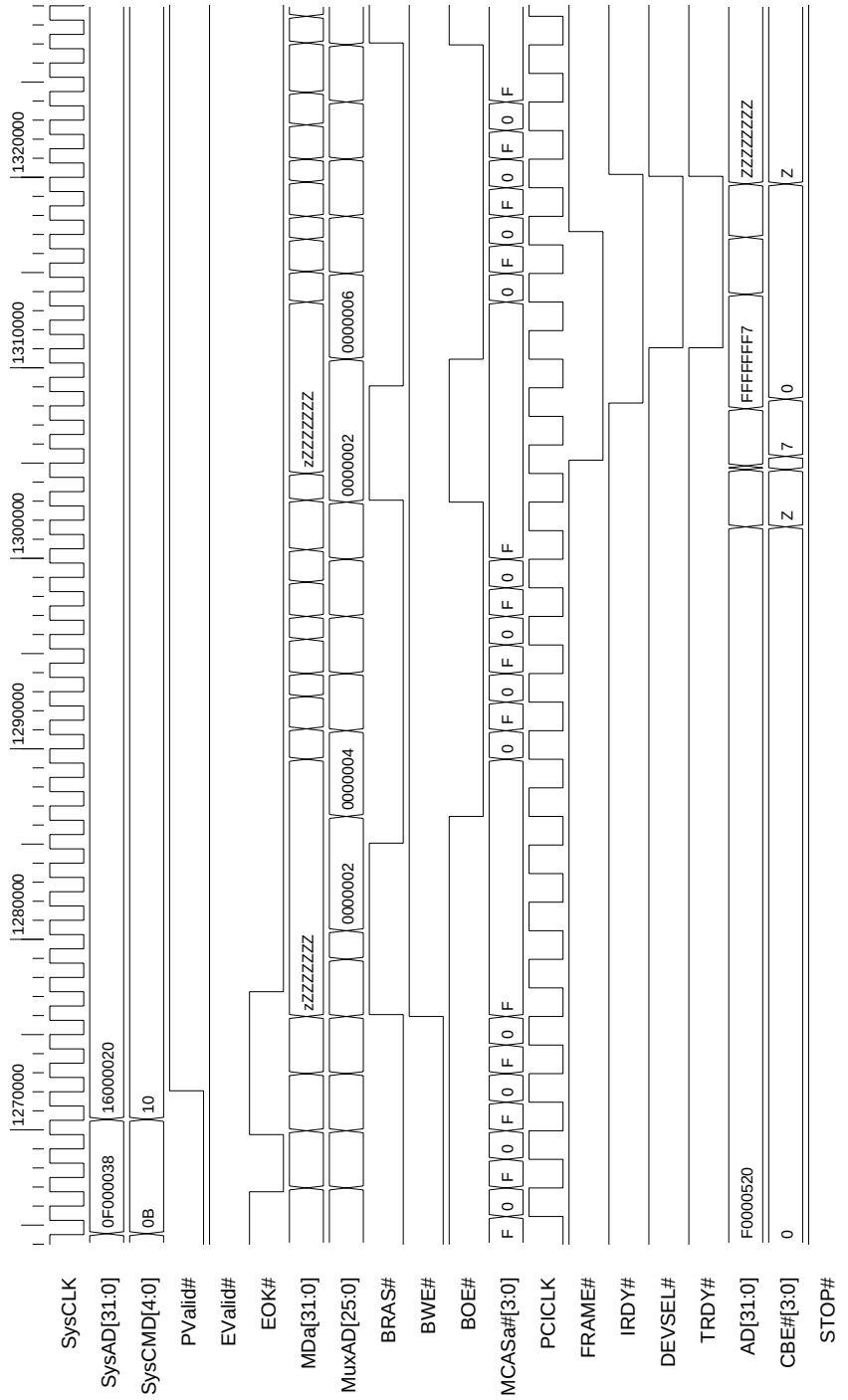


Figure 39. DMA Interrupt (1 of 2)

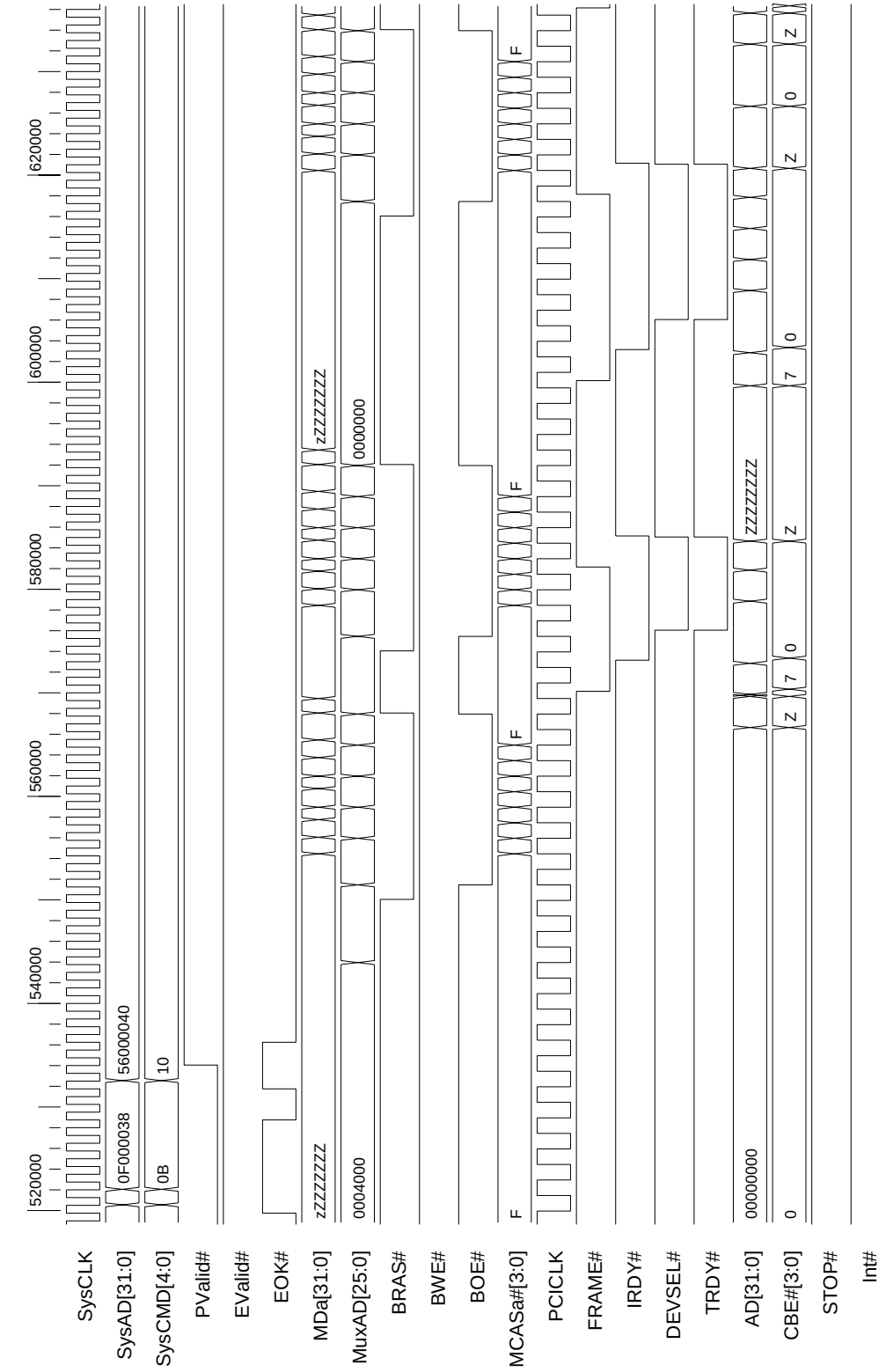


Figure 40. DMA Interrupt (2 of 2)

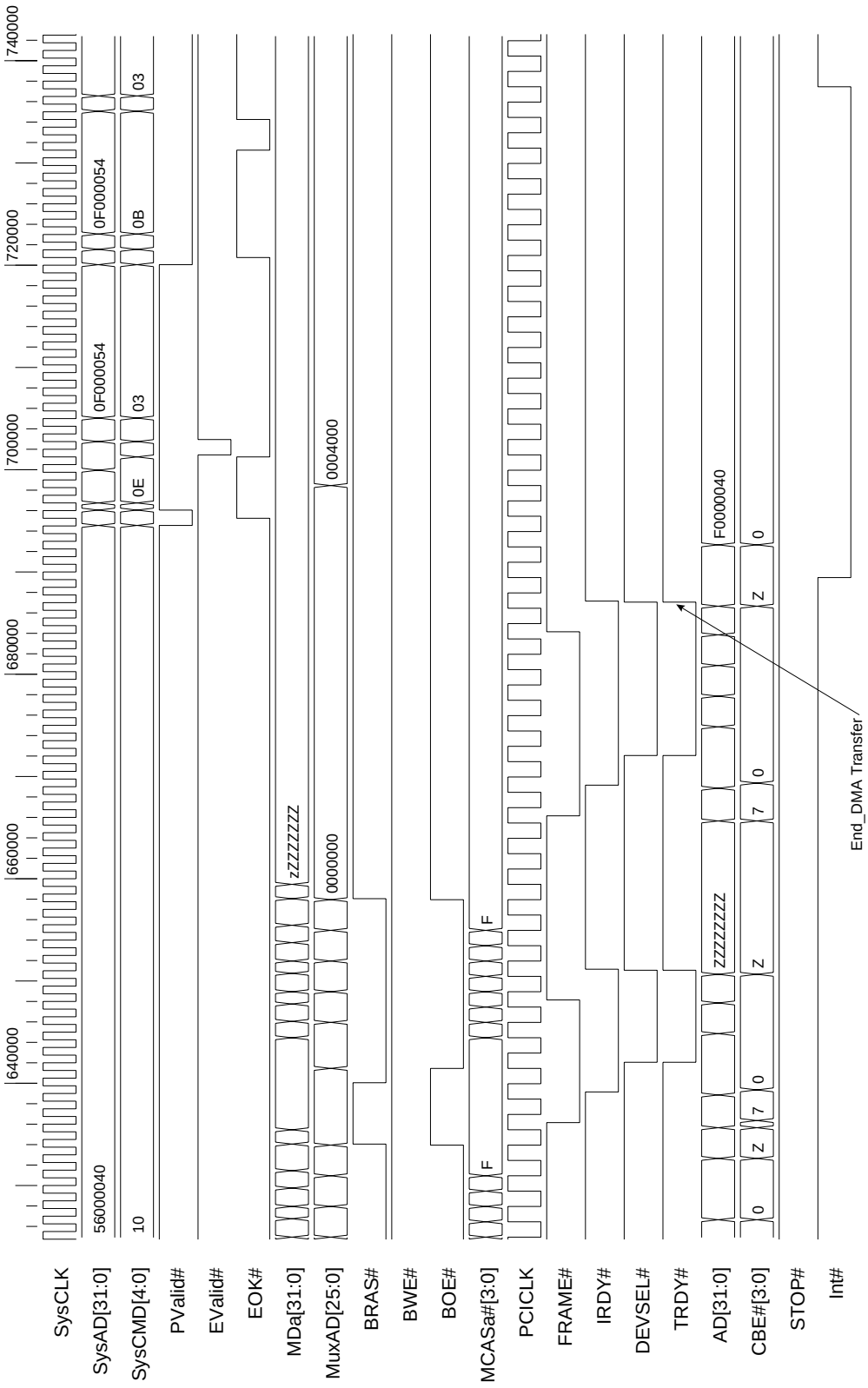


Figure 41. DMA Transaction Start (1 of 3)

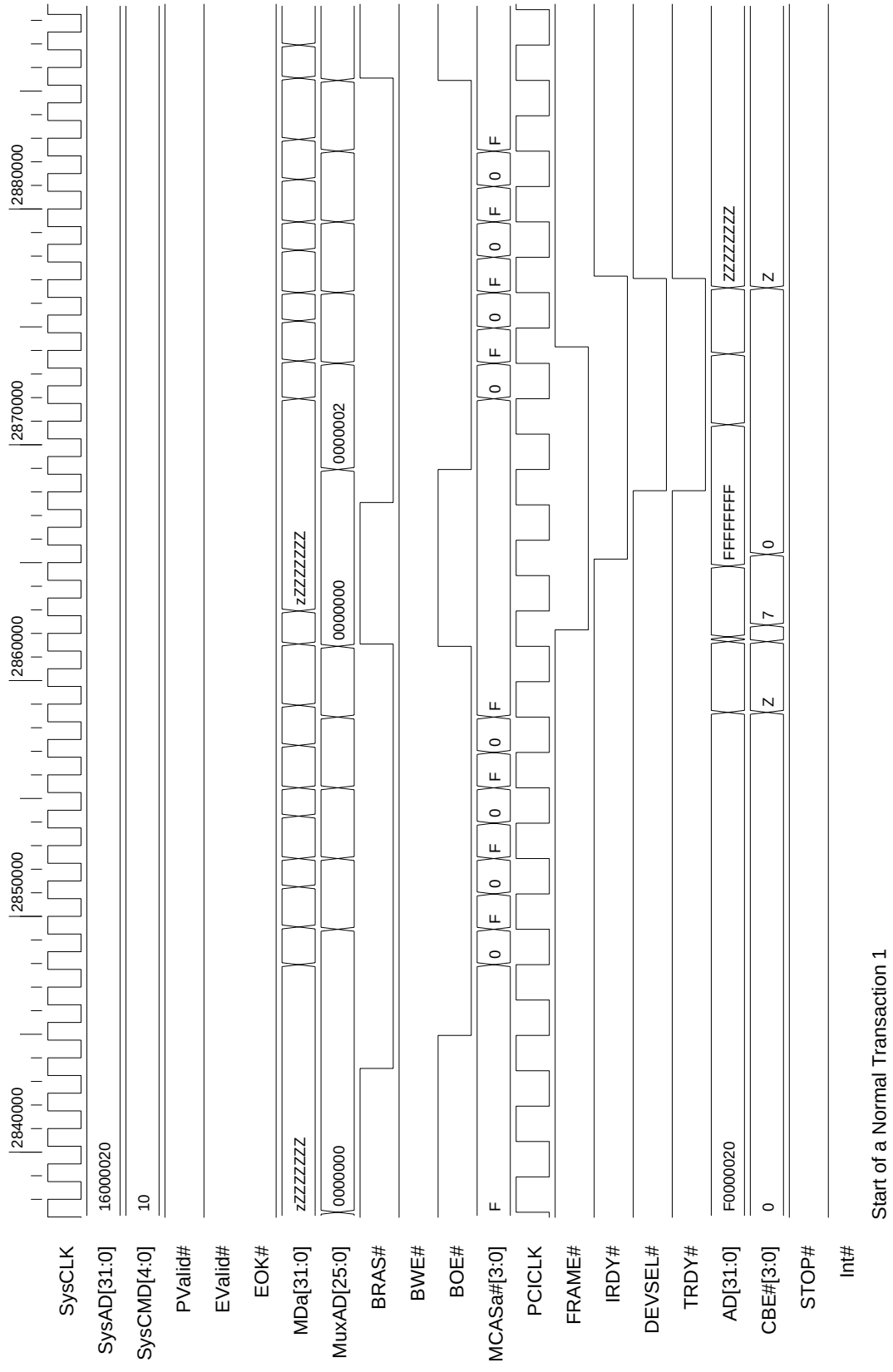


Figure 42. DMA Transaction End and Next Pointer Fetch (2 of 3)

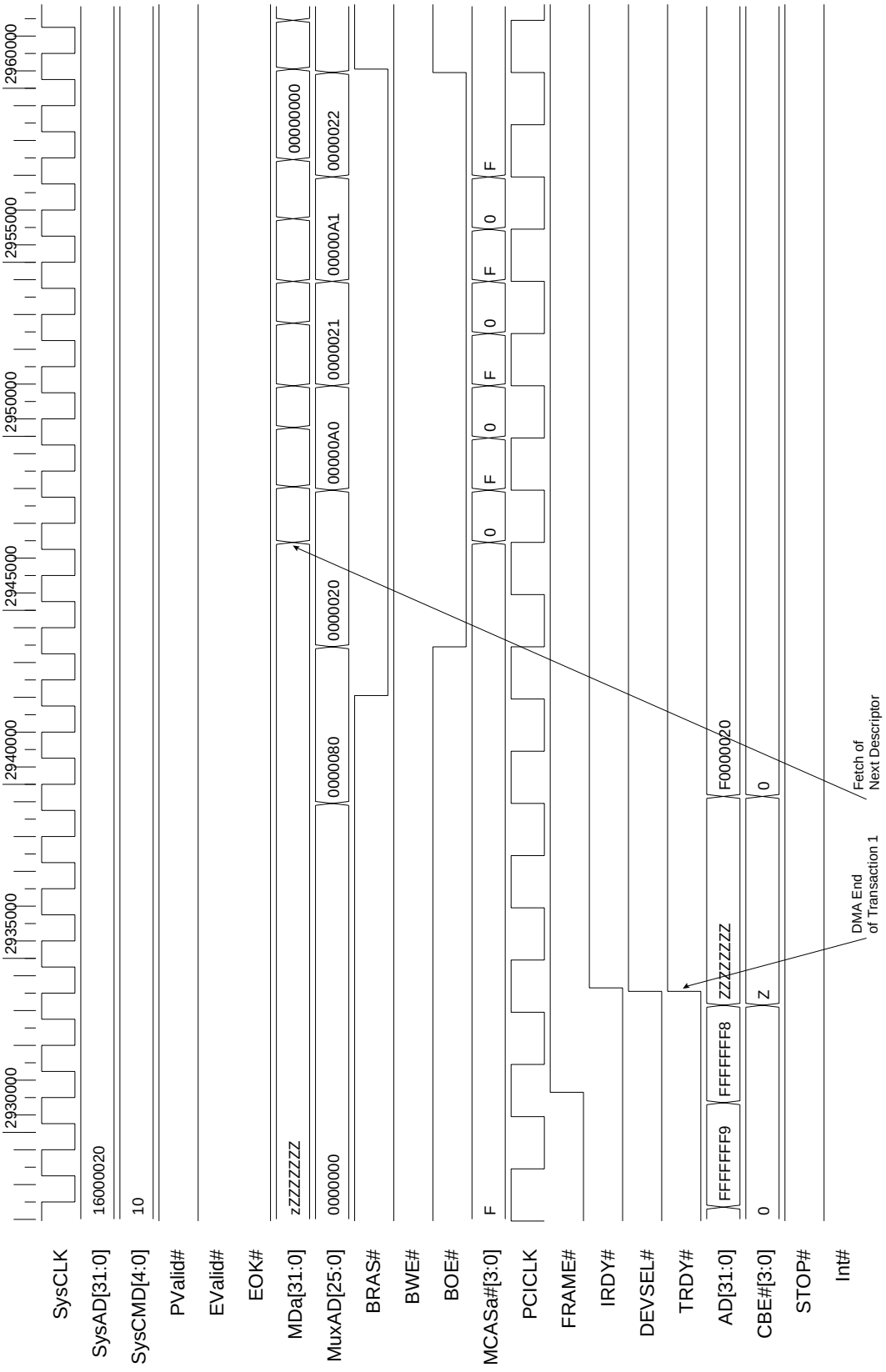
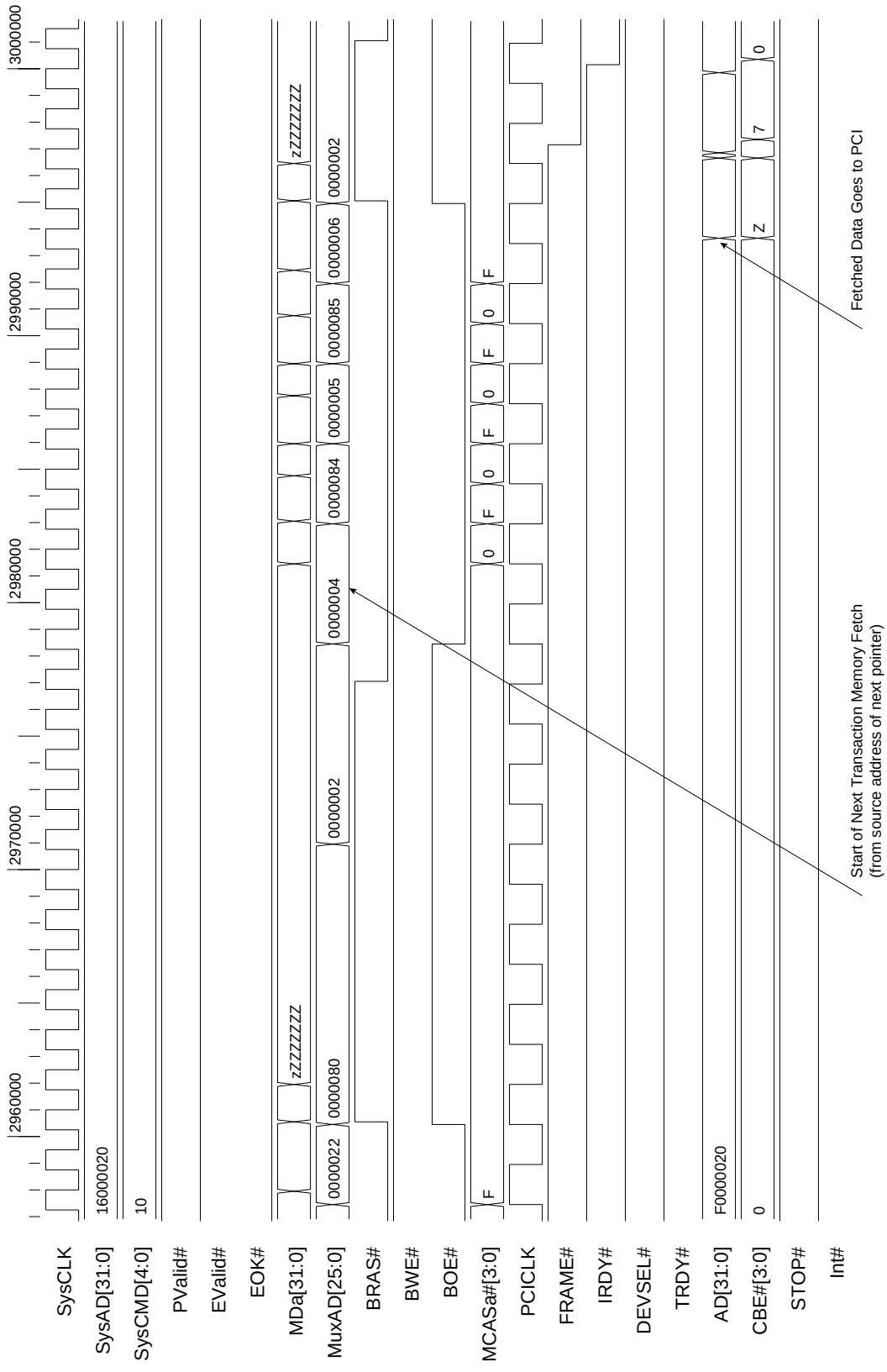


Figure 43. DMA Start of Next Transaction (3 of 3)



13.3

PCI Timing

Figure 44. PCI Burst Write: 16 Words

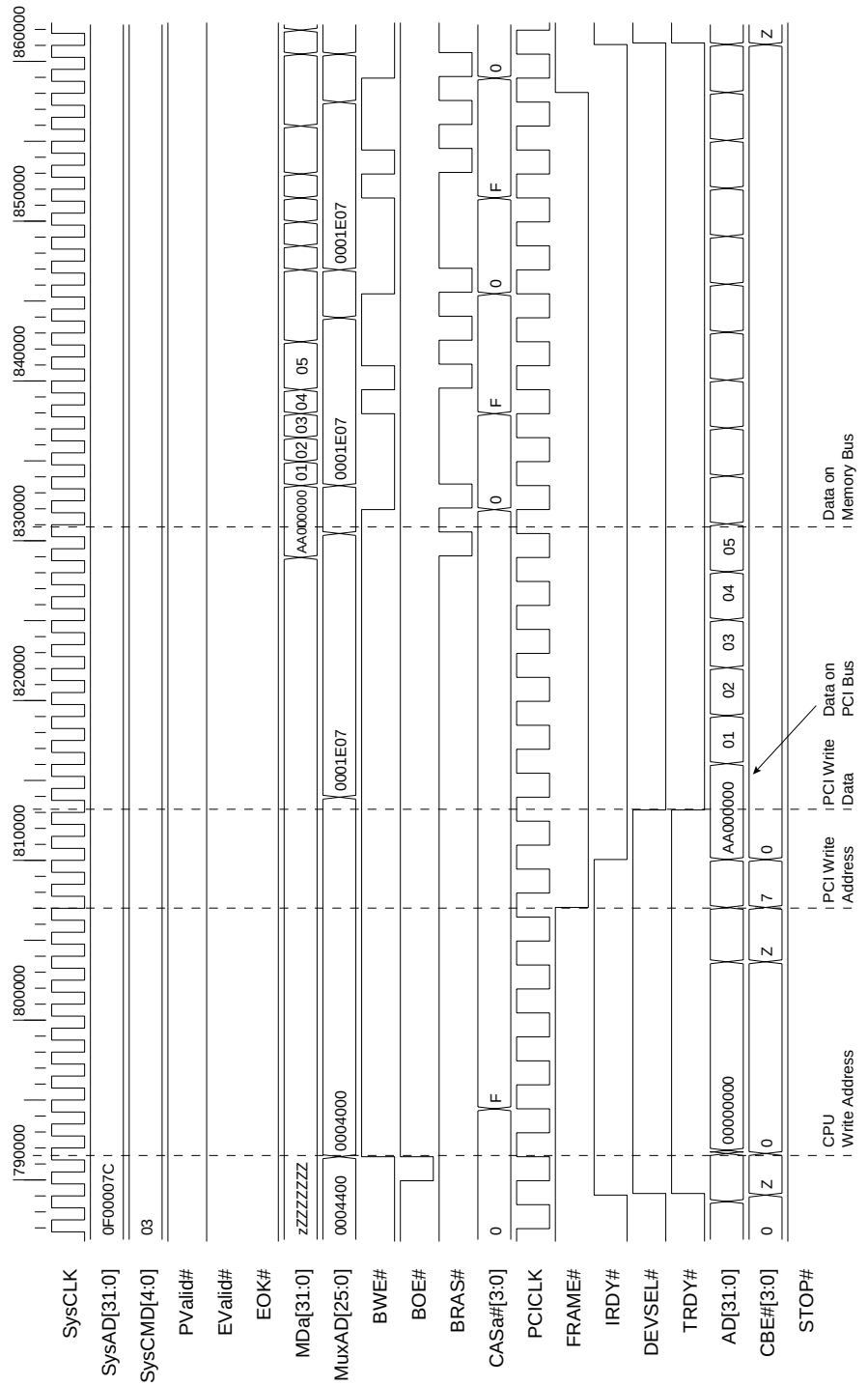


Figure 45. PCI Burst Read: 16 Words

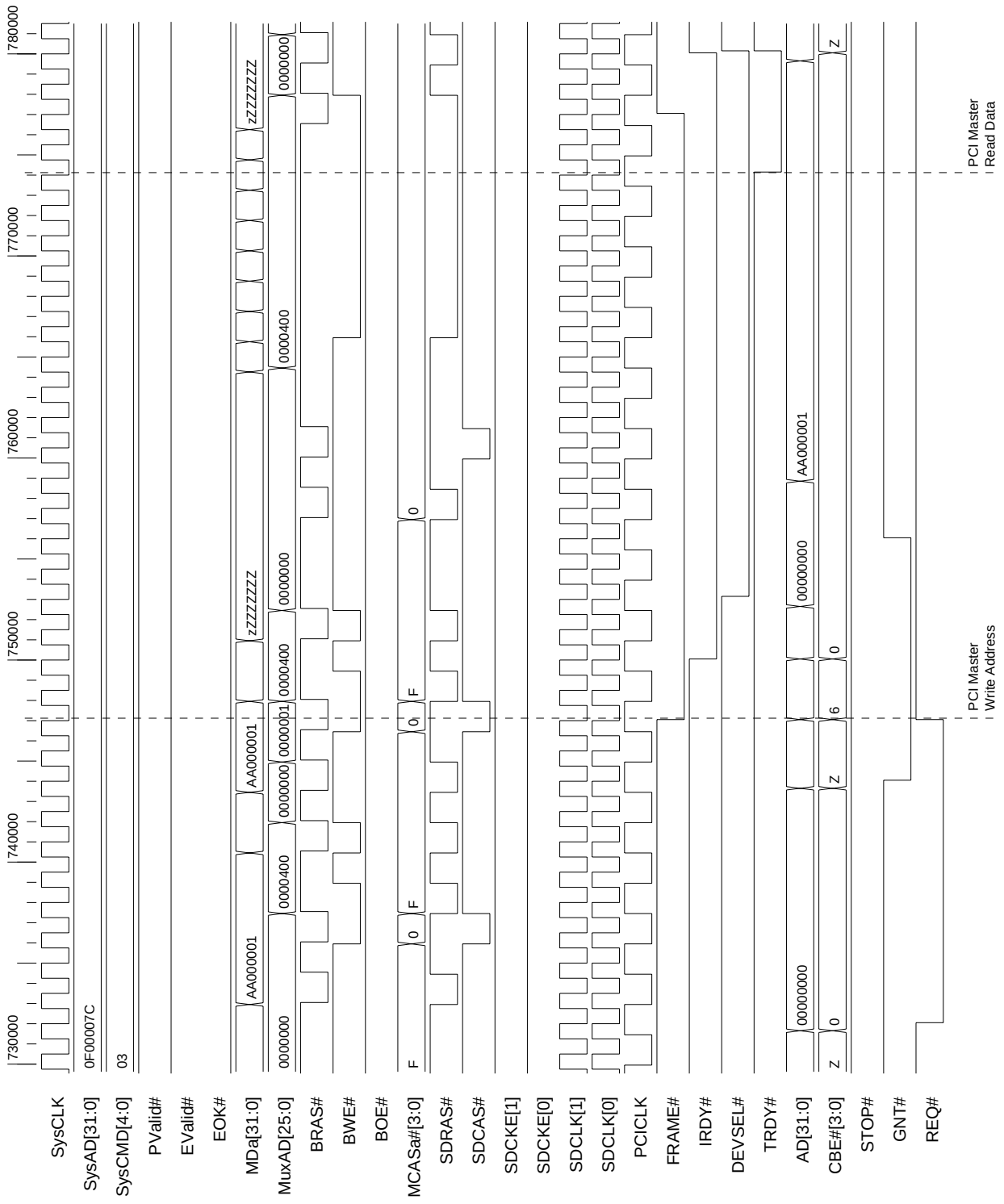


Figure 46. PCI Burst Write: 32 Words

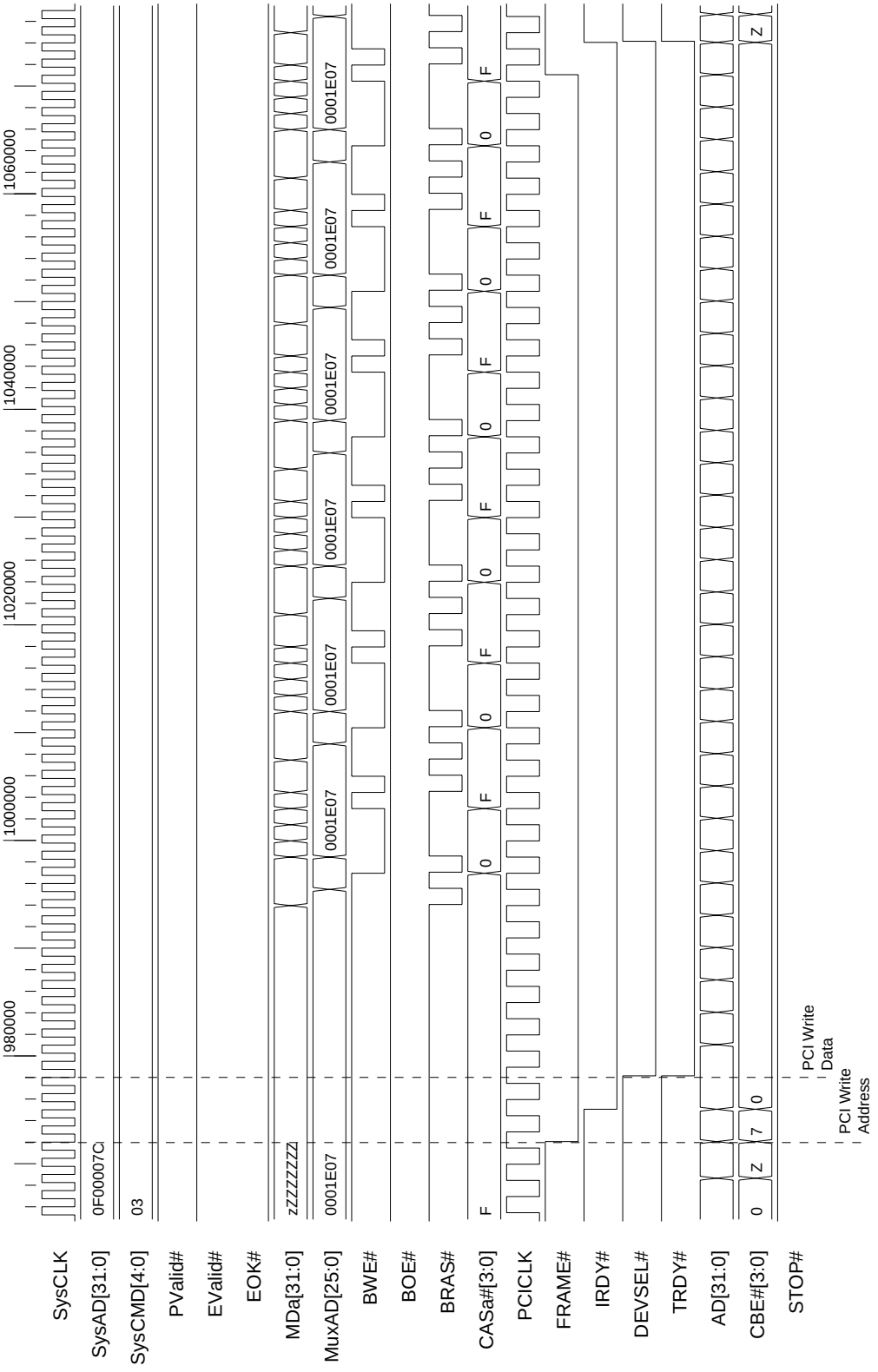


Figure 47. PCI Burst Read: 32 Words

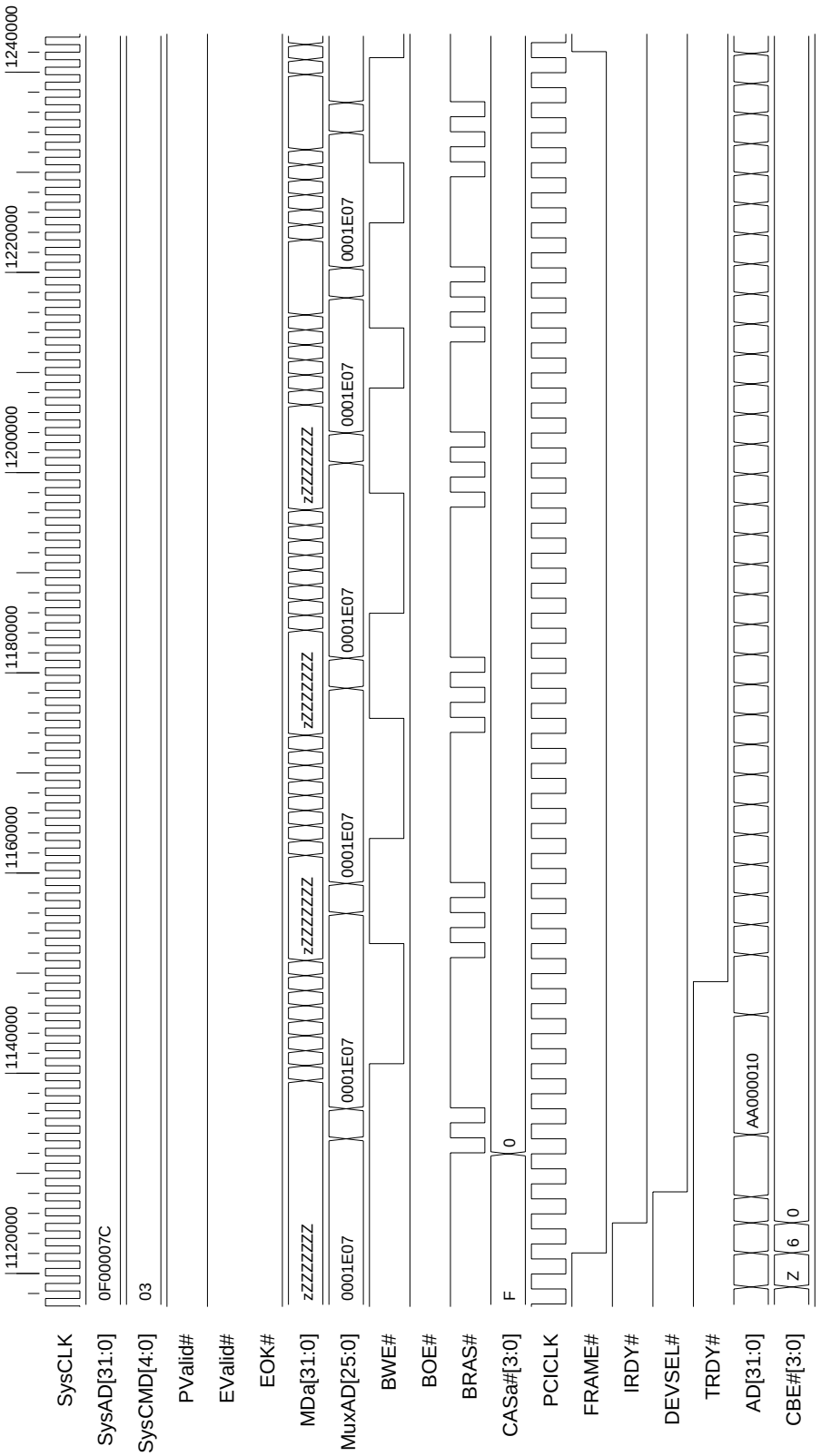


Figure 48. PCI Write: 20 Words Back to Back

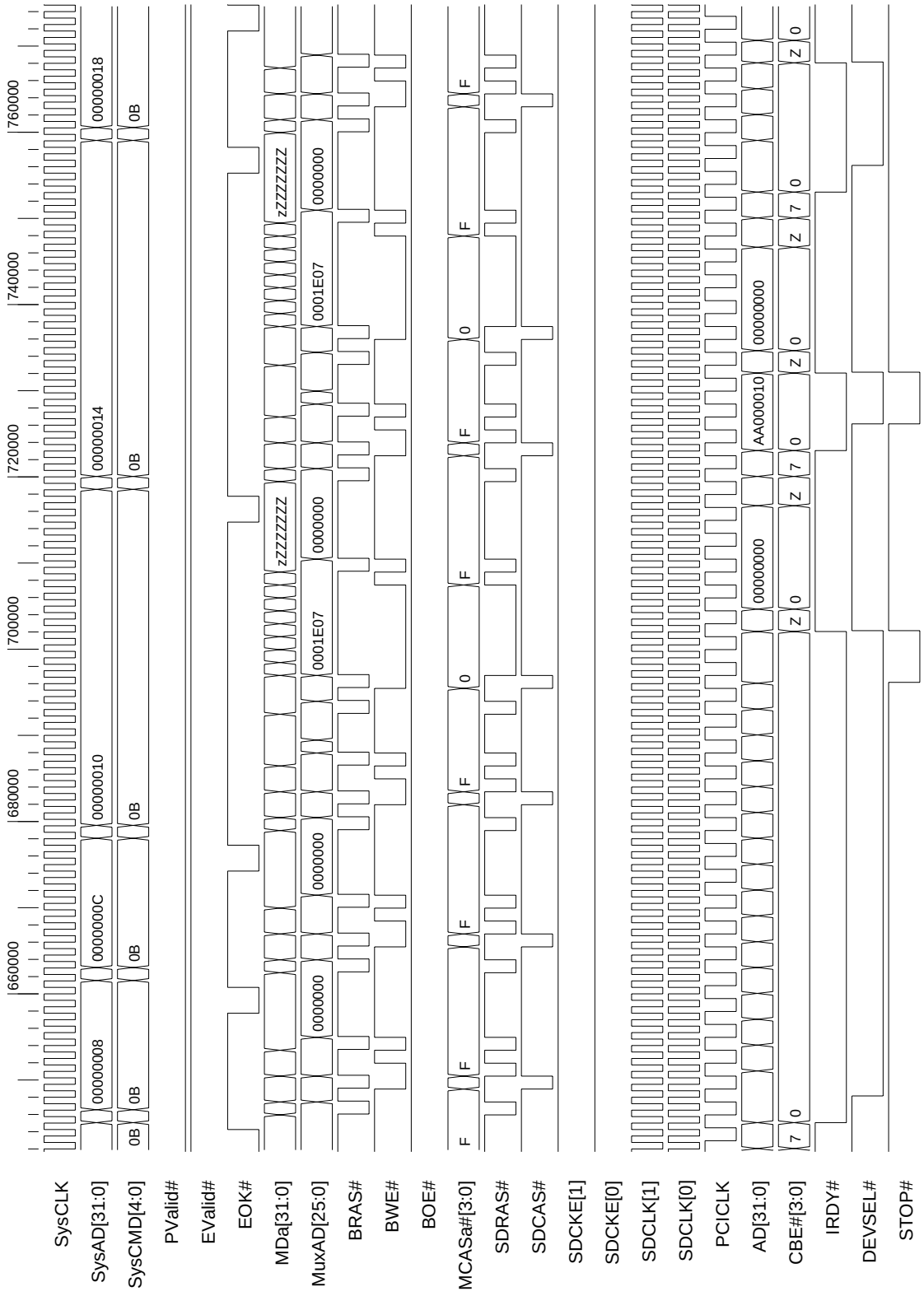


Figure 49. CPU 3-Byte Read/Write from PCI Memory

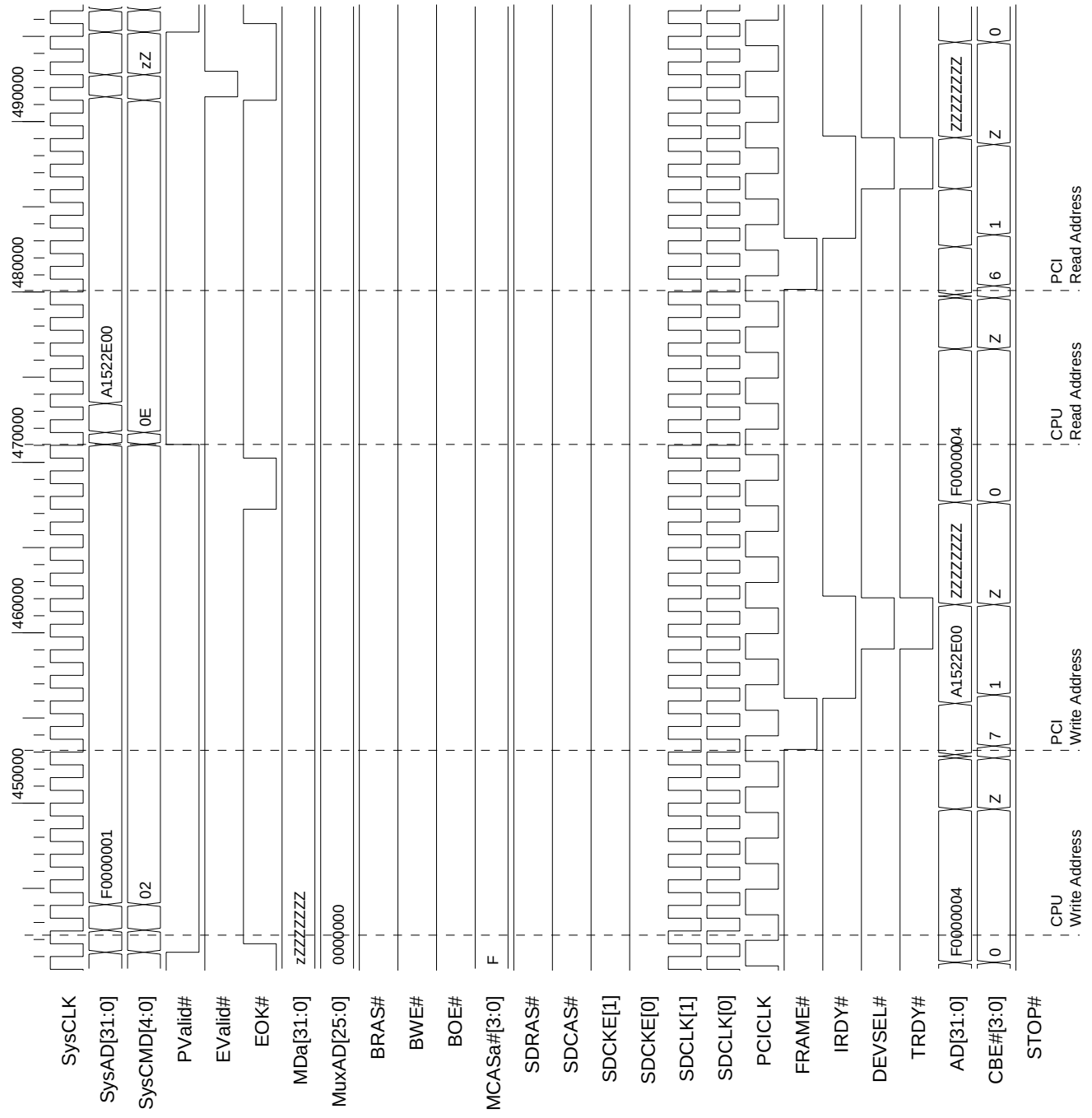


Figure 50. CPU 2-Byte Read/Write from PCI Memory

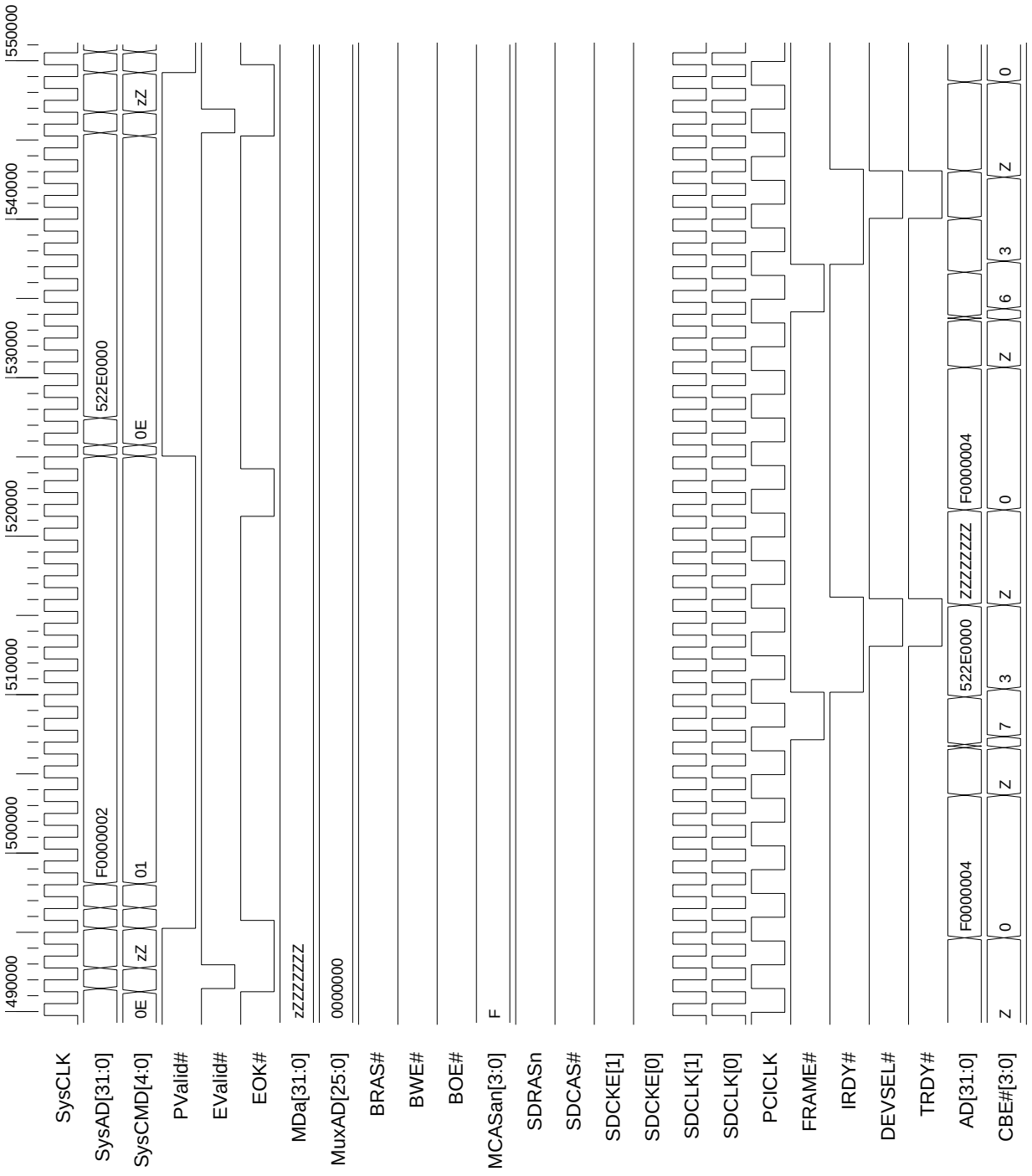


Figure 51. CPU 1-Byte Read/Write from PCI Memory

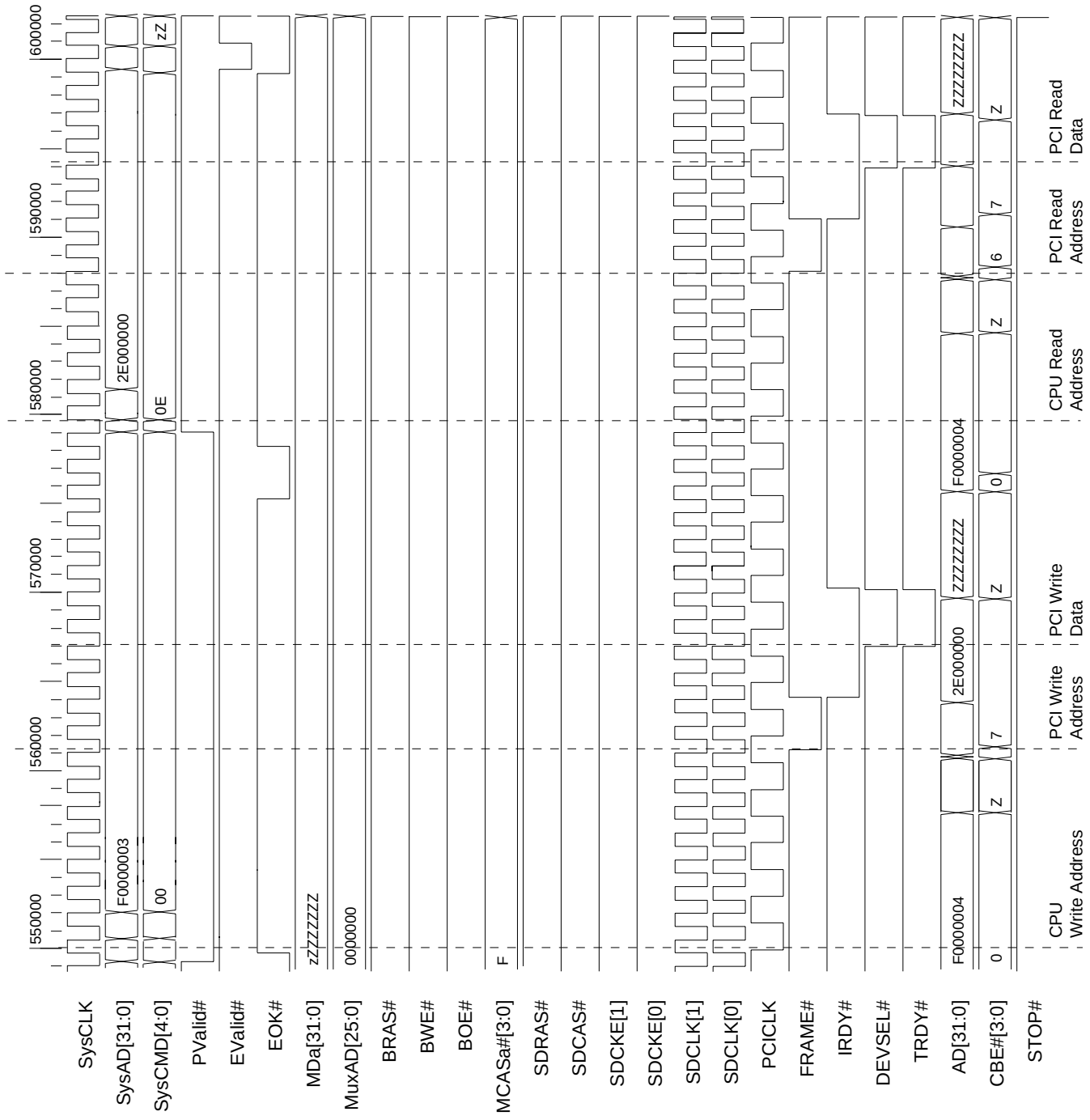


Figure 52. PCI to Controller Single 8-Byte Write

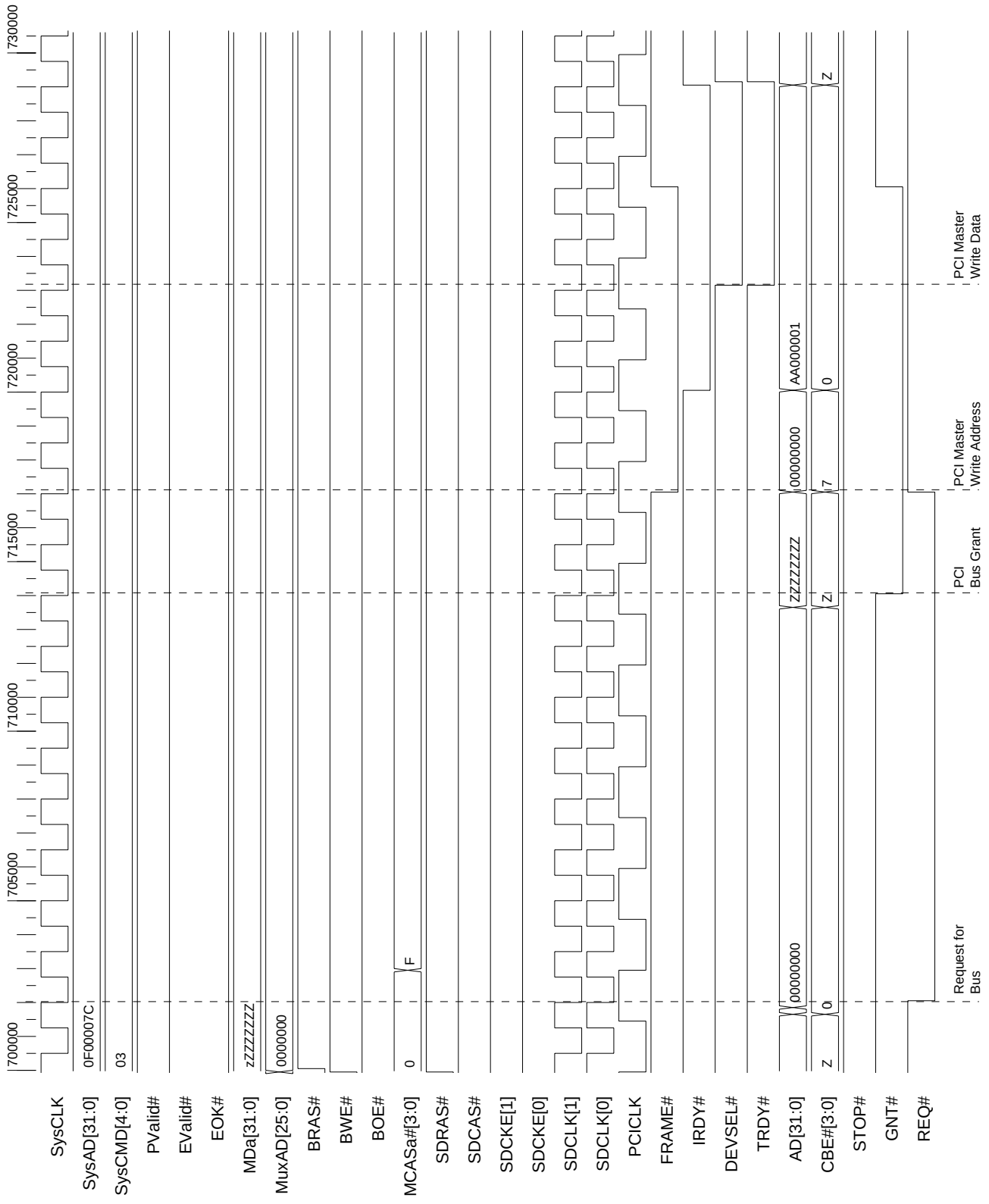


Figure 53. PCI to Controller Single 8-Byte Read

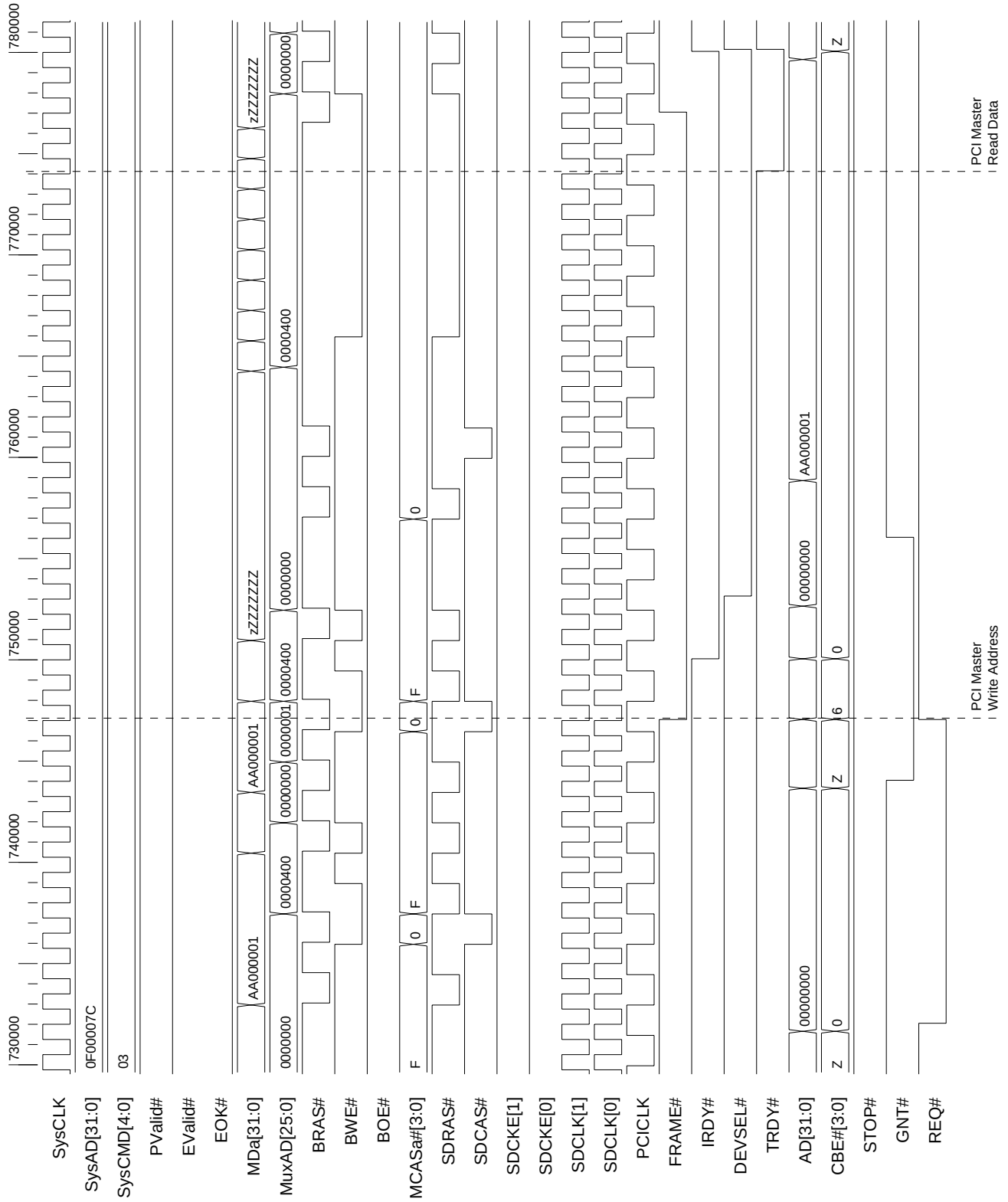


Figure 54. PCI to Controller Burst Write: 16 Words

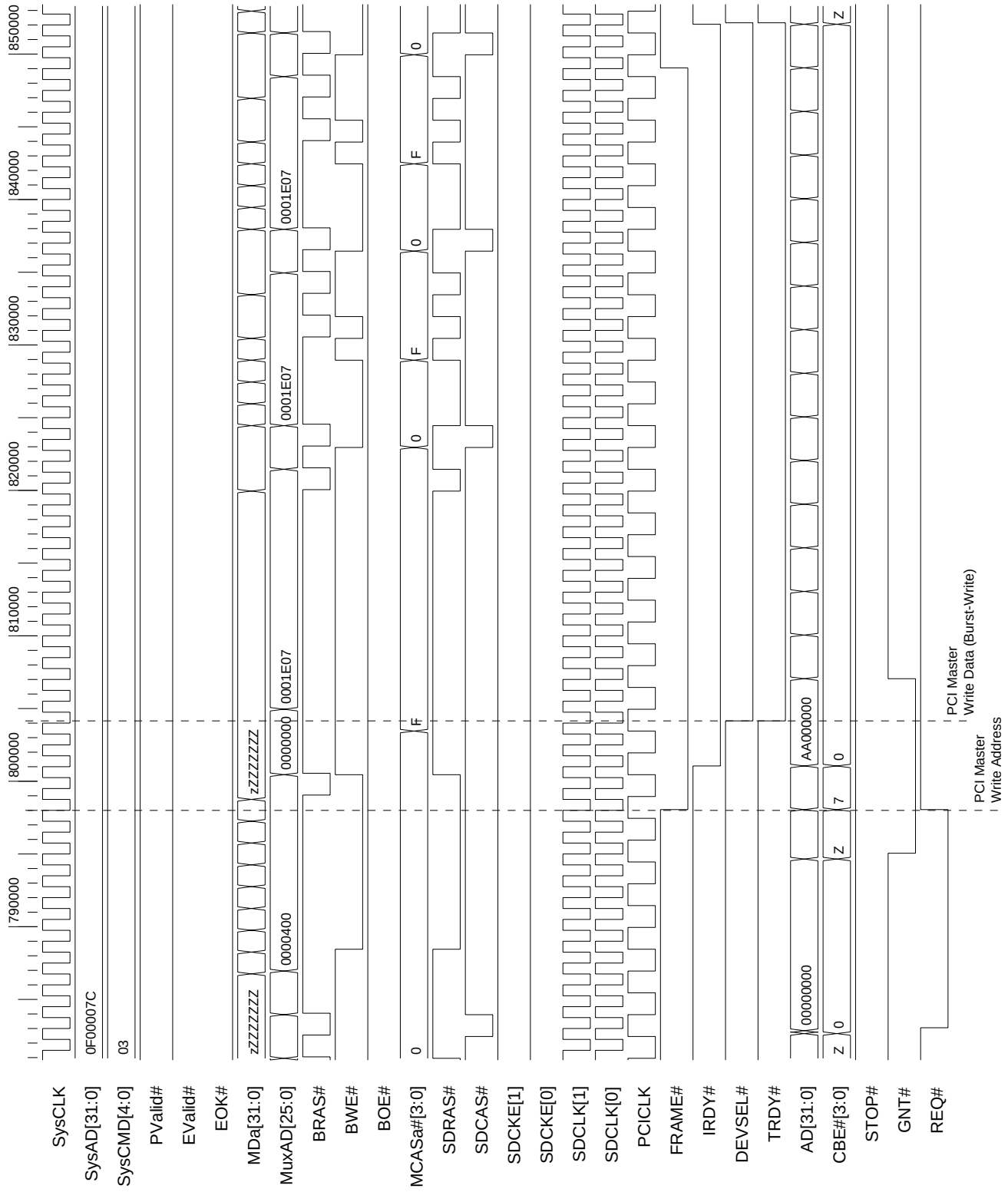


Figure 55. PCI to Controller Burst Read: 16 Words

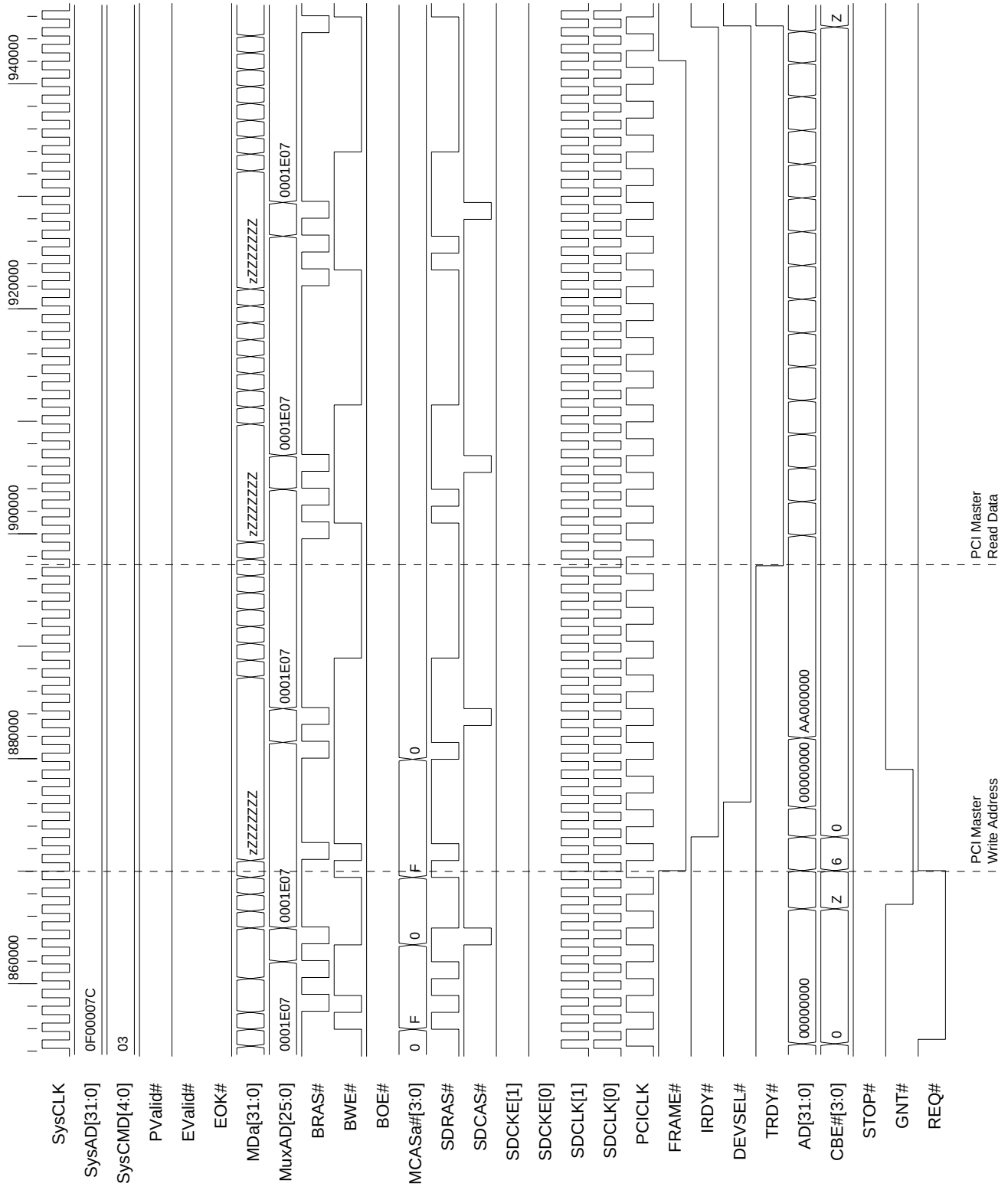


Figure 56. PCI to Controller 1-Byte Write

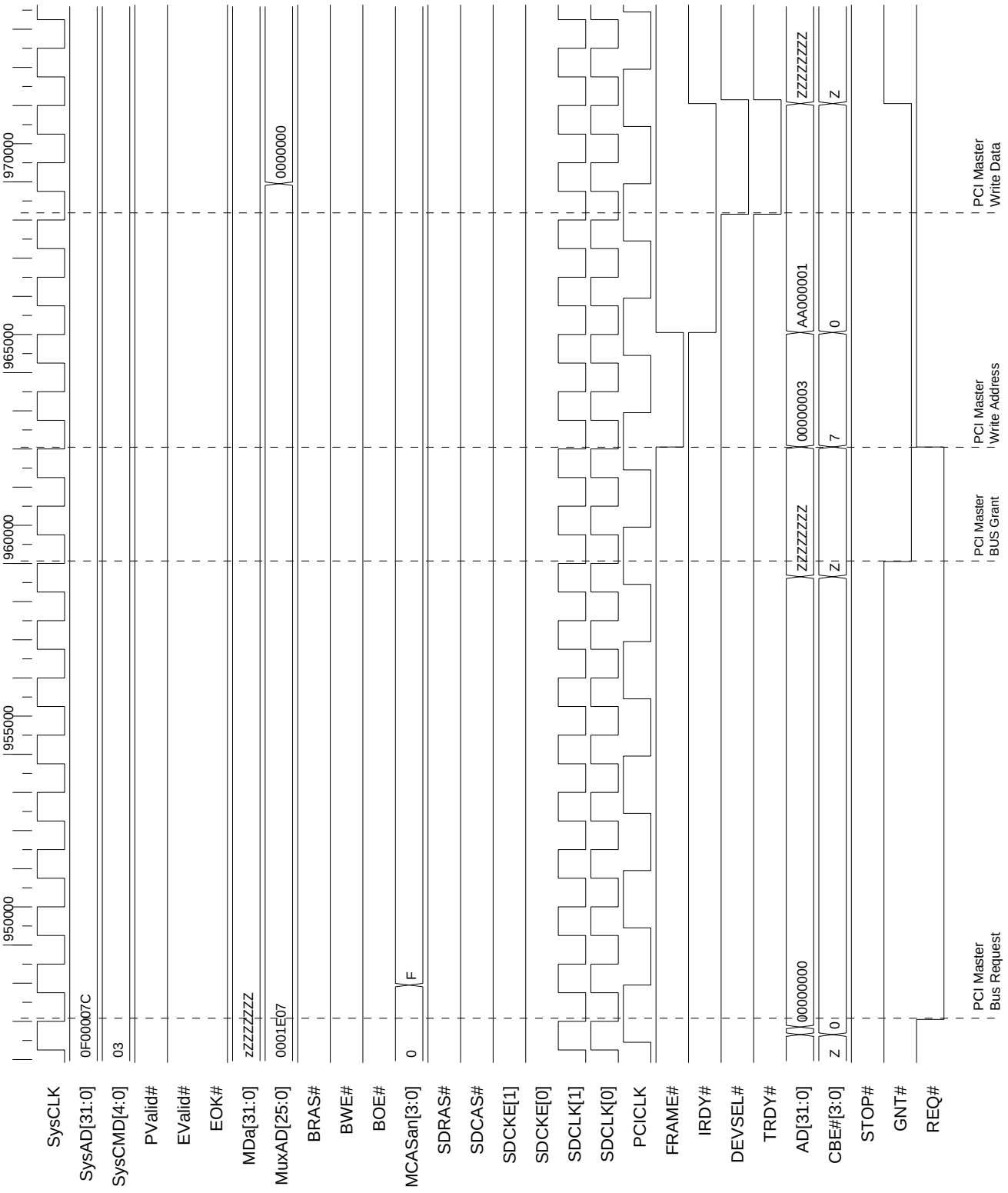


Figure 57. PCI to Controller 1-Byte Read

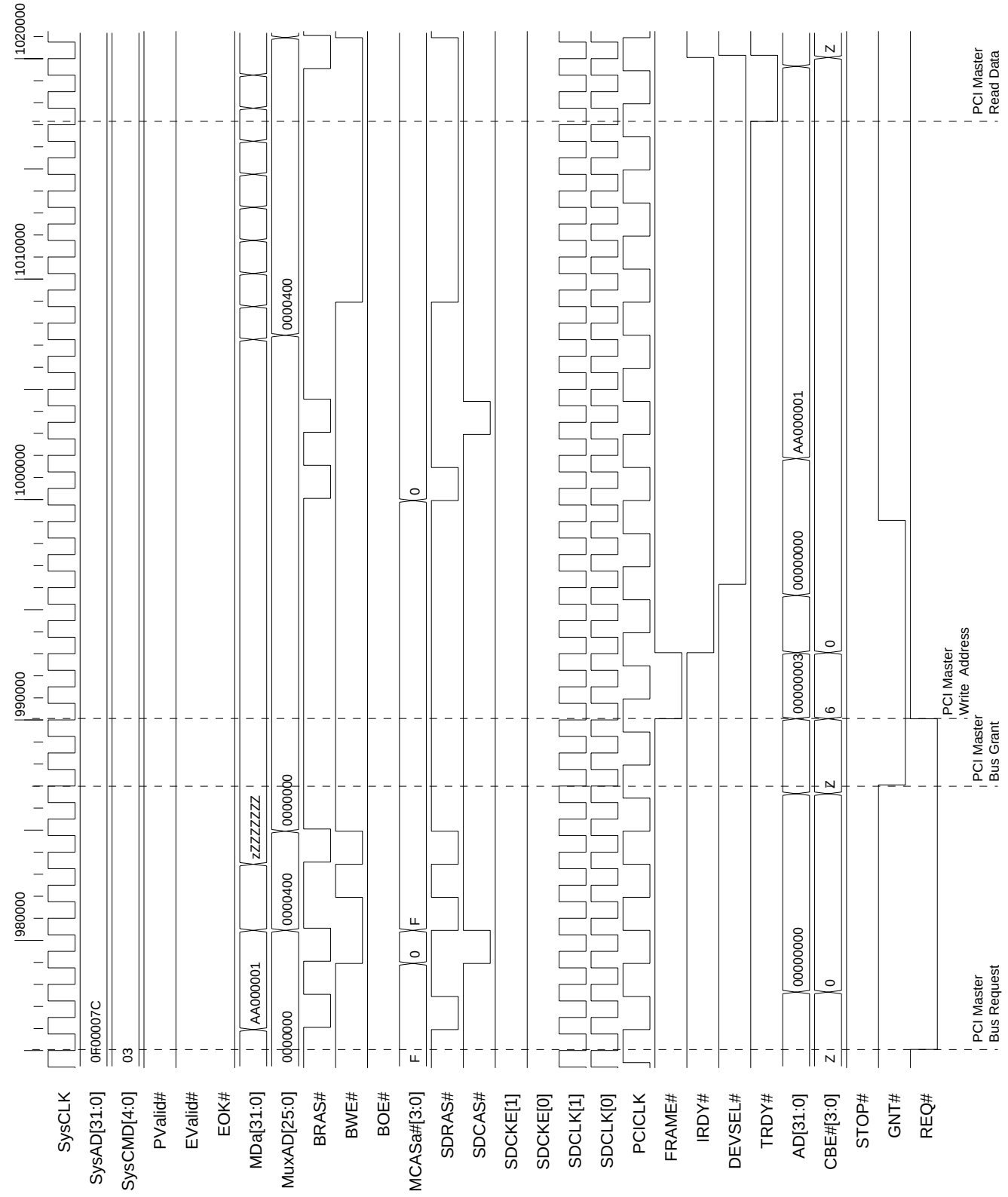


Figure 58. PCI and CPU Simultaneous Write: 8 Words

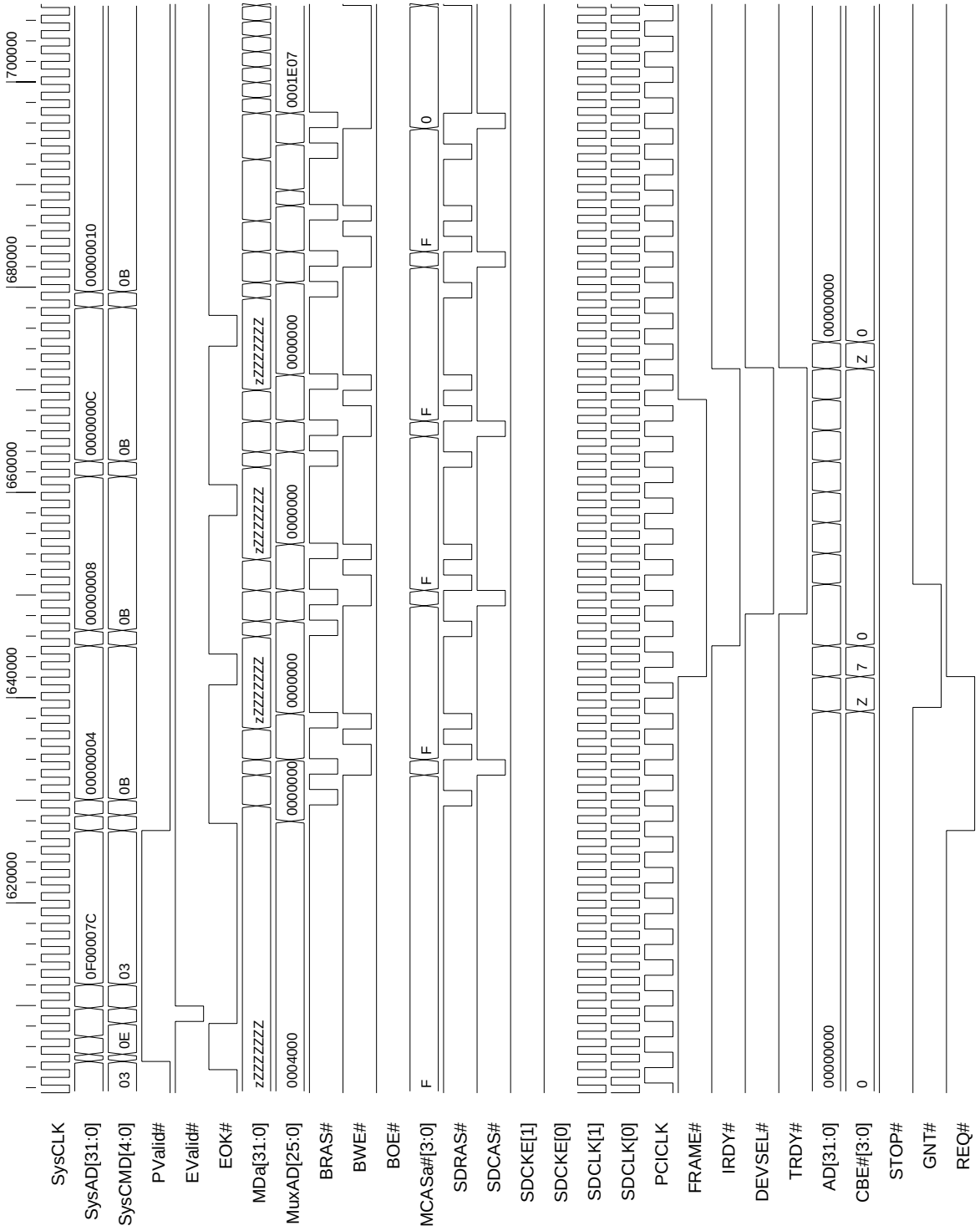


Figure 59. PCI and CPU Simultaneous Read: 8 Words

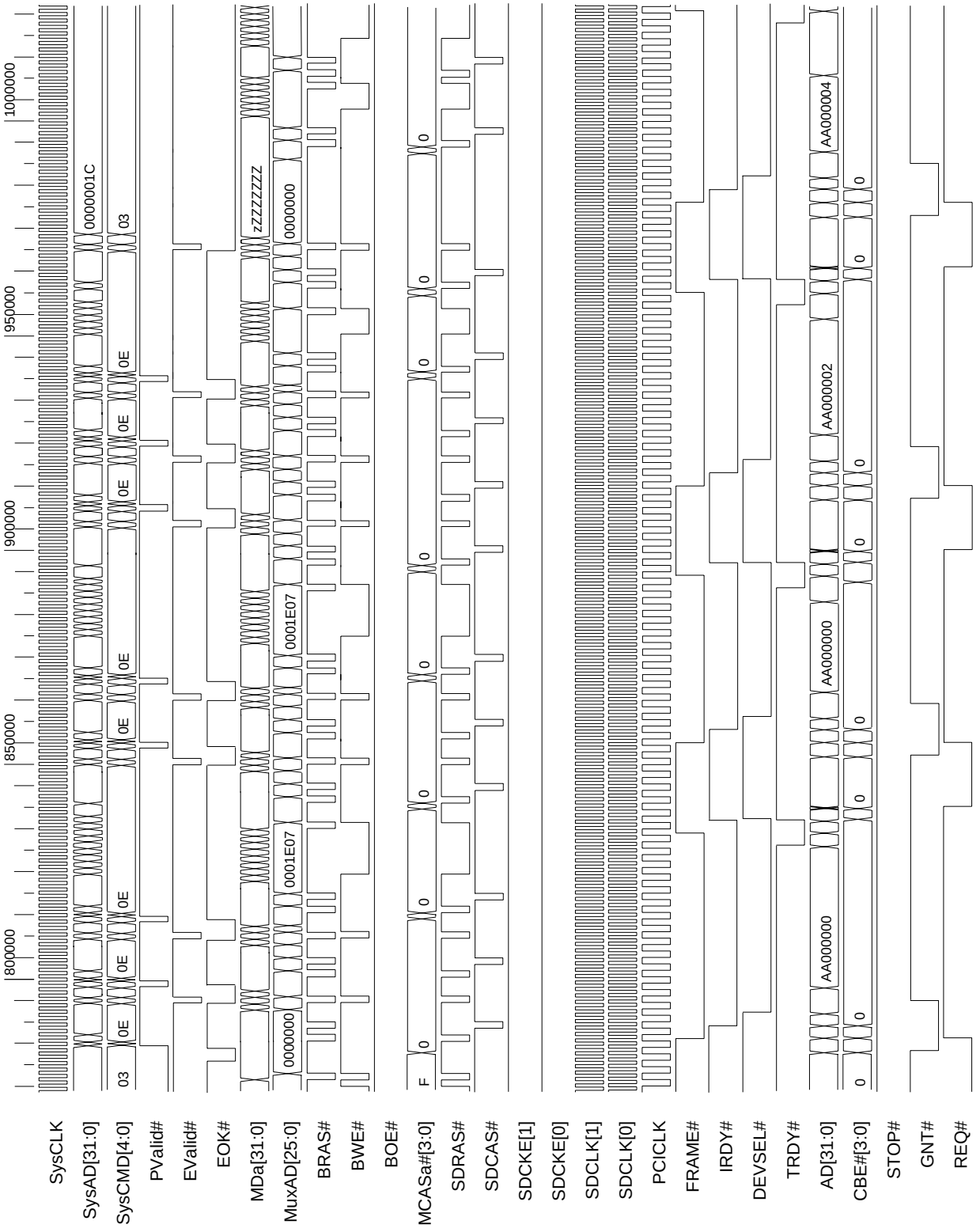


Figure 60. PCI to Controller Read: 2 Words

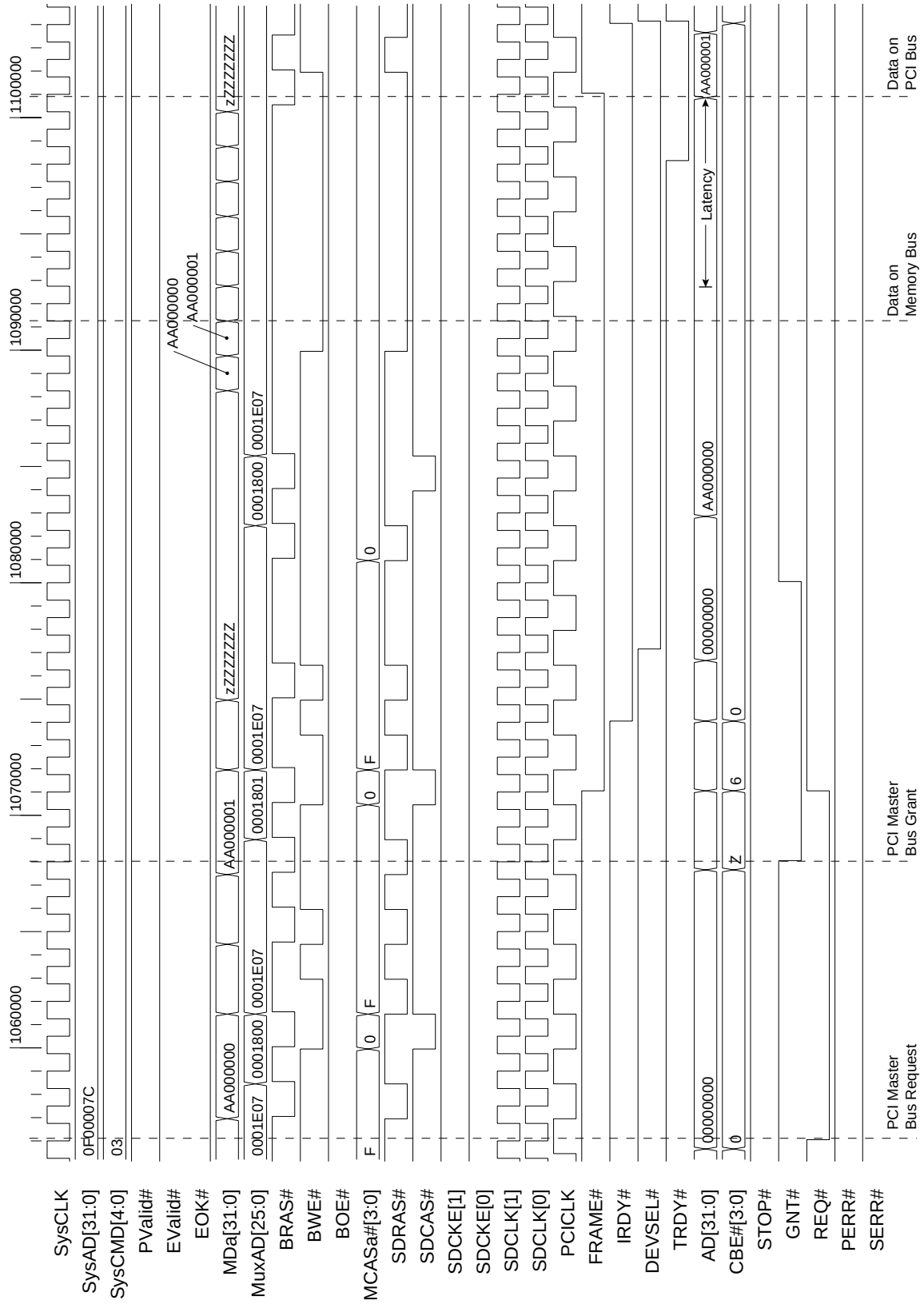
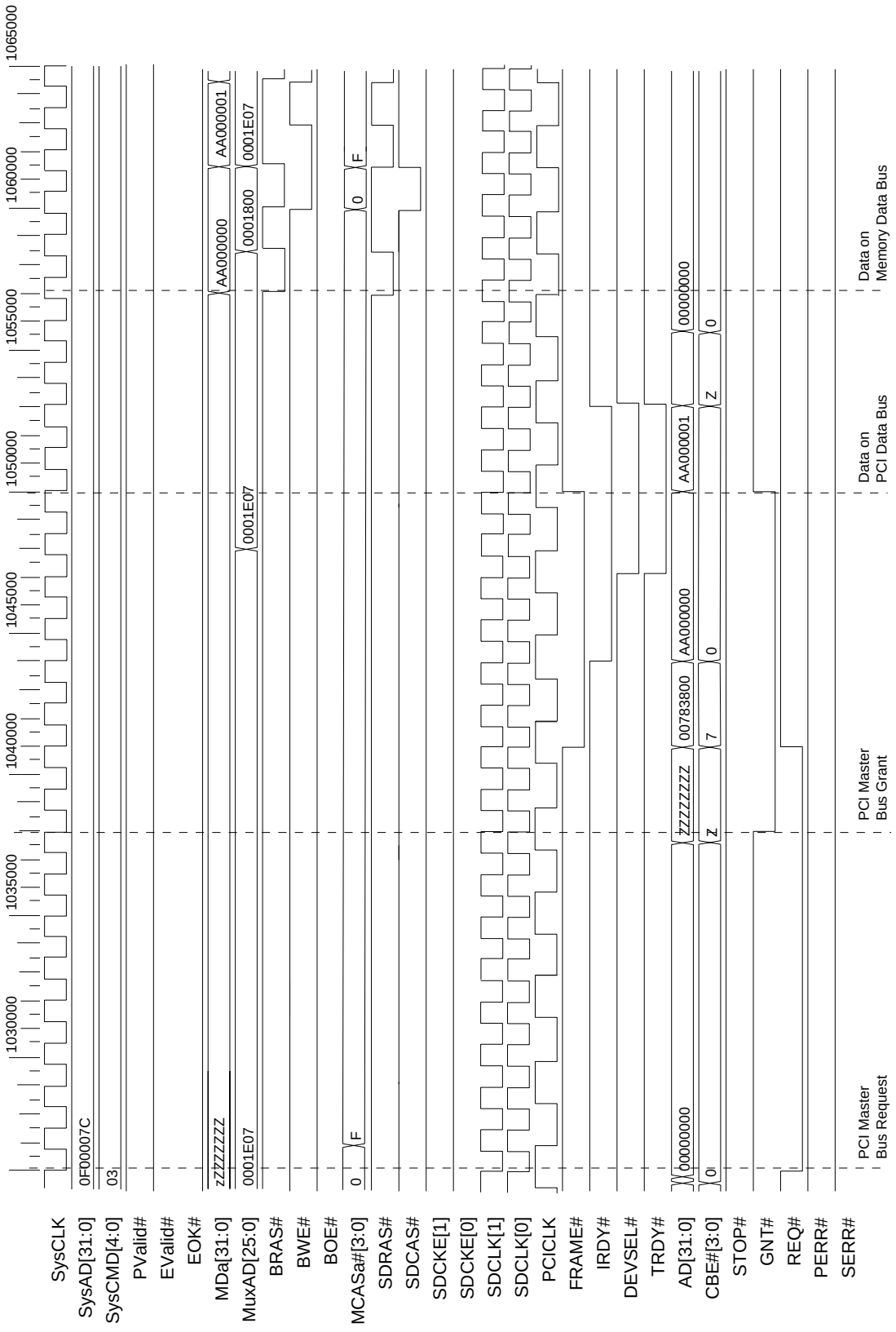


Figure 61. PCI to Controller Write: 2 Words



14.0 Electrical Characteristics

Caution: If any of the parameters exceed the specified value, the device may be physically damaged.

14.1 Absolute Maximum Ratings

Table 34. Maximum Ratings

Parameter	Maximum Rating
Storage temperature	–55°C to 125°C
Operating ambient temperature	0°C to 70°C
DC supply voltage with respect to GND	–0.5 V to 4.6 V
DC voltage on input pins with respect to GND	–0.5 V to 5.5 V
Voltage discharged between any two pins through a 1-K Ω resistor from 100 pF	2000 V
Maximum power dissipation	1 W

14.2 Operating Conditions

Table 35. Operating Conditions

Symbol	Parameter	Minimum	Typical	Maximum	Units
V_{DD}	All power pins	3.0	3.3	3.6	V
I_{DSS}	Static current consumption		2.0	300	µA
T_C	Case temperature	–40		+85	°C
T_I	Junction temperature	–40		+125	°C
t_r	Input rise time	0		200	ns
t_f	Input fall time	0		200	ns

14.3 DC Specifications

Table 36. DC Specifications for Input, Output, and Bidirectional Pins

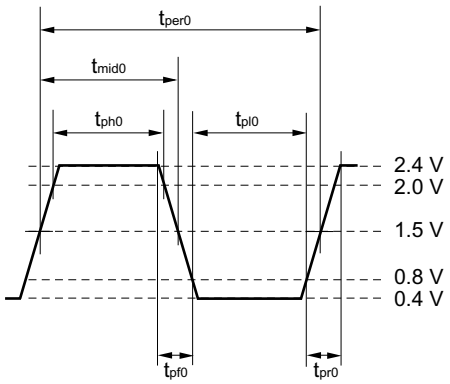
Symbol	Parameter	Minimum	Typical	Maximum	Units	Notes
V_{IL}	Input low voltage	0		0.8	V	
V_{IN}	Input high voltage	2.0		V_{DD}	V	
V_{OL}	Output low voltage			0.4	V	
V_{OH}	Output high voltage	2.4			V	
I_I	Input leakage current with no pull-down resistor			±10	µA	$V_I = V_{DD}$ or GND (maximum)
I_I	Input leakage current with 50-K Ω internal pull-down resistance	28	83	190	µA	$V_I = V_{DD}$
I_{OZ}	Off-state output current			±10	µA	
I_{OS}	Output short-circuit current			–250	mA	$V_{IN} = V_{IL}$ to V_{IN}
C_{IN}	Input capacitance		7	10	pF	
C_{OUT}	Output capacitance		7	10	pF	
$C_{I/O}$	I/O capacitance		7	10	pF	

14.4 AC Specifications

14.4.1 PCI CLK[3:0] Outputs

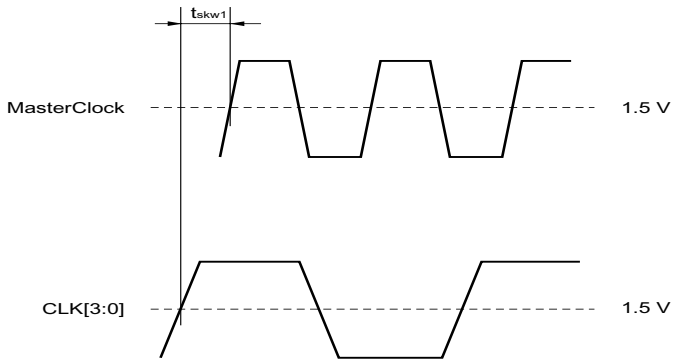
The PCI CLK[3:0] outputs must meet TTL input levels at the destinations. Figure 62, Figure 63, and Table 37 summarize the clock destination requirements per the *PCI Local Bus Specification*. See Section 10.0 for details about clock distribution and routing.

Figure 62. PCI CLK Input Waveform at Destinations



4373-001.eps

Figure 63. CLK[3:0] versus MasterClock Output Skew at Controller Pins



4373-002.ep

Table 37. CLK[3:0] and MasterClock Destination Timing Requirements

Symbol	Parameter	Minimum	Maximum	Units	Notes
Tper0	CLK[3:0] period	30		ns	
Tmid0	CLK[3:0] mid time	0.4 Tper0	0.6 Tper0	ns	
Tph0	CLK[3:0] high time	11		ns	
Tpl0	CLK[3:0] low time	11		ns	
Tpr0	CLK[3:0] rise time	1	8	ns	
Tpf0	CLK[3:0] fall time	1	8	ns	
Tjit0	Jitter CLK[n] to CLK[m] at controller output signals	0	0	ps	Loading on all clocks is exactly identical.
Tskw0	Skew CLK[n] to CLK[m] at destinations	0	1	ns	This includes ±200 ps of system jitter and 800 ps of system skew.
Tskw1	Skew CLK[3:0] at controller to MasterClock at controller	0	1		CLK[3:0] outputs must always be even or ahead of the MasterClock output when both are measured at controller signals.

14.4.2

PCI Inputs, Outputs, and Input/Outputs

The controller can be used in both 3.3- and 5-volt PCI systems. To accommodate both voltages, consideration must be given to switching and signaling levels, per the *PCI Local Bus Specification*.Table 38 reproduces Table 4-7 of the *PCI Local Bus Specification,* with substitutions made for VCC minimum and maximum values for the controller. Timing measurements on the controller PCI pads are taken with respect to a VTEST voltage of 1.5 volts. Setup and hold input times are measured from the 1.5-volt level on the rising edge of CLK[3:0] to the 1.5-volt level on the signal. Likewise, minimum and maximum output valid times are measured from the 1.5-volt level on the rising edge of CLK[0] at the controller to the 1.5-volt level on the signal at the controller.

Table 38. Timing Reference Voltages

Symbol	5 V PCI Signaling Requirement	3.3-V PCI Signaling Requirement			Controller Universal PCI Timing Reference Values
		PCI Specification	Vcc = 3.0	Vcc = 3.6	
Vth	2.4 V	0.6 Vcc	1.8 V	2.16 V	2.4 V
Vtl	0.4 V	0.2 Vcc	0.6 V	0.72 V	0.4 V
Vtest	1.5 V	0.4 Vcc	1.2 V	1.44 V	1.5 V
Vstep1	N/A	0.285 Vcc	.86 V	1.03 V	0.285 Vcc
Vstep2	N/A	0.615 Vcc	1.85 V	2.15 V	0.615 Vcc
Vmax	2.0 V	0.4 Vcc	1.2 V	1.4 V	2.0 V

Table 39. PCI Signal Timing Design Budget[1]

Symbol	Parameter	Minimum	Maximum	Units
Tval	CLK[0] to signal valid delay: all but REQ[3:0]# and GNT[3:0]#	3.4[2]	14.4	ns
Tval(ptp)	CLK[0] to signal valid delay: REQ[3:0]# and GNT[3:0]#	3.4[2]	14.4	ns
Ton	Float to active delay	3.4[2]		ns
Toff	Active to float delay		31.4	ns
Tsu	Input setup to CLK[0]: all but REQ[3:0]# and GNT[3:0]#	6.6		ns
Tsu(ptp1)	Input setup to CLK[0]: GNT[3:0]#	6.6		ns
Th	Input hold time from CLK[0]	3.4		ns
Trst-off	Reset active to output float delay (all output drivers)		43.4	ns
Trst	Reset active time after power stable	1		µs
Trst-clk	Reset active time after CLK stable	100		µs
Ttst	Test active to test output state (all output drivers)		43.4	ns

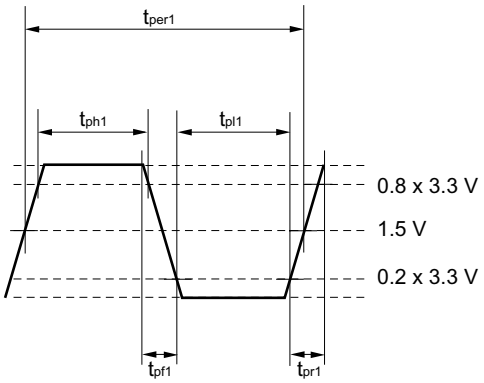
Notes: **1.** The PCI timing here is slightly different from the PCI specification because the controller is the source of the PCI clock. A controller ASIC meeting this timing budget will be PCI compliant in a system that uses the PCI clocking scheme described in Section 10.0 on page 91. All timing is measured with respect to the CLK[0] output signal at the controller.

2. This minimum rating is set at an extra 1 ns to provide 1-ns hold time to all PCI devices.

14.4.3
CPU Interface

All CPU interface signals are measured at the controller with respect to the 1.5-volt level of the rising edge of the MasterClock output signal at the controller. Table 40 shows the CPU timing budget for the controller's CPU interface. The controller drives the MasterClock output to valid CPU MasterClock input levels, as shown in Figure 64. For other CPU interface signals, the controller meets standard TTL input and output level requirements.

Figure 64. MasterClock Input Waveform at CPU



4373-003.eps

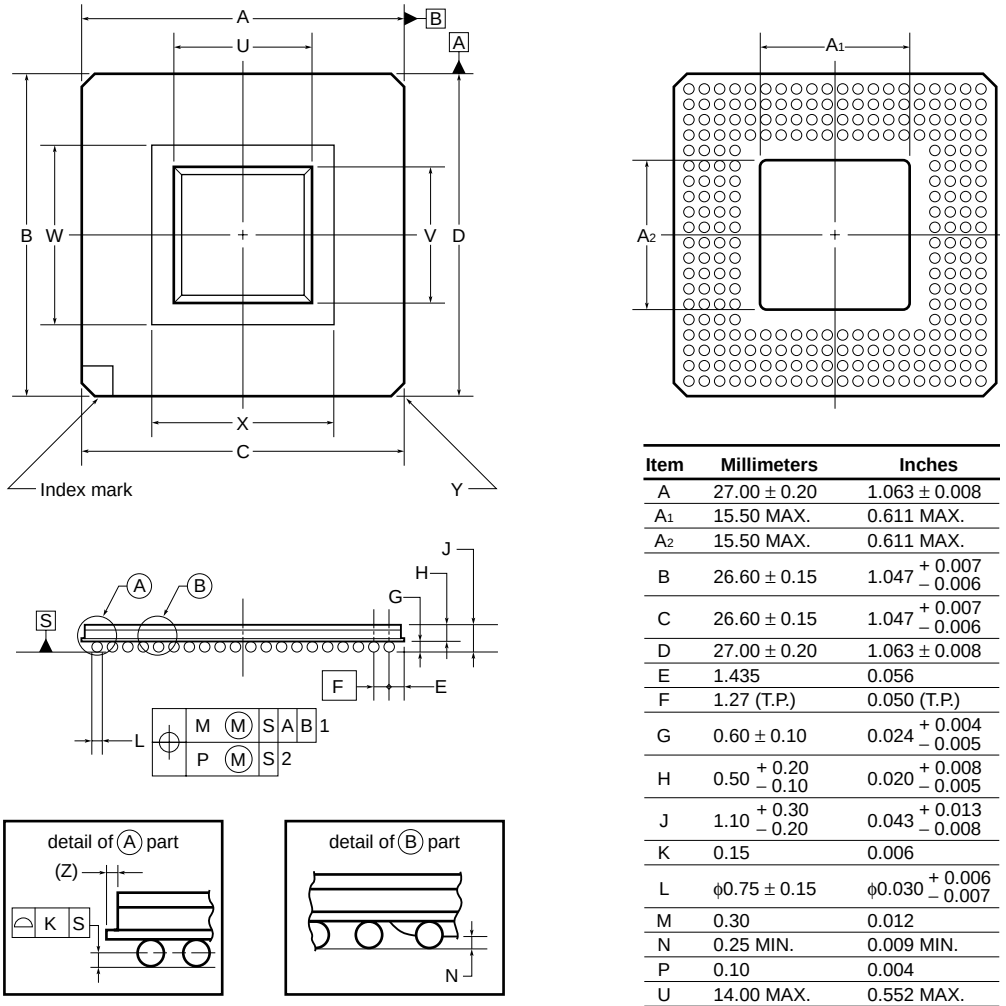
Table 40. CPU Interface Signal Timing

Symbol	Parameter	Minimum	Maximum	Units
Tper1	MasterClock period	15	50	ns
Tph1	MasterClock high time	4		ns
Tpl1	MasterClock low time	4		ns
Tpr1	MasterClock rise time	1	3	ns
Tpf1	MasterClock fall time	1	3	ns
Tov0	MasterClock at the controller to output valid at the controller	2.2	10.5	ns
Trst-off	Reset active to output float delay (all output drivers)		43.4	ns

15.0 Package Drawing

The VRC4375 system controller uses a 256-pin tape ball grid array (TBGA) package, as shown in Figure 65.

Figure 65. Package Drawing



Item	Millimeters	Inches
A	27.00 ± 0.20	1.063 ± 0.008
A_1	15.50 MAX.	0.611 MAX.
A_2	15.50 MAX.	0.611 MAX.
B	26.60 ± 0.15	$1.047^{+0.007}_{-0.006}$
C	26.60 ± 0.15	$1.047^{+0.007}_{-0.006}$
D	27.00 ± 0.20	1.063 ± 0.008
E	1.435	0.056
F	1.27 (T.P.)	0.050 (T.P.)
G	0.60 ± 0.10	$0.024^{+0.004}_{-0.005}$
H	$0.50^{+0.20}_{-0.10}$	$0.020^{+0.008}_{-0.005}$
J	$1.10^{+0.30}_{-0.20}$	$0.043^{+0.013}_{-0.008}$
K	0.15	0.006
L	ϕ0.75 ± 0.15	$\phi 0.030^{+0.006}_{-0.007}$
M	0.30	0.012
N	0.25 MIN.	0.009 MIN.
P	0.10	0.004
U	14.00 MAX.	0.552 MAX.
V	14.00 MAX.	0.552 MAX.
W	15.11 ± 0.15	0.595 ± 0.006
X	15.11 ± 0.15	0.595 ± 0.006
Y	C 0.40	C 0.016
Z	0.20	0.008

NOTES:

Each ball centerline is located within ϕ0.30 mm (ϕ0.012 inch) of its true position (T.P.) at maximum material condition.

! Each ball centerline is located within ϕ0.10 mm (ϕ0.004 inch) of its true position (T.P.) at maximum material condition.

P256N2

NEC

Some of the information contained in this document may vary from country to country. Before using any NEC product in your application, please contact a representative from the NEC office in your country to obtain a list of authorized representatives and distributors who can verify the following:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara
Tel: 800-366-9782
Fax: 800-729-9288

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC Electronics Italiana s.r.l.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, the Netherlands
Tel: 040-2445845
Fax: 040-2444580

For literature, call **1-800-366-9782** 7 a.m. to 6 p.m. Pacific time
or FAX your request to **1-800-729-9288**
or visit our web site at **www.necel.com**

NEC

U13749EU1V0DS00