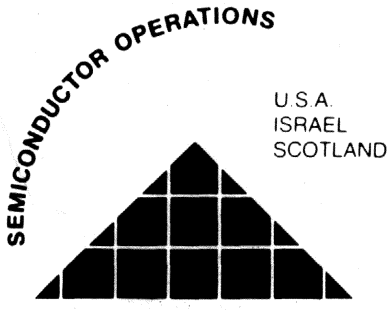
digital

1987

Semiconductor Databook

Volume 2



Semiconductor Databook

Volume 2

Commercial and Engineering

▪ Foreword

We at Semiconductor Operations (SCO) are committed to provide excellence in integrated circuit technologies, products, and services to support our customers, the Digital Systems Groups.

Our primary objective is to optimize Digital's competitive market position by developing leadership system performance at the lowest possible cost and within the appropriate time constraints.

The execution of programs designed to achieve this objective has resulted in the technologies and products described in the *1987 Semiconductor Databook Volumes 1 and 2*. While the basic charter of SCO is to provide strategic and tactical management of all integrated circuit requirements, the VLSI design and manufacturing function of SCO has become the focal point for unique and complex circuits that have contributed significantly to the success of many new Digital products. A strategic investment has been made in CMOS technology and in the design tools necessary to take advantage of this technology. Increased circuit densities and performance have resulted, and capabilities have been extended from full-custom design for maximum performance to semi-custom design for fast time-to-market application. CAD tools are continually being developed to further enhance design and design methodology.

SCO is continually expanding its facilities to provide you with better service. While Hudson and Andover, Massachusetts are the nucleus of the engineering and manufacturing operations, supplemental design facilities are available in Israel and Japan and additional manufacturing capacity is being planned in Scotland. In addition, a new 6-inch wafer pilot fabrication line has been approved for construction in Hudson to aid in the state-of-the-art development of the advanced CMOS devices.

During the past year, many new integrated circuits have been developed and released. Although some are application-specific, the circuits that are suitable for general use are described in Volumes 1 and 2 of this databook. Volume 1 is a revision to the 1986 Databook and includes the latest revisions and changes. Volume 2 contains information related to the new CMOS products that have been recently developed for general use. We encourage you to become familiar with these products and to use them in the design of Digital's systems products when possible. We are ready to assist you in your design process and in support of your production needs.

Our ultimate goal is to ensure that Digital's systems continue to maintain significant competitive advantage through the use of SCO services and products.

• Part Identification Codes

The following identification codes are used with the devices in this databook.

780 Series

78xyz - xx

- x:
  - 0 = Processors
  - 1 = Coprocessor
  - 2 = Memories
  - 3 = I/O devices
  - 4 = Reserved
  - 5 = Controllers
  - 6 = Graphic devices
  - 7 = Bus interfaces
  - 8 = Communications devices
  - 9 = Reserved
- xx:
  - GA = Gullwing
  - FA = Straight
  - PA = Pin grid array

DC Series

DCxyz

- x:
  - 0 = Custom bipolar devices
  - 1 = Custom bipolar devices
  - 3 = MOS devices
  - 5 = MOS devices

• Cross-referencing of Semiconductor Products

Part Name	Part Number	Purchase Number	Description
DC341	78034-GA	21-24674-01	CVAX Central Processing Unit (CVAX CPU)
DC513	78134-GA	21-26604-01	CVAX Floating-point Accelerator (CFPA)
DC509	78135-GA	21-24673-01	CVAX Clock Generator (CCLOCK)
DC551	78332-GA	21-24942-01	MicroVAX System Support Chip (SSC)
DC357	78588-PA	21-25091-01	CVAX Memory Controller (CMCTL)
DC527	78711-GA	21-25972-01	CVAX Q22-bus Interface Chip (CQBIC)
DC514	—	21-24674-01	CMOS VAXBI Bus Interface Chip (CBIC)

Contents

Section 1 • Microprocessor and Support Devices

CVAX 78034 32-bit Central Processing Unit 1–1
CVAX 78134 Floating-point Accelerator 1–91
CVAX 78135 Clock Generator 1–109
MicroVAX 78332 System Support Chip 1–123

Section 2 • Bus Support Devices

CVAX 78588 Memory Controller 2–1
CVAX 78711 Q22-bus Interface Chip 2–55
DC514 CMOS VAXBI Bus Interface Chip 2–107

Appendix • Mechanical Specifications

Contents

Session 1 · Microprocessor and Support Devices

Session 2 · The Support Devices

Appendix · Data Sheet Locations

Section 1—Microprocessor and Support Devices

The CVAX 78034 microprocessor and support devices are the latest development in CMOS devices. They provide the increased performance and the versatility required for the design of new and faster VAX systems.

CVAX 78034 Central Processing Unit—The CVAX 78034 CPU is a low-cost high-performance, 32-bit virtual memory microprocessor. It is implemented in double-metal CMOS and is functionally compatible with the MicroVAX 78032 CPU. It contains a 1-Kbyte cache memory and provides pipeline architectures and instruction prefetch.

CVAX 78134 Floating-point Accelerator—The CVAX 78134 CFPA is a high-performance coprocessor used with the CVAX 78034 CPU to accelerate the execution of floating-point instructions. It eliminates the need to emulate floating-point operations in software.

CVAX 78135 Clock Generator—The CVAX 78135 CCLOCK generates the precision MOS clock signals required by the CVAX 78034 CPU, CVAX 78134 CFPA, and up to two additional support chips.

MicroVAX 78332 System Support Chip—The MicroVAX 78332 SSC is a multifunction device that provides the common functions necessary to support the MicroVAX 78032 and CVAX 78034 CPU. It includes support logic for an external ROM, two asynchronous serial-line ports, programmable address decoders, programmable timers, and a realtime clock.

Features

- High performance
  - —32-bit internal and external data path
  - —1 Kbyte on-chip instruction/data cache
  - —Pipelined architecture
  - —Instruction prefetch
- Optimized floating-point accelerator interface
- VAX instruction set
  - —304 instructions (59 emulated)
  - —21 address modes
  - —14 data types
- Sixteen 32-bit general purpose registers
- 22 interrupt levels
  - —15 software
  - —7 hardware
- Vectored software and hardware interrupts
- VAX memory management
  - —Full memory protection
  - —Four privilege modes
  - —Process and system space mapped
- 4 gigabyte virtual address space
- 1 gigabyte physical address space
  - —512 megabyte memory space
  - —512 megabyte I/O space
- Data parity checking
- Industry compatible external interface
- Single 5-volt power supply
- 84-pin surfacemount package

Description

The CVAX 78034 Central Processing Unit (CVAX CPU) is a 32-bit, virtual memory microprocessor. Implemented in a double-metal CMOS process, the CVAX CPU is a low-cost, high-performance microprocessor for single-board computers, single-user workstations, low-end systems, and other applications such as multiprocessing configurations. The CVAX CPU is functionally compatible with the MicroVAX 78032 CPU and offers the system designer software compatibility, faster bus cycle times, a 1-KByte on-chip cache, and an optimized interface for the CVAX 78134 Floating-point Accelerator (CFPA). Figure 1 is a block diagram of the CVAX 78034 CPU.

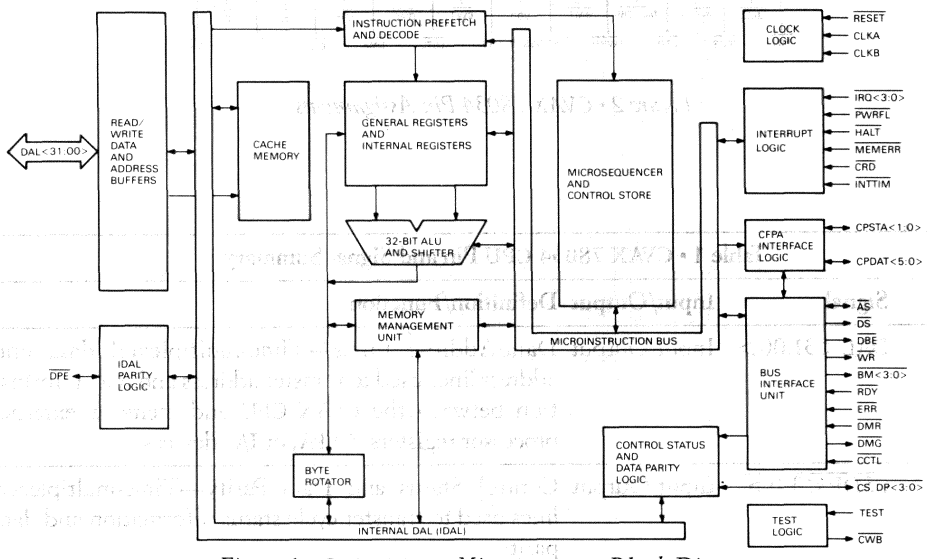


Figure 1 • CVAX 78034 Microprocessor Block Diagram

• Pin and Signal Descriptions

This section provides a description of the input and output signals and power and ground connections used by the CVAX 78034 CPU. The signal pin assignments are identified in Figure 2 and summarized in Table 1.

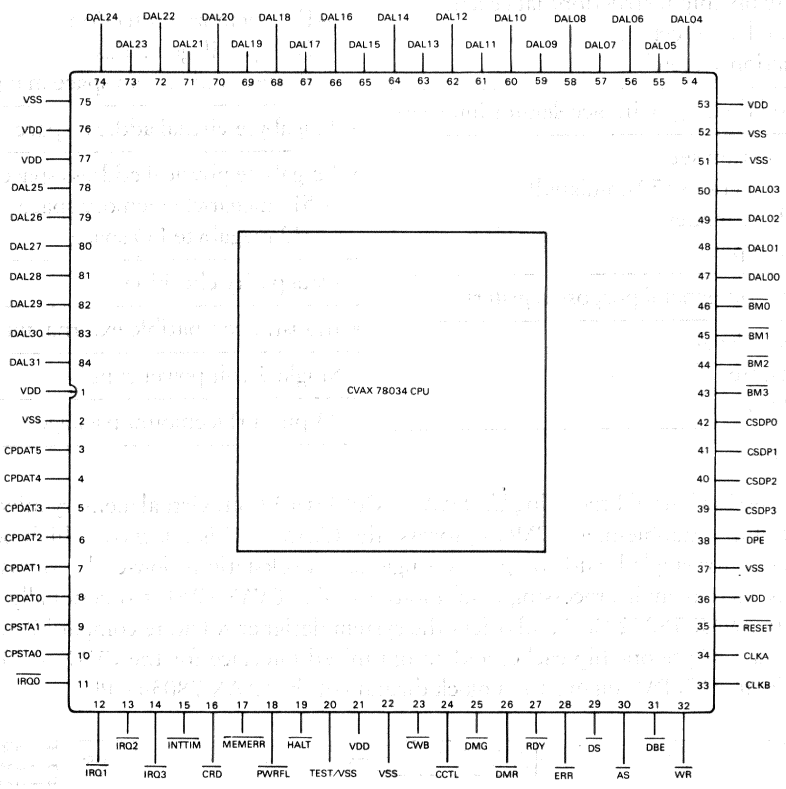


Figure 2 • CVAX 78034 Pin Assignments

Table 1 • CVAX 78034 CPU Pin and Signal Summary

Pin	Signal	Input/Output	Definition/Function
84-78, 74-54, 50-47	DAL<31:00>	Input/Output	Data/Address Lines—Time-multiplexed data and address lines used to transfer address and data information between the CVAX CPU and memory, external processor registers, CFPA, or I/O devices.
39-42	$\overline{\text{CSDP}}$<3:0>	Input/Output	Control Status and Data Parity—Time-multiplexed lines used to transfer cycle status information and data parity.

Pin	Signal	Input/Output	Definition/Function
38	$\overline{DPE}$	Input/Output	Data Parity Enable—A signal used to enable parity checking and to indicate valid parity information on $\overline{CSDP<3:0>}$.
30	$\overline{AS}$	Input/Output	Address Strobe—A strobe that indicates the initial information on DAL<31:00> and $\overline{CSDP<3:0>}$ is valid. The leading edge can be used to latch the information on the lines.
29	$\overline{DS}$	Output	Data Strobe—A strobe that indicates to the system interface that DAL<31:00> and $\overline{CSDP<3:0>}$ are ready to receive data during a CPU read, external processor register read, or interrupt acknowledge cycle. It is deasserted to indicate that the data has been received. It also contains valid outgoing data during a CPU write cycle or external processor register write cycle and is deasserted when the data is ready to be removed.
43-46	$\overline{BM<3:0>}$	Output	Byte Masks—Specify which bytes of DAL data and associated parity bits are valid during the second part of an I/O cycle.
32	$\overline{WR}$	Output	Write—Specifies the direction of data transfer on the DAL.
31	$\overline{DBE}$	Output	Data Buffer Enable—Used with $\overline{WR}$ to control external DAL transceivers.
27	$\overline{RDY}$	Input	Ready—Asserted by external logic to indicate the normal termination of the current bus cycle. $\overline{RDY}$ and $\overline{ERR}$ can be asserted together to request a retry of the bus cycle.
28	$\overline{ERR}$	Input	Error—Asserted by external logic to indicate the abnormal termination of the current bus cycle. $\overline{ERR}$ and $\overline{RDY}$ can be asserted together to request a retry of the bus cycle.
35	$\overline{RESET}$	Input	Reset—Asserted by external logic to initialize the CVAX CPU to a predetermined initial state.
19	$\overline{HALT}$	Input	Halt—A nonmaskable interrupt used to transfer control to console macrocode.
14-11	$\overline{IRQ<3:0>}$	Input	Interrupt Request—Four maskable interrupt request lines for device interrupts.
18	$\overline{PWRFL}$	Input	Powerfail—A maskable interrupt used to indicate a powerfail condition.

Pin	Signal	Input/Output	Definition/Function
16	$\overline{\text{CRD}}$	Input	Corrected Read Data—A maskable interrupt used to signal an ECC correctable read error in a memory subsystem.
15	$\overline{\text{INTTIM}}$	Input	Interval Timer—A maskable interrupt used to provide system timing information from the interval timer.
17	$\overline{\text{MEMERR}}$	Input	Memory Error—A maskable interrupt used to indicate a memory error.
26	$\overline{\text{DMR}}$	Input	DMA Request—Asserted by external logic to request a DMA cycle.
25	$\overline{\text{DMG}}$	Output	DMA Grant—Asserted by the CVAX CPU to acknowledge a DMA request.
24	$\overline{\text{CCTL}}$	Input	Cache Control—Used by external logic to control the operation of the internal cache memory during DMA and CPU read cycles.
3-8	CPDAT<5:0>	Input/Output	Coprocessor Data—Used to transfer opcode, control information, condition code, and exception status between the CVAX CPU and the CFPA.
9,10	CPSTA<1:0>	Input/Output	Coprocessor Status—Used to transfer status information between the CVAX CPU and the CFPA.
1,21,36, 53,76,77	V_{DD}	Input	Voltage—5-volt power supply.
2,22,37, 51,52,75	V_{SS}	Input	Ground—Ground reference.
34,33	CLKA, CLKB	Input	Clock A and Clock B—Supply the basic clock timing to the CVAX CPU. CLKA and CLKB are phase shifted by 180 degrees. These inputs are nominal 20-MHz, MOS level, square-wave signals.
23	$\overline{\text{CWB}}$	Output	Clear Write Buffer—Used to indicate that conditions internal to the CPU require the external write buffer (if included) to be cleared. This signal provides test information when the TEST input is asserted and its test output is reserved for chip manufacturing test.
20	TEST/V_{SS}	Input	Test/V_{SS}—Reserved for chip manufacturing test. TEST must be connected to V_{SS} when not in test mode.

Data/Address

Data/Address Lines (DAL<31:00>)—These are bidirectional time-multiplexed lines used to transfer address, data, and interrupt information. The information on DAL<31:00> depends on the type of bus cycle being executed. During the first part of a CPU read or CPU write cycle, DAL<31:00> specify the length of the memory operand and DAL<29:02> contain the longword address of the memory operand. DAL29 is used to distinguish a memory space address

from an I/O space address. DAL<01:00> are reserved. $\overline{BM}$<3:0> determine which byte(s) of the longword address are to be used. Refer to the *Memory Access Protocol* section for additional information. The DAL information is defined in Table 2.

Table 2 • CVAX 78034 Data and Address Line Information

DAL 31	30	Operand length	DAL 29	Type	DAL <28:02>	DAL<01:00>
0	0	DMA hexword	0	Memory	longword address	reserved
0	1	longword	1	I/O		
1	0	quadword				
1	1	DMA octaword				

During the first part of an interrupt acknowledge cycle, DAL<06:02> transfer the Interrupt Priority Level (IPL) (hexadecimal) of the interrupt being acknowledged. During the first part of an external processor register read or write bus cycle, DAL<07:02> contain the Internal Processor Register (IPR) number that is being accessed. During the second part of a CPU read cycle, external processor register read, or interrupt acknowledge cycle, DAL<31:00> receive incoming information. During the second part of a CPU write or external processor register write cycle, DAL<31:00> are used to transmit the data to be written.

Cycle Status and Data Parity ($\overline{CSDP}$<3:0>)—These time-multiplexed lines transfer cycle status and data parity information between the CPU and external devices. During the first part of a bus cycle, $\overline{CSDP}$<2:0> and $\overline{WR}$ provide status information of the current bus cycle as listed in Table 3. $\overline{CSDP3}$ indicates the set in the internal cache memory that is being allocated during a cachable read operation and is undefined during all other bus cycles. $\overline{CSDP3}$ is asserted to specify set 1 and negated to specify set 2.

Table 3 • CVAX 78034 Bus Cycle Status*

$\overline{WR}$	$\overline{CSDP}$ 2	1	0	Bus cycle type
H	L	L	L	request D-stream read
H	L	L	H	reserved
H	L	H	L	external IPR read
H	L	H	H	interrupt acknowledge
H	H	L	L	request I-stream read
H	H	L	H	demand D-stream read (lock)
H	H	H	L	demand D-stream read modify intent
H	H	H	H	demand D-stream read (no lock or modify intent)
L	L	L	L	reserved
L	L	L	H	reserved
L	L	H	L	external IPR write

$\overline{WR}$	$\overline{CSDP}$ 2	1	0	Bus cycle type
L	L	H	H	reserved for use by DMA devices
L	H	L	L	reserved
L	H	L	H	write unlock
L	H	H	L	reserved
L	H	H	H	write no unlock

*H = high level, L = low level

During the second part of a bus cycle, $\overline{CSDP<3:0>}$ transfer byte parity information for DAL line data during a CPU read, CPU write, or external processor write cycle. Parity checking is not performed during external processor register read cycles or during transfers between the CPU and the optional floating-point accelerator. Even parity is checked or generated on even bytes, and odd parity is checked or generated on odd bytes. During a CPU read cycle, the CPU reads and checks the data parity on the bytes specified by the $\overline{BM<3:0>}$ information. During a CPU write or external processor register write transaction, the CPU generates data parity for all bytes regardless of the state of $\overline{BM<3:0>}$. The $\overline{DPE}$ signal specifies when the CPU is to check or generate parity. It must not be asserted during external processor register read operations or during transfers between the CPU and the optional floating-point accelerator.

Data Parity Enable ($\overline{DPE}$)—This bidirectional signal controls the checking or generation of data parity. During a CPU read cycle or interrupt acknowledge cycle, $\overline{DPE}$ is asserted by external logic with the DAL data to enable parity checking by the CPU of the incoming data. During a CPU write cycle or external processor register write cycle, the CPU asserts $\overline{DPE}$ to indicate that valid parity information is on $\overline{CSDP<3:0>}$. $\overline{DPE}$ must not be asserted during external processor register read transfers or transfers between the CPU and the optional floating-point accelerator. $\overline{DPE}$ requires an external pullup resistor and must be asserted by an external interface that requires the CPU to check parity.

Bus Control

Address Strobe ($\overline{AS}$)—This bidirectional signal indicates that valid address information is on the DAL lines. The CPU asserts $\overline{AS}$ to indicate that the address and control information on DAL<31:00> and $\overline{CSDP<3:0>}$ is valid. During a DMA transfer, the CPU uses the assertion of $\overline{AS}$ by the DMA device to latch the DMA address. The CPU uses this address during a DMA cache invalidate cycle.

Data Strobe ($\overline{DS}$)—This signal provides timing information for data transfers. During a CPU read, external processor register read, or interrupt acknowledge bus cycle, the CPU asserts $\overline{DS}$ to indicate that DAL<31:00> and $\overline{CSDP<3:0>}$ are available to receive incoming data and deasserts $\overline{DS}$ to indicate that the incoming data has been latched into the CPU. During a CPU write or external processor register write cycle, the CPU asserts $\overline{DS}$ to indicate that DAL<31:00> and $\overline{CSDP<3:0>}$ contain valid outgoing data. It deasserts $\overline{DS}$ to indicate that the data is about to be removed.

Byte Masks ($\overline{BM<3:0>}$)—These signals indicate which bytes of the DAL lines contain valid data. During a CPU read cycle, they indicate the bytes of data and associated parity bits that are to be transferred onto the DAL and $\overline{CSDP}$ lines. During a CPU write cycle, they indicate the bytes of the DAL lines and $\overline{CSDP}$ lines that contain valid data and parity information. The $\overline{BM<3:0>}$ line information is qualified by the assertion $\overline{AS}$.

Write (WR)—This signal indicates the direction of data transfer on the DAL bus for the current bus cycle. When asserted during a CPU bus cycle, the CPU will transfer data onto the DAL lines. When deasserted during a CPU bus cycle, the CPU will read data from the DAL lines. WR can be used to control the direction of the external DAL transceivers inputs. WR is qualified by the assertion of AS.

Data Buffer Enable (DBE)—This signal and WR is used to control external DAL transceivers. The CPU asserts DBE to enable the DAL transceivers and deasserts DBE to disable the DAL transceivers. DBE is qualified by the assertion of AS.

Ready (RDY)—This signal is asserted by external logic to indicate the normal termination of the current bus cycle. It may also be asserted with the error signal (ERR) to request a retry of the current bus cycle.

Error (ERR)—This signal is asserted by external logic to indicate the abnormal termination of the current bus cycle. It may also be asserted with RDY to request a retry of the current bus cycle.

System Control

Reset (RESET)—This signal is asserted by external logic to force the CPU to its initial powerup state.

Halt (HALT)—This signal is asserted to generate a nonmaskable interrupt that transfers control of the CPU to the console macrocode. At the end of the current instruction, the CVAX CPU enters the restart process with a restart code equal to 2 and the HALT signal is asserted. HALT is edge-sensitive, sampled every microcycle, and internally synchronized.

Interrupt Control

Interrupt Request (IRQ<3:0>)—These signals allow external logic to transfer interrupt requests to the CPU. The CPU responds to the assertion of one or more of these signals by executing an interrupt acknowledge bus cycle for the highest pending Interrupt Priority Level (IPL). The IPL associated with each line is listed in Table 4. The IRQ<3:0> signals are level-sensitive and are sampled every microcycle.

Table 4 • CVAX 78034 Interrupt Request Line Assignments

Line	Interrupt Priority Level (hexadecimal)
IRQ3	IPL 17
IRQ2	IPL 16
IRQ1	IPL 15
IRQ2	IPL 14

Powerfail (PWRFL)—This signal allows external logic to notify the CVAX CPU of a power failure. When asserted, it results in the generation of an interrupt at IPL 1E (hexadecimal). The CPU responds to the interrupt by accessing System Control Block (SCB) vector 0C (hexadecimal). The CPU does not execute an interrupt acknowledge bus cycle when responding to this interrupt. This signal is edge-sensitive, sampled every microcycle, and internally synchronized by the CPU.

Corrected Read Data (CRD)—This signal allows external logic to notify the CPU of an ECC error in memory. Asserting this signal results in the generation of an interrupt at IPL 1A (hexadecimal). The CPU responds to this interrupt by accessing SCB vector 54 (hexadecimal). The CPU does not

execute an interrupt acknowledge bus cycle when responding to this interrupt. This signal is edge-sensitive, sampled every microcycle, and internally synchronized by the CPU.

Interval Timer ($\overline{\text{INTTIM}}$)—This signal allows external logic to signal an interval timer rollover to the CPU. The assertion of this signal results in the generation of an interrupt at IPL 16 (hexadecimal). The CPU responds to this interrupt by accessing SCB vector C0 (hexadecimal). The CPU does not execute an interrupt acknowledge bus cycle when responding to this interrupt. It is edge-sensitive, sampled every microcycle, and internally synchronized by the CPU.

Memory Error ($\overline{\text{MEMERR}}$)—This signal allows external logic to indicate to the CPU that a memory error has been detected. The assertion of this signal results in the generation of an interrupt at IPL 1D (hexadecimal). The CPU responds to this interrupt by accessing SCB vector 60 (hexadecimal) and does not execute an interrupt acknowledge bus cycle. $\overline{\text{MEMERR}}$ provides support for the implementation of a memory subsystem with multiple write buffers or delayed write transfers. When the CPU writes to this type of memory subsystem, the address and data are latched and the $\overline{\text{RDY}}$ signal is asserted. If an error occurs it is reported to the CPU when $\overline{\text{MEMERR}}$ is asserted. It is edge-sensitive, sampled every microcycle, and internally synchronized by the CPU.

Direct Memory Access Control

DMA Request ($\overline{\text{DMR}}$)—This signal allows external logic to request use of the DAL and related control signals for a DMA transfers or for other purposes. It is a level-sensitive signal, sampled every microcycle, and internally synchronized by the CPU.

DMA Grant ($\overline{\text{DMG}}$)—This signal is asserted by the CPU to grant control of the DAL lines and related control signals to external logic. The CPU sets the DAL<31:00> $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{BM}}$<3:0>, $\overline{\text{DBE}}$, $\overline{\text{DPE}}$, $\overline{\text{CSDP}}$<3:0>, and $\overline{\text{WR}}$ lines to a high-impedance state. When external logic deasserts $\overline{\text{DMR}}$, the CPU responds by deasserting $\overline{\text{DMG}}$ and by starting the next bus cycle.

Cache Control

Cache Control ($\overline{\text{CCTL}}$)—The function of this signal depends on the type of bus cycle.

During a DMA cycle, the assertion of this signal by external logic initiates a conditional cache invalidate cycle. $\overline{\text{CCTL}}$ is edge-sensitive, sampled every microcycle, and internally synchronized by the CPU.

During a CPU read cycle, this signal is asserted to prevent the accessed data from being stored in the internal cache memory of the CPU. $\overline{\text{CCTL}}$ is level-sensitive and must be asserted synchronously with the timing sampling point for the CPU read cycle.

Floating-point Accelerator Control

FPA Data (CPDAT<5:0>)—These bidirectional lines transfer opcodes, control information, condition codes, and exception status information between the CVAX CPU and the CVAX FPA.

FPA Status (CPSTA<1:0>)—These bidirectional lines indicate the interpretation of the CPDAT <5:0> line information to the CVAX CPU or CVAX FPA.

Power Supply

Voltage (V_{DD})—5-volt power supply

Ground (V_{SS})—Ground reference

Clock Timing

Clock A and Clock B (CLKA and CLKB)—These inputs supply the basic clock timing to the CVAX CPU. The inputs are nominally 20 MHz and are MOS-level square-wave signals. CLKA is phase shifted from CBKB by 180 degrees.

Miscellaneous

Clear Write Buffer ($\overline{\text{CWB}}$)—This signal is asserted by the CPU to indicate that internal conditions of the CPU require clearing of the external write buffer (if included). This signal provides test information when the TEST input is asserted. It is reserved for manufacturing test. The CPU asserts $\overline{\text{CWB}}$

- At the start of an instruction or sequence that can change the processor state. These are CHMx, REI, start of an interrupt, exception or abort (including machine check, BPT, etc.), or entry to the console (including $\overline{\text{HALT}}$).
- As a part of an instruction or sequence that can change context such as end of LDPCTX or end of SVPCTX.
- As a part of an instruction or sequence involved in error recovery such as a write to the MAPEN, CADR, or MSER registers.

Test (TEST)—Reserved for manufacturing test. This input provides the ground for the $\overline{\text{AS}}$ logic and must be connected to V_{SS} during normal operation.

Architecture Summary

The programming model for the CVAX 78034 architecture is shown in Figure 3. It is grouped into application programming (user) area and system programming area.

The sixteen general registers and Processor Status Word (PSW) are user accessible. The system registers are privileged registers that are used by the operating system. These registers are used for context switching, memory management, cache memory control, reporting of memory subsystem status, exception and interrupt handling, and processor control.

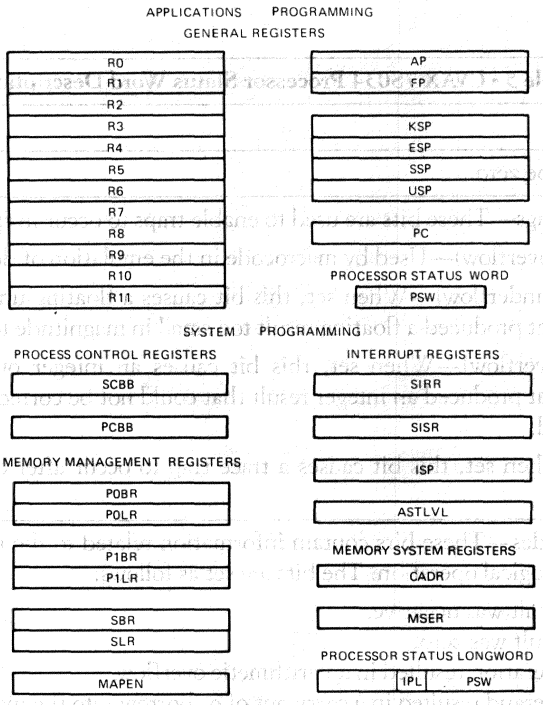


Figure 3 • CVAX 78034 Programming Model

General Registers

The CVAX 78034 has sixteen 32-bit general registers that can be used for temporary storage as accumulators, base registers, and index registers. The registers used for specific functions are the Stack Pointer (SP), Argument Pointer (AP), Frame Pointer (FP), and Program Counter (PC).

Stack Pointer (SP)—Five SP registers are included, one for each operating mode of the processor and one for use by the system when handling interrupts. The SP contains the address of the processor defined stack. The stack pointer(s) used is determined by the operating mode of the processor.

Argument Pointer (AP)—The VAX procedure call convention uses a data structure called an argument list. The AP register contains the address of the base of this structure.

Frame Pointer (FP)—The VAX procedure call convention builds a data structure on the stack called a stack frame. The FP register contains the address of the base of this structure.

Program Counter (PC)—The PC register contains the address of the next byte of the program and is not used as an accumulator, index, or temporary register.

Processor Status Word (PSW)—The PSW contains the condition codes and trap enable flags for the CVAX 78034 CPU. The PSW is the user accessible portion of the processor status longword. The lower 16 bits of the PSL contain the PSW. The format of the PSW is shown in Figure 4 and described in Table 5.

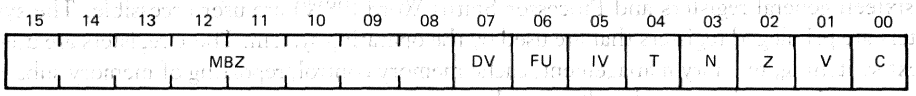


Figure 4 • CVAX 78034 Processor Status Word Format

Table 5 • CVAX 78034 Processor Status Word Description

Bit	Description
15:08	MBZ—Must be zero.
07:04	Trap enable flags—These bits are used to enable traps to occur in special circumstances. DV (Decimal overflow)—Used by macrocode in the emulation of decimal instructions. FU (Floating underflow)—When set, this bit causes a floating underflow trap after an instruction that produced a floating result too small in magnitude to be represented. IV (Integer overflow)—When set, this bit causes an integer overflow trap after an instruction that produced an integer result that could not be correctly represented in the space provided. T (Trace)—When set, this bit causes a trace trap to occur after execution of the next instruction.
03:00	Condition Codes—These bits contain information related to the result of the last CPU arithmetic or logical operation. The bits are set as follows: N = 1 if the result was negative. Z = 1 if the result was zero. V = 1 if the operation resulted in an arithmetic overflow. C = 1 if the operand resulted in a carry out of or borrow into the most significant bit.

Process Control Registers

The process control registers are used by the system to access the system control block and the process control block.

System Control Block Base register (SCBB)—The SCBB register contains the base address of the System Control Block (SCB). The SCB contains the vectors used for servicing interrupts and exceptions.

Process Control Block Base register (PCBB)—The PCBB contains the base address of the Process Control Block (PCB). The PCB contains the hardware context of the current process.

Memory Management Registers

These registers are used by the system to enable the internal memory management unit of the CVAX CPU and to access the page-table entries in memory used to translate virtual addresses into physical addresses. The function of each of these registers is described in the *Memory Management* section.

Interrupt Registers

These registers are used to control the interrupt system of the processor. They monitor interrupt requests, current interrupt priority level, and the interrupt stack pointer. The function of each of these registers is described in the *Exceptions and Interrupt* section.

Memory System Registers

These registers are used to control the operation of the internal cache memory and to report status and errors for both the cache memory and the external memory subsystem.

Cache Disable Register (CADR)—The CADR controls the internal cache memory. This register enables and disables cache memory operation, selects the set (Set 1 and Set 2) to be used, and selects the type of reference(s) to be stored. The format of the CADR register is shown in Figure 5 and described in Table 6.

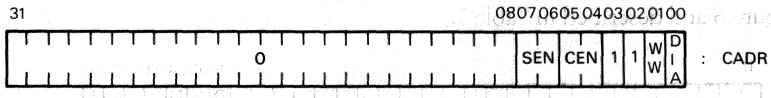


Figure 5 • CVAX 78034 Cache Disable Register Format

Table 6 • CVAX 78034 Cache Disable Register Description

Bit	Description
31:08	Read as zeros
07:06	SEN (Set enable)—These bits are read/write and are used to enable Set 1 and Set 2 sections of cache memory.

Bit 07	Bit 06	Set 2	Set 1
0	0	disabled	disabled
0	1	disabled	enabled
1	0	enabled	disabled
1	1	enabled	enabled

Bit	Description
05:04	CEN (Cache enable)—These read/write bits are used to enable cache and to select the type of references to be stored in cache.

Bit 05	Bit 04	Result
0	0	cache disabled
0	1	D-stream only (for diagnostic use)
1	0	I-stream only
1	1	I-stream and D-stream

03:02	Read as ones.
01	WW (Write wrong parity)—This bit is set to cause wrong parity to be stored when the cache is written.
00	DIA (Diagnostic)—This bit is set to select diagnostic mode for cache memory. The cache cannot be cleared when this bit is set.

When CADR bits 5:4 select I-stream only (10) to be stored in cache, the CVAX CPU automatically clears the cache when an REI instruction is executed. The REI instruction must be executed prior to running code from an updated page of memory as defined by the VAX System Reference Manual (VAX SRM). Therefore, systems that adhere to the VAX SRM are not required to monitor DMA write operations in order to prevent stale data from accumulating in the cache. When CADR bits 5:4 select D-stream only (01) or I-stream and D-stream (11), invalidate-on-hit cycles must be used to remove stale data from the cache.

Diagnostic mode should be selected only when one set (Set 1 or Set 2) is enabled. The diagnostic mode prevents clearing of the cache when the CADR is written.

Memory System Error Register (MSER)—The MSER contains status and error information for the internal cache memory and the external memory subsystem. The format of the MSER register is shown in Figure 6 and described in Table 7.

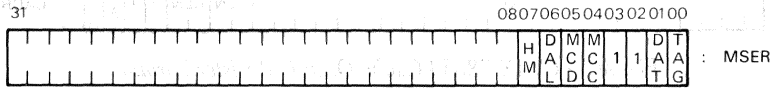


Figure 6 • CVAX 78034 Memory System Error Register Format

Table 7 • CVAX 78034 Memory System Error Register Description

Bit	Description
31:08	Read as zeros.
07	HM (Hit/Miss)—Set for a cache miss and cleared for a cache hit.
06	DAL (DAL parity)—Set when a parity error is detected on the DAL during a demand or request read operation. It is cleared by writing to the MSER.
05	MCD (Machine check on DAL parity error)—Set when a DAL parity error causes a machine check. It is cleared by writing to the MSER.

Bit	Description
04	MCC (Machine check on cache parity error)—Set when a cache parity error (tag or data) causes a machine check. It is cleared by writing to the MSER.
03:02	1—Read as ones.
01	DAT (Cache parity error in data)—Set when the cache parity error was detected in the data. It is cleared by writing to the MSER.
00	TAG (Cache parity error in tag)—Set when the cache parity error was in the tag. It is cleared by writing to the MSER. This is the only bit set when a cache entry has a parity error in both its tag and data.

Processor Status Longword (PSL)—The PSL contains processor status information. The lower 16 bits are the user accessible Processor Status Word (PSW). The upper 16 bits are privileged and accessed only by the system. The format of the PSL is shown in Figure 7 and described in Table 8. Refer to the *Cache Memory* section for a description of the PSW.

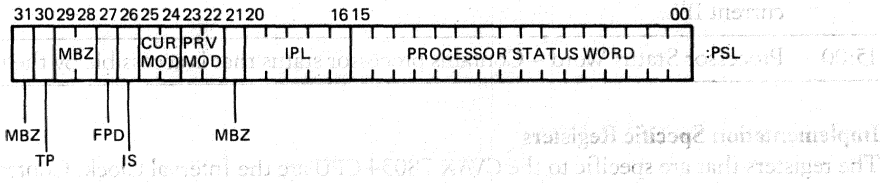


Figure 7 • CVAX 78034 Processor Status Longword Format

Table 8 • CVAX 78034 Processor Status Longword Description

Bit	Description
31	MBZ—Must be zero.
30	TP (Trace pending)—Forces a trace trap when set at the beginning of any instruction. Set by the processor if the T bit in the PSW is set at the beginning of an instruction.
29:28	MBZ—Must be zero.
27	FPD (First part done)—Set when an exception or interrupt occurs during an instruction that can be suspended. If FPD is set when the processor returns from an exception or interrupt, it resumes the interrupted operation from where it stopped rather than restarting the complete instruction.
26	IS (Interrupt stack)—Set when the processor is executing on the interrupt stack.

Bit	Description
25:24	CUR MOD (Current mode)—Indicates the access mode of the currently executing process.

Bit 25	Bit 24	Mode
0	0	Kernel
0	1	Executive
1	0	Supervisor
1	1	User

23:22	PRV MOD (Previous mode)—Loaded from CUR MOD bits 25:24 by exceptions and Change mode (CHMx) instructions. Cleared by interrupts and restored by Return from Exception or Interrupt (REI) instruction.
21	MBZ—Must be zero.
20:16	IPL (Interrupt Priority Level)—Contains the current processor priority in the range 0 to 1F (hexadecimal). The processor will accept interrupts only on levels greater than the its current IPL.
15:00	Processor Status Word—Contains processor status that is accessible by the user.

Implementation Specific Registers

The registers that are specific to the CVAX 78034 CPU are the Interval Clock, Control and Status (ICCS) register, Console Saved PSL (SAVPSL) register, and the Console Saved PC (SAVPC) register.

Interval Clock, Control, and Status Register (ICCS)—The ICCS register, Figure 8, controls the interval timer interrupt. It contains a read/write IE bit 06 that is used to enable or disable interval timer interrupts generated by the assertion of the INTTIM input. When this bit is set, the interval timer interrupts are enabled and the assertion of INTTIM results in an interrupt request at IPL 16 (hexadecimal). When this bit is clear, the interval timer interrupts are disabled and the assertion of INTTIM does not generate an interrupt request. Bits 31:07 and 05:00 are read as zeros and are ignored during write operations.

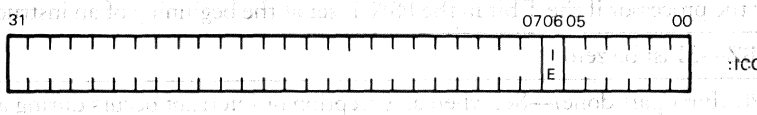


Figure 8 • CVAX 78034 Interval Clock, Control, and Status Register Format

Console Saved Registers (SAVPC and SAVPSL)—The SAVPC and SAVPSL registers record the value of the PC and PSL when the CVAX CPU restarts. The SAVPC register contains the previous value of the PC before the restart operation. The SAVPC and SAVPSL register formats are shown in Figure 9. The SAVPSL register contains the information described in Table 9.

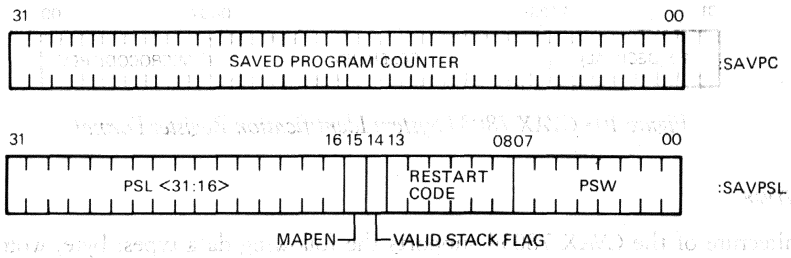


Figure 9 • CVAX 78034 Console Saved Register Formats

Table 9 • CVAX 78034 Console Saved Processor Status Longword Register Description

Bit	Description
31:16	PSL (Processor Status Longword)—Contains the previous PSL value.
15	MAPEN (Map Enable)—Set to enable the map.
14	Valid stack flag—Set to indicate a valid stack flag.
13:08	Restart Code—Contains the restart code (hexadecimal) as follows

Code	Definition
2	$\overline{\text{HALT}}$ asserted
3	initial power on
4	interrupt stack not valid during exception
5	machine check normal exception
6	HALT instruction executed in kernel mode
7	SCB vector bits 01:00 = 11
8	SCB vector bits 01:00 = 10
A	CHMx executed while on interrupt stack
10	ACV or TNV during machine check exception
11	ACV or TNV during kernel stack not valid exception
12	machine check during machine check exception
13	machine check during kernel stack not valid exception
19	PSL bits 26:24 = 101 during interrupt or exception
1A	PSL bits 26:24 = 110 during interrupt or exception
1B	PSL bits 26:24 = 111 during interrupt or exception
1D	PSL bits 26:24 = 101 during REI
1E	PSL bits 26:24 = 110 during REI
1F	PSL bitd 26:24 = 111 during REI

07:00	PSW (Processor Status Word)—Contains the previous PSW value.

System Identification Register (SID)—The SID register is a read-only register that specifies the processor type as a CVAX CPU and defines its microcode revision level. Figure 10 shows the register format.

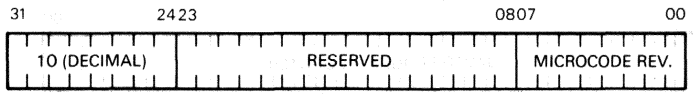


Figure 10 • CVAX 78034 System Identification Register Format

• Data Types

The architecture of the CVAX 78034 supports the following data types: byte, word, longword, quadword, character string, variable-length bit field, and, through the optional floating-point accelerator, F_floating, D_floating, and G_floating. Figures 11 shows the integer, character string, and field data types. Figure 12 shows the floating-point data types.

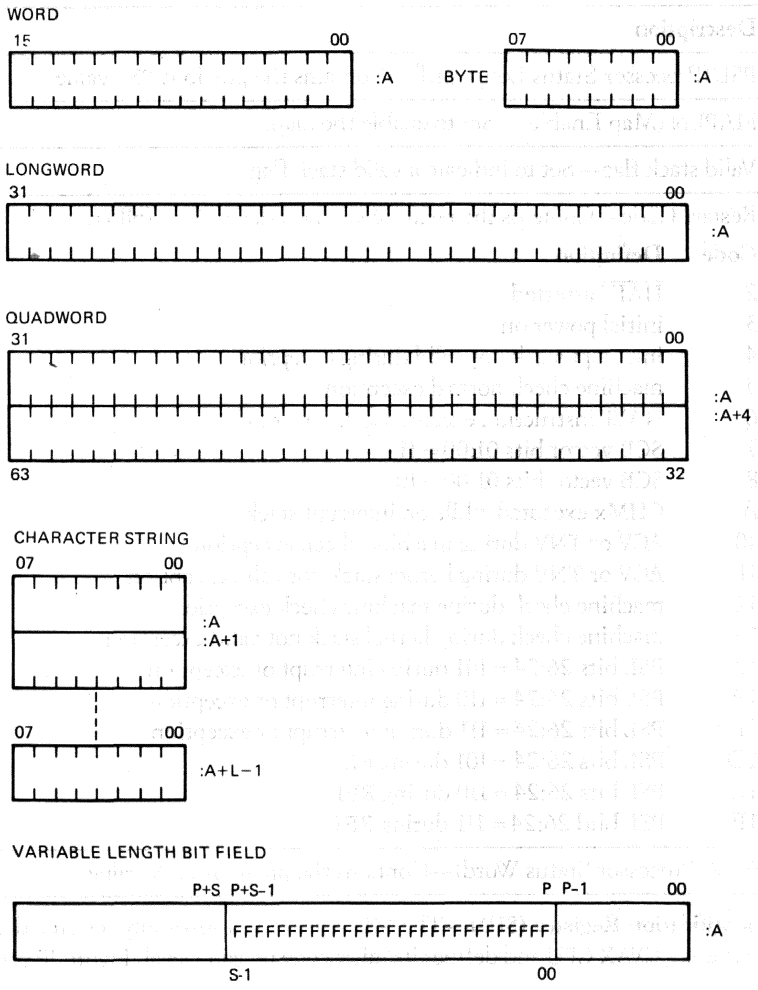


Figure 11 • CVAX 78034 Integer, Character String, and Field Data Types

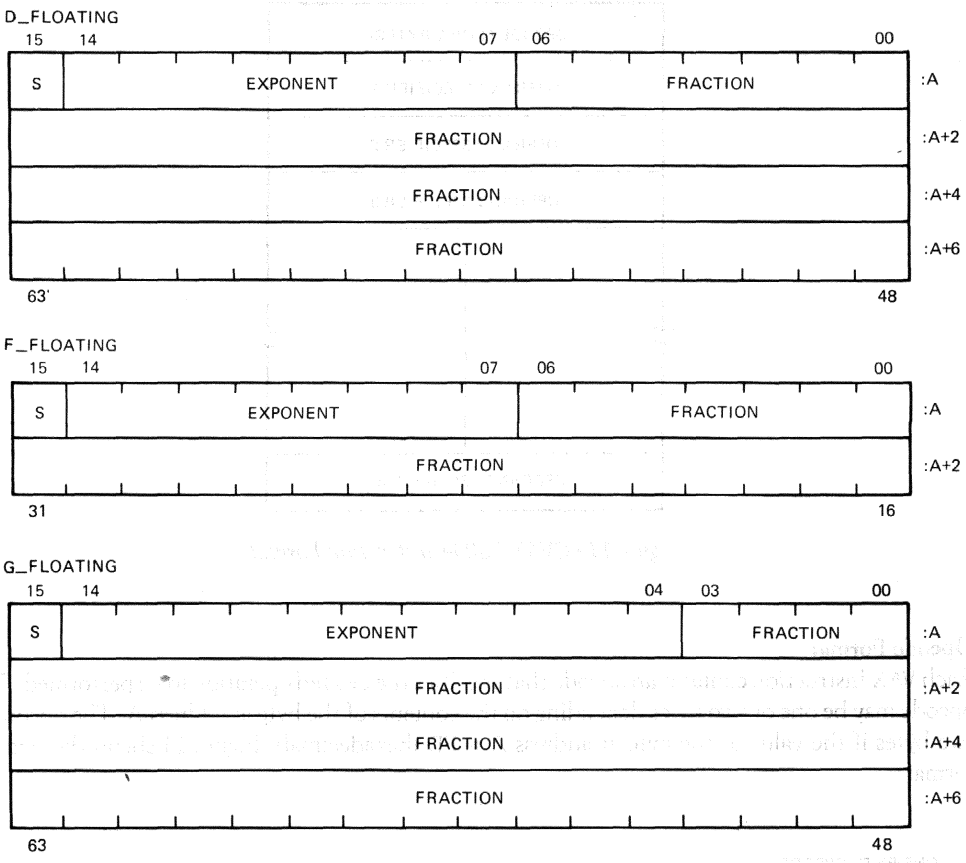


Figure 12 ▪ CVAX 78034 Floating-point Data Types

▪ Instruction Formats

The VAX instruction set has a variable-length instruction format that may be one byte or more depending on the type of instruction. The general format of a VAX instruction is shown in Figure 13. Each instruction is made up of an operation code (opcode) followed by no operand or up to six operand specifiers. The number and type of operand specifiers depend on the opcode. All operand specifiers are similar and consist of an address mode plus additional information used to locate the operand. This additional information contains up to two register designators and addresses, data, or displacement values. The use of the operand is determined implicitly from the opcode and is the operand type. It includes both the access type and the data type.

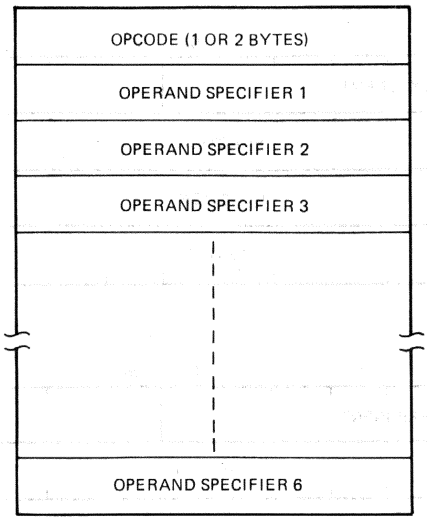


Figure 13 • CVAX 78034 Instruction Format

Opcode Format

Each VAX instruction contains an opcode that specifies the desired operation to be performed. The opcode may be one or two bytes depending on the contents of the byte at address A. The opcode is two bytes if the value of the byte at address A is FD (hexadecimal). Figure 14 shows the opcode format.

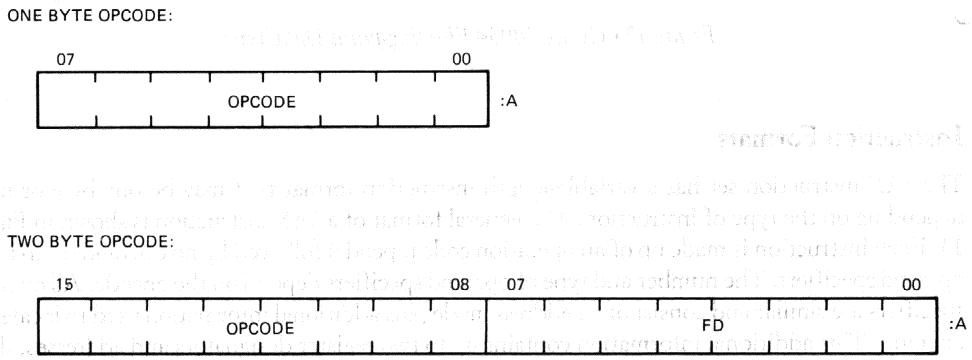


Figure 14 • CVAX 78034 Opcode Format

Operand Type

The operand type specifies the use of the operand associated with an instruction. Information provided by the opcode includes the data type of each operand and its method of access. An operand may be accessed as follows:

- Read—The specified operand is read-only.
- Write—The specified operand is write-only.
- Modify—The specified operand is read, may or may not be modified, and is written.
- Address—Address calculation occurs until the actual address of the operand is obtained. In this mode, the data type indicates the operand size to be used in the address calculation. The specified operand is not accessed directly although the instruction may use the address to access that operand.
- Variable bit field base address—If only R[n] is specified, the field is in general register R[n] or in R[n+1]'R[n] (i.e., R[n+1] concatenated with R[n]). Otherwise, the address calculation occurs until the actual address of the operand is obtained. This address specifies the base to which the field position (offset) is applied.
- Branch—No operand is accessed. The operand specifier is the branch displacement. In the specifier, the data type indicates the size of the branch displacement.

Addressing Modes

A summary of the addressing modes used by the CVAX 78034 is listed in Table 10. A brief description of each mode follows.

Table 10 • CVAX 78034 Summary of Addressing Modes

General Register Addressing Mode

			Access							
Hexadecimal	**Name**	**Assembler**	**r**	**m**	**w**	**a**	**v**	**PC**	**SP**	**Indexable?**
0–3	literal	S^ #literal	y	f	f	f	f	–	–	f
4	index	i (Rx)	y	y	y	y	y	f	y	f
5	register	Rn	y	y	y	f	y	u	uq	f
6	register deferred	(Rn)	y	y	y	y	y	u	y	y
7	autodecrement	-(Rn)	y	y	y	y	y	u	y	ux
8	autoincrement	(Rn)+	y	y	y	y	y	p	y	ux
9	autoincrement deferred	@(Rn)+	y	y	y	y	y	p	y	ux
A	byte displacement	B^ d(Rn)	y	y	y	y	y	p	y	y
B	byte displacement deferred	@B^ d(Rn)	y	y	y	y	y	p	y	y
C	word displacement	W^ d(Rn)	y	y	y	y	y	p	y	y
D	word displacement deferred	@W^ d(Rn)	y	y	y	y	y	p	y	y
E	longword displacement	L^ d(Rn)	y	y	y	y	y	p	y	y
F	longword displacement deferred	@L^ d(Rn)	y	y	y	y	y	p	y	y

Program Counter Addressing Mode

Hexadecimal	Name	Assembler	Access r	m	w	a	v	Indexable?
8	immediate	I^ #constant	y	u	u	y	y	u
9	absolute	@#address	y	y	y	y	y	y
A	byte relative	B^ address	y	y	y	y	y	y
B	byte relative deferred	@B^ address	y	y	y	y	y	y
C	word relative	W^ address	y	y	y	y	y	y
D	word relative deferred	W^ address	y	y	y	y	y	y
E	longword relative	L^ address	y	y	y	y	y	y
F	longword relative deferred	L^ address	y	y	y	y	y	y

Addressing Legend

Access:	Syntax:
r = read	i = any indexable address mode
m = modify	d = displacement
w = write	Rn = general register, n = 0 to 15
a = address	Rx = general register, x = 0 to 14
v = field	

Results:

y = yes, always valid address mode
f = reserved address mode fault
– = logically impossible
p = program counter addressing
u = unpredictable
uq = unpredictable for quad, D_/G_floating, or field if pos + size > 32
ux = unpredictable if index reg = base reg

General Register Address Modes

The general register address modes use one or more general registers, depending on the instruction and data type, to contain the operand(s) or information required to locate the operand(s) to be used by the specified instruction.

Register Mode—The operand is contained in one of the general registers (Rn).

Register Deferred Mode—Register Rn contains the address of the operand.

Autoincrement Mode—Register Rn contains the address of the operand. After the operand address is determined, the size of the operand in bytes (determined by its data type) is added to the contents of Rn and the result is placed in Rn.

Autoincrement Deferred Mode—Register Rn contains a longword address that is a pointer to the operand address. After the operand address has been determined, the value of four is added to the contents of Rn and the contents of Rn are replaced by the result.

Autodecrement Mode—The size of the operand in bytes (determined by its data type) is subtracted from the contents of Rn and the contents of Rn are replaced by the result. The updated contents of Rn are the address of the operand.

Literal Mode—Literal mode addressing provides an efficient means of specifying integer constants in the range from 0 to 63 (decimal). In addition to short integer literals, this mode can be used to specify floating-point literals. The value is contained in the operand specifier.

Displacement Mode—The displacement contained in the operand specifier, after being sign-extended to 32 bits if it is a byte or word, is added to the contents of register Rn. The result is the operand address.

Displacement Deferred Mode—The displacement contained in the operand specifier, after being sign-extended to 32 bits if it is a byte or word, is added to the contents of register Rn. The result is the longword address of the operand address.

Index Mode—The operand specifier consists of a minimum of two bytes, a primary operand specifier, and a base operand specifier. The primary operand specifier contained in bits 0 through 7 includes the index register (Rx) and a mode specifier of 4. The address of the primary operand is determined by multiplying the contents of index register Rx by the size of the primary operand in bytes as determined by operand type. This value is then added to the address specified by the base operand specifier (bits 15:08) and the result is used as the primary operand address.

Program Counter Addressing

Register 15 is used as the Program Counter (PC). It can also be used as a register in addressing modes. The processor increments the program counter as the opcode, operand specifier, and immediate data or addresses of the instruction are evaluated. The incremented value is determined by the opcode, number of operand specifiers, etc. The PC can be used with all VAX addressing modes except register, index, register deferred, or autodecrement.

Immediate mode—This mode is autoincrement mode when the PC is used as the general register. The contents of the location following the addressing mode are immediate data.

Absolute mode—This mode is autoincrement deferred using the PC as the general register. The contents of the location following the addressing mode are used as the operand address. This is interpreted as an absolute address (an address that remains constant regardless of the location memory where the assembled instruction is executed).

Relative mode—This mode is displacement mode with the PC used as the general register. The displacement that follows the operand specifier is added to the contents of the PC, and the result is the address of the operand.

Relative deferred mode—This mode is similar to relative mode except that the displacement that follows the addressing mode is added to the contents of the PC, and the result is the longword address of the operand.

Branch Addressing

During branch displacement addressing, the byte or word displacement is sign-extended to 32 bits and added to the updated content of the PC. The updated content of the PC is the address of the first byte beyond the operand specifier.

• Instruction Set

This section provides a summary of the VAX instructions implemented by the CVAX 78034, the floating-point instructions supported by the floating-point accelerator, and the emulated instructions that are assisted by the microcode of the CVAX 78034. The standard notation used for the operand specifiers is

<name> <access type> <data type>

1. Name—A suggestive name for the operand in the context of the instruction. It is the capitalized name of a register or block for implied operands.
2. Access type—A letter denoting the operand specifier access type.
   a = address operand
   b = branch displacement
   m = modified operand (both read and written)
   r = read only operand
   v = if not "Rn," same as address operand, otherwise R[n + 1]'R[n]
   w = write only operand
3. Data type—A letter denoting the data type of the operand.
   b = byte
   d = D_floating
   f = F_floating
   g = G_floating
   l = longword
   q = quadword
   v = field (used only in implied operands)
   w = word
   * = multiple longwords (used only in implied operands)
4. Implied operands—Locations that are accessed by the instruction, but not specified in an operand, are denoted by braces { }. The abbreviations for condition codes are
   * = conditionally set/cleared
   – = not affected
   0 = cleared
   1 = set

   The abbreviations for exceptions are
   rsv = reserved operand fault
   iov = integer overflow trap
   idvz = integer divide by zero trap
   fov = floating overflow fault
   fuv = floating underflow fault
   fdvz = floating divide by zero fault
   dov = decimal overflow trap
   ddvz = decimal divide by zero trap
   sub = subscript range trap
   prv = privileged instruction fault

   Opcode values are given in hexadecimal.

Integer Arithmetic and Logical Instructions

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
58	ADAW1 add.rw, sum.mw	Add aligned word interlocked	*	*	*	*	iov
80	ADDB2 add.rb, sum.mb	Add byte 2-operand	*	*	*	*	iov
C0	ADDL2 add.rl, sum.ml	Add long 2-operand	*	*	*	*	iov
A0	ADDW2 add.rw, sum.mw	Add word 2-operand	*	*	*	*	iov
81	ADDB3 add1.rb, add2.rb, sum.wb	Add byte 3-operand	*	*	*	*	iov
C1	ADDL3 add1.rl, add2.rl, sum.wl	Add long 3-operand	*	*	*	*	iov
A1	ADDW3 add1.rw, add2.rw, sum.ww	Add word 3-operand	*	*	*	*	iov
D8	ADWC add.rl, sum.ml	Add with carry	*	*	*	*	iov
78	ASHL cnt.rb src.rl, dst.wl	Arithmetic shift left	*	*	*	0	iov
79	ASHQ cnt.rb src.rq, dst.wq	Arithmetic shift quad	*	*	*	0	iov
8A	BICB2 mask.rb, dst.mb	Bit clear byte 2-operand	*	*	0	–	
CA	BICL2 mask.rl, dst.ml	Bit clear long 2-operand	*	*	0	–	
AA	BICW2 mask.rw, dst.mw	Bit clear word 2-operand	*	*	0	–	
8B	BICB3 mask.rb, src.rb, dst.wb	Bit clear byte 3-operand	*	*	0	–	
CB	BICL3 mask.rl, src.rl, dst.ml	Bit clear long 3-operand	*	*	0	–	
AB	BICW3 mask.rw, src.rw, dst.mw	Bit clear word 3-operand	*	*	0	–	
88	BISB2 mask.rb, dst.mb	Bit set byte 2-operand	*	*	0	–	
C8	BISL2 mask.rl, dst.ml	Bit set long 2-operand	*	*	0	–	
A8	BISW2 mask.rw, dst.mw	Bit set word 2-operand	*	*	0	–	
89	BISB3 mask.rb, src.rb, dst.mb	Bit set byte 3-operand	*	*	0	–	
C9	BISL3 mask.rl, src.rl, dst.ml	Bit set long 3-operand	*	*	0	–	
A9	BISW3 mask.rw, src.rw, dst.mw	Bit set word 3-operand	*	*	0	–	
93	BITB mask.rb, src.rb	Bit test byte	*	*	0	–	
D3	BITL mask.rl, src.rl	Bit test long	*	*	0	–	
B3	BITW mask.rw, src.rw	Bit test word	*	*	0	–	
94	CLRB dst.wb	Clear byte	0	1	0	–	
D4	CLRL dst.wl	Clear long	0	1	0	–	
7C	CLRQ dst.wq	Clear quad	0	1	0	–	
B4	CLRW dst.ww	Clear word	0	1	0	–	
91	CMPB src1.rb, src2.rb	Compare byte	*	*	0	*	
D1	CMPL src1.rl, src2.rl	Compare long	*	*	0	*	
B1	CMPW src1.rw, src2.rw	Compare word	*	*	0	*	
98	CVTBL src.rb, dst.wl	Convert byte to long	*	*	0	0	
99	CVTBW src.rb, dst.wl	Convert byte to word	*	*	0	0	
F6	CVTLB src.rl, dst.wb	Convert long to byte	*	*	*	0	iov
F7	CVTLW src.rl, dst.ww	Convert long to word	*	*	*	0	iov
33	CVTWB src.rw, dst.wb	Convert word to byte	*	*	*	0	iov
32	CVTWL src.rw, dst.wl	Convert word to long	*	*	0	0	
97	DECB dif.mb	Decrement byte	*	*	*	*	iov
D7	DECL dif.l	Decrement long	*	*	*	*	iov
97	DECW dif.mw	Decrement word	*	*	*	*	iov
86	DIVB2 divr.rb, quo.mb	Divide byte 2-operand	*	*	*	0	iov, idvz
C6	DIVL2 divr.rl, quo.ml	Divide long 2-operand	*	*	*	0	iov, idvz
A6	DIVW2 divr.rw, quo.mw	Divide word 2-operand	*	*	*	0	iov, idvz

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
87	DIVB3 divr.rb, divd.rb, quo.wb	Divide byte 3-operand	*	*	*	0	iov, idvz
C7	DIVL3 divr.rl, divd.rl, quo.wl	Divide long 3-operand	*	*	*	0	iov, idvz
A7	DIVW3 divr.rw, divd.rw, quo.ww	Divide word 3-operand	*	*	*	0	iov, idvz
7B	EDIV divr.rl, divd.rq, quo.wl, rem.wl	Extended divide	*	*	*	0	iov, idvz
7A	EMUL mulr.rl, muld.rl, add.rl, prod.wq	Extended multiply	*	*	0	0	
96	INCB sum.mb	Increment byte	*	*	*	*	iov
D6	INCL sum.ml	Increment long	*	*	*	*	iov
B6	INCW sum.mw	Increment word	*	*	*	*	iov
92	MCOMB src.rb, dst.wb	Move complemented byte	*	*	0	–	
D2	MCOML src.rl, dst.wl	Move complemented long	*	*	0	–	
B2	MCOMW src.rw, dst.ww	Move complemented word	*	*	0	–	
8E	MNEGB src.rb, dst.wb	Move negated byte	*	*	*	*	iov
CE	MNEGL src.rl, dst.wl	Move negated long	*	*	*	*	iov
AE	MNEGW src.rw, dst.ww	Move negated word	*	*	*	*	iov
90	MOVB src.rb, dst.wb	Move byte	*	*	0	–	
D0	MOVL src.rl, dst.wl	Move long	*	*	0	–	
B0	MOVW src.rw, dst.ww	Move word	*	*	0	–	
9A	MOVZBW src.rb, dst.wb	Move zero-extended byte to word	0	*	0	–	
9B	MOVZBL src.rb, dst.wl	Move zero-extended byte to long	0	*	0	–	
3C	MOVZWL src.rw, dst.ww	Move zero-extended word to long	0	*	0	–	
84	MULB2 mulr.rb, prod.mb	Multiply byte 2-operand	*	*	*	0	iov
C4	MULL2 mulr.rl, prod.ml	Multiply long 2-operand	*	*	*	0	iov
A4	MULW2 mulr.rw, prod.mw	Multiply word 2-operand	*	*	*	0	iov
85	MULB3 mulr.rb, muld.rb, prod.mb	Multiply byte 3-operand	*	*	*	0	iov
C5	MULL3 mulr.rl, muld.rl, prod.ml	Multiply long 3-operand	*	*	*	0	iov
A5	MULW3 mulr.rw, muld.rw, prod.mw	Multiply word 3-operand	*	*	*	0	iov
DD	PUSHL src.rl,	Push long	*	*	0	–	
9C	ROTL cnt.rb, src.rl, dst.wl	Rotate long	*	*	0	–	
D9	SBWC sub.rl, dif.ml	Subtract with carry	*	*	*	*	iov
82	SUBB2 sub.rb, dif.mb	Subtract byte 2-operand	*	*	*	*	iov
C2	SUBL2 sub.rl, dif.ml	Subtract long 2-operand	*	*	*	*	iov
A2	SUBW2 sub.rw, dif.mw	Subtract word 2-operand	*	*	*	*	iov
83	SUBB3 sub.rb, min.rb, dif.mb	Subtract byte 3-operand	*	*	*	*	iov
C3	SUBL3 sub.rl, min.rl, dif.ml	Subtract long 3-operand	*	*	*	*	iov
A3	SUBW3 sub.rw, min.rw, dif.mw	Subtract word 3-operand	*	*	*	*	iov
95	TSTB src.rb	Test byte	*	*	0	0	
D5	TSTL src.rl	Test long	*	*	0	0	
B5	TSTW src.rw	Test word	*	*	0	0	
8C	XORB2 mask.rb, dst.mb	Exclusive or byte 2-operand	*	*	0	–	
CC	XORL2 mask.rl, dst.ml	Exclusive or long 2-operand	*	*	0	–	
AC	XORW2 mask.rw, dst.mw	Exclusive or word 2-operand	*	*	0	–	
8D	XORB3 mask.rb, src.rb, dst.wb	Exclusive or byte 3-operand	*	*	0	–	
CD	XORL3 mask.rl, src.rl, dst.wl	Exclusive or long 3-operand	*	*	0	–	
AD	XORW3 mask.rw, src.rw, dst.ww	Exclusive or word 3-operand	*	*	0	–	

• Address Instructions

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
9E	MOVAB src.ab, dst.wl	Move address of byte	*	*	0	–	
DE	MOVAL {=F} src.al, dst.wl	Move address of long	*	*	0	–	
7E	MOVAQ {=D=G} src.aq, dst.wl	Move address of quad	*	*	0	–	
3E	MOVAW src.aw, dst.wl	Move address of word	*	*	0	–	
9F	PUSHAB src.ab, {-(SP).wl}	Push address of byte	*	*	0	–	
DF	PUSHAL {=F} src.al, {-(SP).wl}	Push address of long	*	*	0	–	
7F	PUSHAQ {=D=G} src.aq, {-(SP).wl}	Push address of quad	*	*	0	–	
3F	PUSHAW src.aw, {-(SP).wl}	Push address of word	*	*	0	–	

• Variable-length Bit Field Instructions

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
EC	CMPV pos.rl, size.rb, base.rb, {field.rv}, src.rl	Compare field	*	*	0	*	rsv
ED	CMPZV pos.rl, size.rb, base.vb, {field.rv}, src.rl	Compare zero-extended field	*	*	0	*	rsv
EE	EXTV pos.rl, size.rb, base.vb, {field.rv}, dst.wl	Extract field	*	*	0	–	rsv
EF	EXTZV pos.rl, size.rb, base.vb, {field.rv}, dst.wl	Extract zero-extended field	*	*	0	–	rsv
F0	INSV src.rl, pos.rl, size.rb, base.vb, {field.wv}	Insert field	–	–	–	–	rsv
EB	FFC startpos.rl size.rb, base.vb, {field.rv}, findpos.wl	Find first clear bit	*	*	0	–	rsv
EA	FFS startpos.rl, size.rb, base.vb, {field.rv}, findpos.wl	Find first set bit	*	*	0	–	rsv

• Control Instructions

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
9D	ACBB limit.rb, add.rb, index.mb, displ.bw	Add compare and branch byte	*	*	*	–	iov
F1	ACBL limit.rl, add.rl, index.ml, displ.bw	Add compare and branch long	*	*	*	–	iov
3D	ACBW limit.rw, add.rw, index.mw, displ.bw	Add compare and branch word	*	*	*	–	iov
F3	AOBLEQ limit.rl, index.ml, displ.bb	Add one and branch on less or equal	*	*	*	–	iov
F2	AOBLSS limit.rl, index.ml, displ.bb	Add one and branch on less	*	*	*	–	iov

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
1E	BCC{=BGEQU} displ.bb	Branch on carry clear	–	–	–	–	–
1F	BCS{=BLSSU} displ.bb	Branch on carry set	–	–	–	–	–
13	BEQL{=BEQLU} displ.bb	Branch on equal	–	–	–	–	–
18	BGEQ displ.bb	Branch on greater or equal	–	–	–	–	–
14	BGTR displ.bb	Branch on greater	–	–	–	–	–
1A	BGTRU displ.bb	Branch on greater unsigned	–	–	–	–	–
15	BLEQ displ.bb	Branch on less or equal	–	–	–	–	–
1B	BLEQU displ.bb	Branch on less or equal unsigned	–	–	–	–	–
19	BLSS displ.bb	Branch on less	–	–	–	–	–
12	BNEQ {=BNEQU} displ.bb	Branch on not equal	–	–	–	–	–
1C	BVC displ.bb	Branch on overflow clear	–	–	–	–	–
1D	BVS displ.bb	Branch on overflow set	–	–	–	–	–
E1	BBC pos.rl, base.vb, displ.bb, {field.rv}	Branch on bit clear	–	–	–	–	rsv
E0	BBS pos.rl, base.vb, displ.bb, {field.rv}	Branch on bit set	–	–	–	–	rsv
E5	BBCC pos.rl, base.vb, displ.bb, {field.mv}	Branch on bit clear and clear	–	–	–	–	rsv
E3	BBCS pos.rl, base.vb, displ.bb, {field.mv}	Branch on bit clear and set	–	–	–	–	rsv
E4	BBSC pos.rl, base.vb, displ.bb, {field.mv}	Branch on bit set and clear	–	–	–	–	rsv
E2	BBSS pos.rl, base.vb, displ.bb, {field.mv}	Branch on bit set and set	–	–	–	–	rsv
E7	BBCCI pos.rl, base.vb, dislp.bb, {field.mv}	Branch on bit clear and clear interlocked	–	–	–	–	rsv
E6	BBSSI pos.rl, base.vb, dislp.bb, {field.mv}	Branch on bit set and set interlocked	–	–	–	–	rsv
E9	BLBC src.rl, displ.bb	Branch on low bit clear	–	–	–	–	–
E8	BLBS src.rl, displ.bb	Branch on low bit set	–	–	–	–	–
11	BRB displ.bb	Branch with byte displacement	–	–	–	–	–
31	BRW displ.bw	Branch with word displacement	–	–	–	–	–
10	BSBB displ.bb {-(SP).wl}	Branch to subroutine with byte displacement	–	–	–	–	–
30	BSBW displ.bw {-(SP).wl}	Branch to subroutine with word displacement	–	–	–	–	–
8F	CASEB selector.rb, base.rb, limit.rb, displ.bw-list	Case byte	*	*	0	*	
CF	CASEL selector.rl, base.rl, limit.rl, displ.bw-list	Case long	*	*	0	*	
AF	CASEW selector.rw, base.rw, limit.rw, displ.bw-list	Case word	*	*	0	*	
17	JMP dst.ab	Jump	–	–	–	–	–
16	JSB dst.ab, {-(SP).wl}	Jump to subroutine	–	–	–	–	–
05	RSB {(SP)+.rl}	Return from subroutine	–	–	–	–	–
F4	SOBGEQ index.ml, displ.bb	Subtract one and branch on greater or equal	*	*	*	–	iov
F5	SOBGTR index.ml, displ.bb	Subtract one and branch on greater	*	*	*	–	iov

• Variable-length Bit Field Instructions

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
EC	CMPV pos.rl, size.rb, base.rb, {field.rv}, src.rl	Compare field	*	*	0	*	rsv
ED	CMPZV pos.rl, size.rb, base.vb, {field.rv}, src.rl	Compare zero-extended field	*	*	0	*	rsv
EE	EXTV pos.rl, size.rb, base.vb, {field.rv}, dst.wl	Extract field	*	*	0	–	rsv
EF	EXTZV pos.rl, size.rb, base.vb, {field.rv}, dst.wl	Extract zero-extended field	*	*	0	–	rsv
F0	INSV src.rl, pos.rl, size.rb, base.vb, {field.wv}	Insert field	–	–	–	–	rsv
EB	FFC startpos.rl size.rb, base.vb, {field.rv}, findpos.wl	Find first clear bit	*	*	0	–	rsv
EA	FFS startpos.rl, size.rb, base.vb, {field.rv}, findpos.wl	Find first set bit	*	*	0	–	rsv

• Control Instructions

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
9D	ACBB limit.rb, add.rb, index.mb, displ.bw	Add compare and branch byte	*	*	*	–	iov
F1	ACBL limit.rl, add.rl, index.ml, displ.bw	Add compare and branch long	*	*	*	–	iov
3D	ACBW limit.rw, add.rw, index.mw, displ.bw	Add compare and branch word	*	*	*	–	iov
F3	AOBLEQ limit.rl, index.ml, displ.bb	Add one and branch on less or equal	*	*	*	–	iov
F2	AOBLSS limit.rl, index.ml, displ.bb	Add one and branch on less	*	*	*	–	iov
1E	BCC{=BGEQU} displ.bb	Branch on carry clear	–	–	–	–	–
1F	BCS{=BLSSU} displ.bb	Branch on carry set	–	–	–	–	–
13	BEQL{=BEQLU} displ.bb	Branch on equal	–	–	–	–	–
18	BGEQ displ.bb	Branch on greater or equal	–	–	–	–	–
14	BGTR displ.bb	Branch on greater	–	–	–	–	–
1A	BGTRU displ.bb	Branch on greater unsigned	–	–	–	–	–
15	BLEQ displ.bb	Branch on less or equal	–	–	–	–	–
1B	BLEQU displ.bb	Branch on less or equal unsigned	–	–	–	–	–
19	BLSS displ.bb	Branch on less	–	–	–	–	–
12	BNEQ {=BNEQU} displ.bb	Branch on not equal	–	–	–	–	–
1C	BVC displ.bb	Branch on overflow clear	–	–	–	–	–
1D	BVS displ.bb	Branch on overflow set	–	–	–	–	–

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
E1	BBC pos.rl, base.vb, displ.bb, {field.rv}	Branch on bit clear	–	–	–	–	rsv
E0	BBS pos.rl, base.vb, displ.bb, {field.rv}	Branch on bit set	–	–	–	–	rsv
E5	BBCC pos.rl, base.vb, displ.bb, {field.mv}	Branch on bit clear and clear	–	–	–	–	rsv
E3	BBCS pos.rl, base.vb, displ.bb, {field.mv}	Branch on bit clear and set	–	–	–	–	rsv
E4	BBSC pos.rl, base.vb, displ.bb, {field.mv}	Branch on bit set and clear	–	–	–	–	rsv
E2	BBSS pos.rl, base.vb, displ.bb, {field.mv}	Branch on bit set and set	–	–	–	–	rsv
E7	BBCCI pos.rl, base.vb, dislp.bb, {field.mv}	Branch on bit clear and clear interlocked	–	–	–	–	rsv
E6	BBSSI pos.rl, base.vb, dislp.bb, {field.mv}	Branch on bit set and set interlocked	–	–	–	–	rsv
E9	BLBC src.rl, displ.bb	Branch on low bit clear	–	–	–	–	–
E8	BLBS src.rl, displ.bb	Branch on low bit set	–	–	–	–	–
11	BRB displ.bb	Branch with byte displacement	–	–	–	–	–
31	BRW displ.bw	Branch with word displacement	–	–	–	–	–
10	BSBB displ.bb {-(SP).wl}	Branch to subroutine with byte displacement	–	–	–	–	–
30	BSBW displ.bw {-(SP).wl}	Branch to subroutine with word displacement	–	–	–	–	–
8F	CASEB selector.rb, base.rb, limit.rb, displ.bw-list	Case byte	*	*	0	*	
CF	CASEL selector.rl, base.rl, limit.rl, displ.bw-list	Case long	*	*	0	*	
AF	CASEW selector.rw, base.rw, limit.rw, displ.bw-list	Case word	*	*	0	*	
17	JMP dst.ab	Jump	–	–	–	–	–
16	JSB dst.ab, {-(SP).wl}	Jump to subroutine	–	–	–	–	–
05	RSB {(SP)+.rl}	Return from subroutine	–	–	–	–	–
F4	SOBGEQ index.ml, displ.bb	Subtract one and branch on greater or equal	*	*	*	–	iov
F5	SOBGTR index.ml, displ.bb	Subtract one and branch on greater	*	*	*	–	iov

• Procedure Call Instructions

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
FA	CALLG arglist.ab, dst.ab, {-(SP).w*}	Call with general argument list	0	0	0	0	rsv
FB	CALLS numarg.rl, dst.ab, {-(SP).w*}	Call with argument list on stack	0	0	0	0	rsv
04	RET {(SP)+.r*}	Return from procedure	*	*	*	*	rsv

• Miscellaneous Instructions

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
B9	BICPSW mask.rw	Bit clear processor status word	*	*	*	*	rsv
B8	BISPSW mask.rw	Bit set processor status word	*	*	*	*	rsv
03	BPT {-(KSP).w*}	Break point fault	0	0	0	0	
00	HALT {-(KSP).w*}	Halt (kernel mode only)	–	–	–	–	prv
0A	INDEX subscript.rl, low.rl, high.rl, size.rl, indexin.rl, indexout.wl	Index calculation	*	*	0	0	sub
DC	MOVPSL dst.wl	Move processor status longword	–	–	–	–	
01	NOP	No operation	–	–	–	–	
BA	POPR mask.rw, ;{(SP)+.r*}	Pop registers	–	–	–	–	
BB	PUSHR mask.rw, ;{-(SP)+.w*}	Push registers	–	–	–	–	
FC	XFC {unspecified oeprands}	Extended function call	0	0	0	0	

• Queue Instructions

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
5C	INSQHI entry.ab header.aq	Insert at head of queue, interlocked	0	*	0	*	rsv
5D	INSQTI entry.ab header.aq	Insert at tail of queue, interlocked	0	*	0	*	rsv
0E	INSQUE entry.ab, pred.ab	Insert into queue	*	*	0	*	
5E	REMQHI header.aq, addr.wl	Remove from head of queue, interlocked	0	*	*	*	rsv
5F	REMQTI header.aq, addr.wl	Remove from tail of queue, interlocked	0	*	*	*	rsv
0E	REMQUE entry.ab, addr.wl	Remove from queue	*	*	*	*	

• Character String Instructions

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
29	CMPC3 len.rw, src1addr.ab, src2addr.ab	Compare character 3-operand	*	*	0	*	
2D	CMPC5 src1len.rw, src1addr.ab, fill.rb, src2len.rw, src2addr.ab	Compare character 5-operand	*	*	0	*	
3A	LOCC char.rb, len.rw, addr.ab	Locate character	0	*	0	0	
28	MOVC3 len.rw, srcaddr.ab, dstaddr.ab, {R0-5.wl}	Move character 3-operand	0	1	0	0	
2C	MOVC5 srclen.rw, srcaddr.ab, fill.rb, dstlen.rw, dstaddr.ab, {R0-5.wl}	Move character 5-operand	*	*	0	*	
2A	SCANC len.rw, addr.ab, tbladdr.ab, mask.rb	Scan for character	0	*	0	0	
3B	SKPC char.rb, len.rw, addr.ab	Skip character	0	*	0	0	
2B	SPANC len.rw, len.rw, tbladdr.ab, mask.rb	Scan characters	0	*	0	0	

• System Support Instructions

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
BD	CHME param.rw, {-(ySP).w*}	Change mode to executive	0	0	0	0	
BC	CHMK param.rw, {-(ySP).w*}	Change mode to kernel	0	0	0	0	
BE	CHMS param.rw, {-(ySP).w*}	Change mode to supervisor	0	0	0	0	
BF	CHMU param.rw, {-(ySP).w*}	Change mode to user	0	0	0	0	
	Where y = MINU(x.PSL < current_mode <)						
06	LDPCTX {PCB.r*, -(KSP).w*}	Load process context (kernel mode only)	–	–	–	–	rsv, prv
DB	MFPR procreg.rl, dst.wl	Move from processor register (kernel mode only)	*	*	0	-	rsv, prv
DA	MTPR src.rl, procreg.rl	Move to processor register (kernel mode only)	*	*	0	–	rsv, prv
0C	PROBER mode.rb, len.rw, base.ab	Probe read access	0	*	0	–	
0D	PROBEW mode.rb, len.rw, base.ab	Probe write access	0	*	0	–	
02	REI {(SP)+.r*}	Return from exception or interrupt	*	*	*	*	rsv
07	SVPCTX {(SP)+.r*, PCB.w*}	Save process context (kernel mode only)	–	–	–	–	prv

• Microcode-assisted Emulated Instructions

The CVAX 78034 provides microcode assistance for the emulation of these instructions by system software. The processor processes the operand specifiers, creates a standard argument list, and takes an emulated instruction fault.

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
20	ADDP4 addlen.rw, addaddr.ab, sumlen.rw, sumaddr.ab	Add packed 4-operand	*	*	*	0	rsv, dov
21	ADDP6 add1len.rw, add1addr.ab, add2len.rw, add2addr.ab, sumlen.rw, sumaddr.ab	Add packed 6-operand	*	*	*	0	rsv, dov
F8	ASHP cnt.rb, srclen.rw, srcaddr.ab, round.rb, dstlen.rw, dstaddr.ab	Arithmetic shift and round packed	*	*	*	0	rsv, dov
35	CMPP3 len.rw, src1addr.ab, src2addr.ab	Compare packed 3-operand	*	*	0	0	
37	CMPP4 src1len.rw, src1addr.ab, src2len.rw, src2add.ab	Compare packed 3-operand	*	*	0	*	
0B	CRC tbl.ab, inicrc.rl, strien.rw, stream.ab	Calculate cyclic redundancy check	*	*	0	0	
F9	CVTLP src.rl, dstlen.rw, dstaddr.ab	Convert long to packed	*	*	*	0	rsv, dov
36	CVTPL srclen.rw, srcaddr.ab, dst.wl	Convert packed to long	*	*	*	0	rsv, iov
08	CVTPS, srclen.rw, srcaddr.ab, dstlen.rw, dstaddr.ab	Convert packed to leading separate	*	*	*	0	rsv, dov
09	CVTSP, srclen.rw, srcaddr., dstlen.rw, dstaddr.ab	Convert leading separate to packed	*	*	*	0	rsv, dov

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
24	CVTPT srclen.rw, srcaddr.ab, tbladdr.ab, dstlen.rw, dstaddr.ab	Convert packed to trailing	*	*	*	0	rsv, dov
26	CVTTP srclen.rw, srcaddr.ab, tbladdr.ab, dstlen.rw, dstaddr.ab	Convert packed to trailing	*	*	*	0	rsv, dov
27	DIVP divrien.rw, divraddr.ab, divdien.rw, quolen.rw, quoaddr.ab	Divide packed	*	*	*	0	rsv, dov, ddvz
38	EDITPC srclen.rw, srcaddr.ab, pattern.ab, dstaddr.ab	Edit packed to character string	*	*	*	*	rsv, dov
39	MATCHC objlen.rw, objaddr.ab, srclen.rw, srcaddr.ab	Match characters	0	*	0	0	
34	MOVP len.rw, srcaddr.ab, dstaddr.ab	Move packed	*	*	0	0	
2E	MOVTC srclen.rw, srcaddr.ab, fill.rb, tbladdr.ab, dstlen.rw, dstaddr.ab	Move translated characters	*	*	0	*	
2F	MOVTUC srclen.rw, srcaddr.ab, esc.rb, tbladdr.ab, dstlen.rw, dstaddr.ab	Move translated until character	*	*	*	*	
25	MULP mulrien.rw, mulraddr.ab, muldlen.rw, muldaddr.ab, prodlen.rw, prodaddr.ab	Multiply packed	*	*	*	0	rsv, dov
22	SUBP4 sublen.rw, subaddr.ab, diflen.rw, difaddr.ab	Subtract packed 4-operand	*	*	*	0	rsv, dov
23	SUBP6 sublen.rw, subaddr.ab, minlen.rw, minaddr.ab, diflen.rw, difaddr.ab	Subtract packed 6-operand	*	*	*	0	rsv, dov

• Floating-point Instructions

These instructions are implemented in hardware only if the optional CVAX 78134 Floating-point accelerator is present in the system. They must be software emulated if the CVAX 78134 is not included.

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
06F	ACBD limit.rd, add.rd, index.md	Add compare and branch D_floating	*	*	0	–	rsv, fov, fuv
04F	ACBF limit.rf, add.rf, index.rf	Add compare and branch F_floating	*	*	0	–	rsv, fov, fuv
4FFD	ACBG limit.rg, add.rg, index.mg	Add compare and branch G_floating	*	*	0	–	rsv, fov, fuv
060	ADDD2 add.rd, sum.md	Add D_floating 2-operand	*	*	0	0	rsv, fov, fuv
040	ADDF2 add.rf, sum.mf	Add F_floating 2-operand	*	*	0	0	rsv, fov, fuv
40FD	ADDG2 add.rg, sum.mg	Add G_floating 2-operand	*	*	0	0	rsv, fov, fuv
061	ADDD3 add1.rd, add2.rd, sum.wd	Add D_floating 3-operand	*	*	*	0	rsv, fov, fuv
041	ADDF3 add1.rf, add2.rf, sum.wf	Add F_floating 3-operand	*	*	*	0	rsv, fov, fuv
41FD	ADDG3 add1.rg, add2.rg, sum.wg	Add G_floating 3-operand	*	*	*	0	rsv, fov, fuv

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
071	CMPD src1.rd, src2.rd	Compare D_floating	*	*	0	0	rsv
051	CMPF src1.rf, src2.rf	Compare F_floating	*	*	0	0	rsv
51FD	CMPG src1.rg, src2.rg	Compare G_floating	*	*	0	0	rsv
06C	CVTBD src.rb, dst.wd	Convert byte to D_floating	*	*	0	0	
04C	CVTBF src.rb, dst.wf	Convert byte to F_floating	*	*	0	0	
4CFD	CVTBG src.rb, dst.wg	Convert byte to G_floating	*	*	0	0	
068	CVTDB src.rb, dst.wb	Convert D_floating to byte	*	*	*	0	rsv, iov
076	CVTDF src.rd, dst.wf	Convert D_floating to F_float	*	*	0	0	rsv, fov
06A	CVTDL src.rd, dst.wl	Convert D_floating to long	*	*	*	0	rsv, iov
069	CVTDW src.rd, dst.ww	Convert D_floating to word	*	*	*	0	rsv, iov
048	CVTFB src.rf, dst.wb	Convert F_floating to byte	*	*	*	0	rsv, iov
056	CVTFD src.rf, dst.wg	Convert F_floating to D_float	*	*	0	0	rsv
99FD	CVTFG src.rf, dst.wg	Convert F_floating to G_float	*	*	0	0	rsv
04A	CVTFL src.rf, dst.wl	Convert F_floating to long	*	*	*	0	rsv, iov
049	CVTFW src.rf, dst.ww	Convert F_floating to word	*	*	*	0	rsv, iov
48FD	CVTGB src.rg, dst.wb	Convert G_floating to byte	*	*	*	0	rsv, iov
33FD	CVTGF src.rg, dst.wf	Convert G_floating to F_float	*	*	0	0	rsv, fov, fuv
4AFD	CVTGL src.rg, dst.wl	Convert G_floating to long	*	*	*	0	rsv, iov
49FD	CVTGW src.rg, dst.ww	Convert G_floating to word	*	*	*	0	rsv, iov
06E	CVTLD src.rl, dst.wb	Convert long to D_floating	*	*	0	0	
04E	CVTLF src.rl, dst.wf	Convert long to F_floating	*	*	0	0	
4EFD	CVTLG src.rl, dst.wg	Convert long to G_floating	*	*	0	0	
06D	CVTWD src.rw, dst.wd	Convert word to D_floating	*	*	0	0	
04D	CVTWF src.rw, dst.wf	Convert word to F_floating	*	*	0	0	
4DFD	CVTWG src.rw, dst.wg	Convert word to G_floating	*	*	0	0	
06B	CVTRDL src.rd, dst.wl	Convert rounded D_floating to long	*	*	0	0	rsv, iov
04B	CVTRFL src.rf, dst.wl	Convert rounded F_floating to long	*	*	*	0	rsv, iov
4BFD	CVTRGL src.rg, dst.wl	Convert rounded G_floating to long	*	*	*	0	rsv, iov
066	DIVD2 divr.rd, quo.md	Divide D_floating 2-operand	*	*	0	0	rsv fov fuv fdvz
046	DIVF2 divr.rf, quo.mf	Divide F_floating 2-operand	*	*	0	0	rsv fov fuv fdvz
46FD	DIVG2 divr.rg, quo.mg	Divide G_floating 2-operand	*	*	0	0	rsv fov fuv fdvz
067	DIVD3 divr.rd, divr.rd, quo.wd	Divide D_floating 3-operand	*	*	0	0	rsv fov fuv fdvz
047	DIVD3 divr.rf, divr.rf, quo.wf	Divide F_floating 3-operand	*	*	0	0	rsv fov fuv fdvz
47FD	DIVD3 divr.rg, divr.rg, quo.wg	Divide G_floating 3-operand	*	*	0	0	rsv fov fuv fdvz
074	EMODD muir.rd, mulrx.rd, muld.rd, int.wl, fract.wd	Extended modulus D_floating	*	*	*	0	rsv fov fuv iov
054	EMODF muir.rf, mulrx.rb, muld.rd int.wl, fract.wf	Extended modulus F_floating	*	*	*	0	rsv fov fuv iov
54FD	EMODG muir.rg, mulrx.rw, muld.rg int.wl, fract.wg	Extended modulus G_floating	*	*	*	0	rsv fov fuv iov
072	*MNEGD src.rd, dst.wd	Move negated D_floating	*	*	0	0	rsv
052	*MNEGF src.rf, dst.wf	Move negated F_floating	*	*	0	0	rsv
52FD	*MNEGG src.rg, dst.wg	Move negated G_floating	*	*	0	0	rsv
070	*MOVD src.rd, dst.wd	Move D_floating	*	*	0	–	rsv
050	*MOVF src.rf, dst.wf	Move F_floating	*	*	0	–	rsv
50FD	*MOVG src.rg, dst.wg	Move G_floating	*	*	0	–	rsv

OP	Mnemonic and Arguments	Description	N	Z	V	C	Exceptions
064	MULD2 mulr.rd, prod.md	Multiply D_floating 2-operand	*	*	0	0	rsv, fov, fuv
044	MULF2 mulr.rf, prod.mf	Multiply F_floating 2-operand	*	*	0	0	rsv, fov, fuv
44FD	MULG2 mulr.rg, prod.mg	Multiply G_floating 2-operand	*	*	0	0	rsv, fov, fuv
065	MULD3 mulr.rd, muld.rd, prod.wd	Multiply D_floating 3-operand	*	*	0	0	rsv, fov, fuv
045	MULF3 mulr.rf, muld.rf, prod.wf	Multiply F_floating 3-operand	*	*	0	0	rsv, fov, fuv
45FD	MULG3 mulr.rf, muld.rg, prod.wg	Multiply G_floating 3-operand	*	*	0	0	rsv, fov, fuv
075	POLYD arg.rd, degree rw, tbladder.ab	Evaluate polynomial D_floating	*	*	0	0	rsv, fov, fuv
055	POLYF arg.rf, degree rw, tbladder.ab	Evaluate polynomial F_floating	*	*	0	0	rsv, fov, fuv
55FD	POLYD arg.rg, degree rw, tbladder.ab	Evaluate polynomial G_floating	*	*	0	0	rsv, fov, fuv
062	SUBD2 sub.rd, dif.md	Subtract D_floating 2-operand	*	*	0	0	rsv, fov, fuv
042	SUBF2 sub.rf, dif.mf	Subtract F_floating 2-operand	*	*	0	0	rsv, fov, fuv
42FD	SUBG2 sub.rg, dif.mg	Subtract G_floating 2-operand	*	*	0	0	rsv, fov, fuv
063	SUBD3 sub.rd, min rd, dif.md	Subtract D_floating 3-operand	*	*	*	0	rsv fov fuv
043	SUBF2 sub.rf, min rf, dif.mf	Subtract F_floating 3-operand	*	*	0	0	rsv fov fuv
43FD	SUBG2 sub.rg, min rg, dif.mg	Subtract G_floating 3-operand	*	*	0	0	rsv fov fuv
073	*TSTD src.rd	Test D_floating	*	*	0	0	rsv
053	*TSTF src.rf	Test F_floating	*	*	0	0	rsv
53FD	*TSTG src.rg	Test G_floating	*	*	0	0	rsv

• Memory Management

The memory management unit of the CVAX 78034 provides a flexible and efficient virtual memory programming environment. Memory management, together with the operating system, provides both paging (with user control) and swapping. It also provides four hierarchical modes: kernel, executive, supervisor, and user, and has read and write access control for each mode.

A virtual memory system provides a large address space while allowing programs to run on hardware with small memory configurations. Programs execute in an environment defined as a process. The virtual memory system for the CVAX 78034 provides each process with a 4 billion byte address space.

Virtual Address Space

Virtual address space consists of two address spaces of equal size—a system space and a process space. The process space contains the P0 and P1 regions. Figure 15 shows the virtual address space assignments.

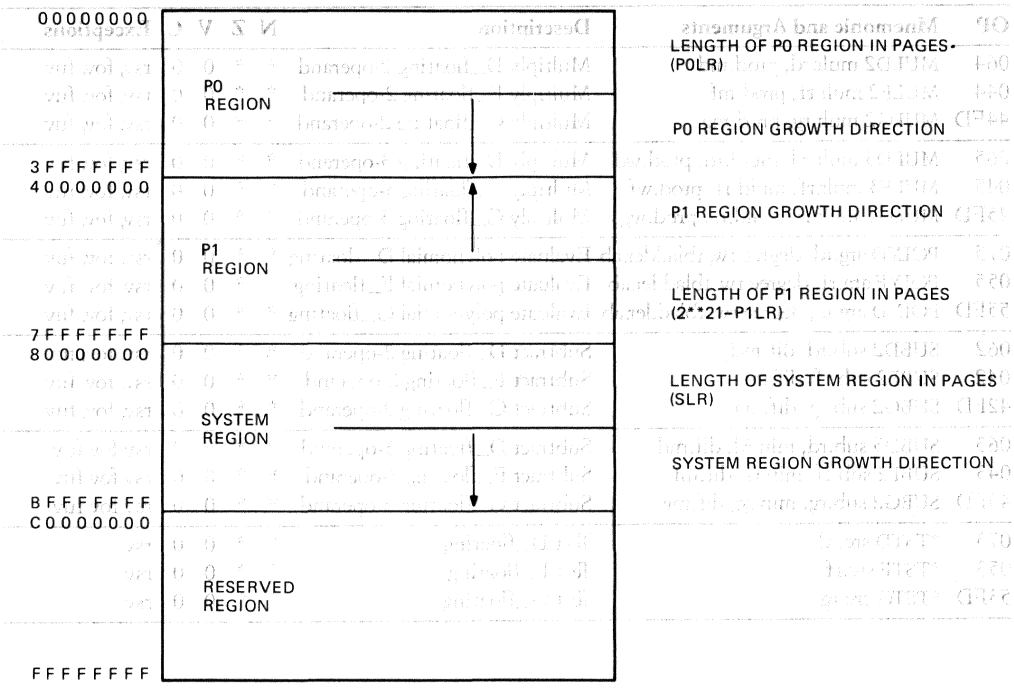


Figure 15 • CVAX 78034 Virtual Address Space Assignments

Virtual address format—The CVAX 78034 generates a 32-bit virtual address for each instruction and operand in memory. As the process is executed, the processor translates each virtual address into a physical address. The format of a virtual address is shown in Figure 16. Table 11 defines the fields of a virtual address.

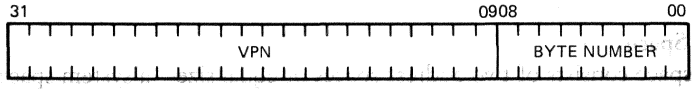


Figure 16 • CVAX 78034 Virtual Address Format

Table 11 • CVAX 78034 Virtual Address Description

Bit	Description
31:09	VPN (Virtual page number)—This field specifies the virtual page to be referenced. Virtual address space contains 8,388,608 pages of 512 bytes each. Bits 31:30 of the VPN are used to select the region of virtual address space being referenced as follows. **Bits 31:30 value** — **Region** 0 — P0 1 — P1 2 — system 3 — reserved
08:00	Byte Number—This field specifies the byte number within the page.

Page protection—Independent of its location in virtual address space, a page (512 bytes) can be protected according to its use. Although the system space is shared and a program can generate any address, the program can be prevented from modifying or accessing portions of the system space. A program can also be prevented from accessing or modifying portions of process space.

Virtual address space allocations—Access to the P0, P1, and System region is controlled by a length register. The P0 region is controlled by the P0 Length Register (P0LR), the P1 region by the P1 Length Register (P1LR), and the system region by the System Length Register (SLR). Within the limits defined by the length registers, the access is controlled by a page table that specifies the validity, access requirements, and location of each page in the region.

Access control

The access control function validates the type of memory access that is allowed to access a page. Each page has a protection code for each mode that determines if read or write references are allowed.

Four hierarchical modes are used by the CVAX 78034. The processor mode that is currently running is stored in the current mode field of the Processor Status Longword (PSL). The modes in order of most to least privileged are

- 0 Kernel—Used by the kernel of the operating system for page management, scheduling, and I/O drivers.
- 1 Executive—Used for many of the operating system service calls.
- 2 Supervisor—Used for services such as command interpretation.
- 3 User—Used for user-level code, utilities, compilers, debuggers, etc.

The protection code, located in the page-table entry for that page, specifies whether the page can be accessed for each mode. These codes are described in Table 12.

Table 12 • CVAX 78034 Protection Codes Assignments

Code decimal	binary	Mnemonic	Current mode[1] K	E	S	U	Comment
0	0000	NA	—	—	—	—	no access
1	0001		**	**	**	**	reserved
2	0010	KW	RW	—	—	—	
3	0011	KR	R	—	—	—	
4	0100	UW	RW	RW	RW	RW	all access
5	0101	EW	RW	RW	—	—	
6	0110	ERKW	RW	R	—	—	
7	0111	ER	R	R	—	—	
8	1000	SW	RW	RW	RW	—	
9	1001	SREW	RW	RW	R	—	
10	1010	SRKW	RW	R	R	—	
11	1011	SR	R	R	R	—	
12	1100	URSW	RW	RW	RW	R	
13	1101	UREW	RW	RW	R	R	
14	1110	URKW	RW	R	R	R	
15	1111	UR	R	R	R	R	

[1] — = no access
** = unpredictable
R = read only
RW = read/write
K = Kernel
E = Executive
S = Supervisor
U = User

Memory-management Control

The three registers used to control the memory management function are described as follows:

Map Enable register (MAPEN)—This register is used to enable and disable memory management. The format of the register is shown in Figure 17 and described in Table 13.

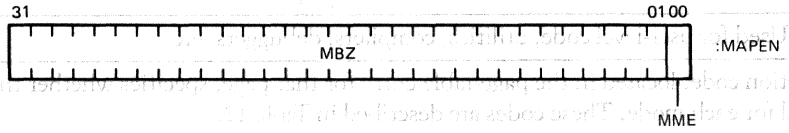


Figure 17 • CVAX 78034 Map Enable Register Format

Table 13 • CVAX 78034 Map Enable Register Description

Bit	Description
31:01	MBZ—Must be zero
00	MME (Memory management enable)—Enable and disable memory management as follows: MME = 1 (enabled) MME = 0 (disabled)

Translation buffer—This buffer is used to save the actual memory references when pages are repeatedly referenced. The CVAX CPU uses this buffer to record successful virtual address translations and page status. The translation buffer contains 28 fully associative entries. Both system space and process space references share the entries. Translation buffer entries are replaced using a Not Last Used (NLU) algorithm to ensure that the replacement pointer is not pointing to the last translation buffer entry to be used. This is accomplished by rotating the replacement pointer to the next sequential translation buffer entry if the pointer is pointing to an entry that has just been accessed. Both D-stream and I-stream references can cause the NLU to cycle. When the translation buffer does not contain a virtual address and page status of the memory referenced, the CPU updates the translation buffer. The entry to be replaced is pointed to by the replacement pointer.

System control of the translation buffer is through the Translation Buffer Invalidate Single (TBIS) register and the Translation Buffer Invalidate All (TBIA) register.

The TBIS register is used to invalidate single PTE entries in the translation buffer. This is accomplished by writing a virtual address into the TBIS register that invalidates any translation buffer entry that maps the virtual address. Figure 18 shows the register format.

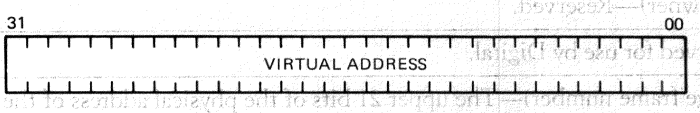


Figure 18 • CVAX CVAX 78034 Translation Buffer Invalidate Single Register Format

The TBIA register is used to clear the translation buffer by invalidating all page table entries in the translation buffer. This is performed by writing a 0 into the TBIA register. Figure 19 shows the format of the register.

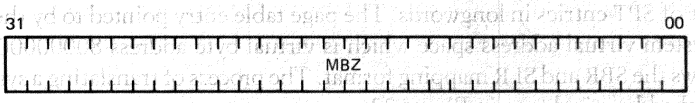


Figure 19 • CVAX 78034 Translation Buffer Invalidate All Register

Address Translation

The translation of a virtual address to a physical address by the memory management unit is controlled by the Memory Management Enable bit 00 of the MAPEN register. When MME is cleared, memory mapping is disabled and the low-order bits of the virtual address bits 29:00 are the physical address. When MME is set, memory mapping is enabled and the virtual address is mapped to a physical address by memory management.

All virtual addresses are translated to physical addresses by a page table entry (PTE). The PTE has a valid bit that controls only the validity of the modify bit and page frame number field. The protection field is always valid and is checked first. The page table entry is shown in Figure 20 and described in Table 14.

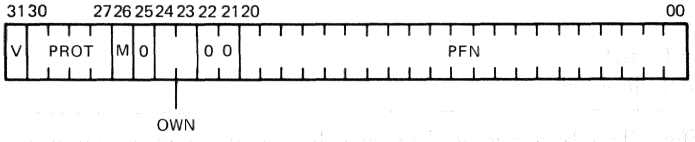


Figure 20 • CVAX 78034 Page Table Entry Format

Table 14 • CVAX 78034 Page Table Entry Description

Bit	Description
31	V (Valid bit)—Determines the validity of the modify bit 26 and the page frame number field bits 20:00. V is set for valid and cleared for not valid.
30:27	PROT (Protection)—Defines the protection for the page. This field is always valid and is used by the hardware even when V bit 31 is cleared.
26	M (Modify bit)—This bit is set 1 if the page has already been recorded as modified. If M is cleared, the page has not been recorded as modified. Used only if V is set.
25	0—Reserved for used by Digital.
24:23	OWN (Owner)—Reserved.
22:21	0—Reserved for use by Digital.
20:00	PFN (Page frame number)—The upper 21 bits of the physical address of the base of the page. Used only if V is set.

System Space Address Translation

A virtual address with bits 31:30 equal to 2 is identified as an address in the system virtual address space that is mapped by the System Page Table (SPT) in physical memory. The System Base Register (SBR) contains the physical address of the SPT and the System Length Register (SLR) defines the number of SPT entries in longwords. The page table entry pointed to by the SBR maps the first page of system virtual address space which is virtual byte address 80000000 (hexadecimal). Figure 21 shows the SBR and SLR mapping format. The process of translating a system virtual address to a physical address is shown in Figure 22.

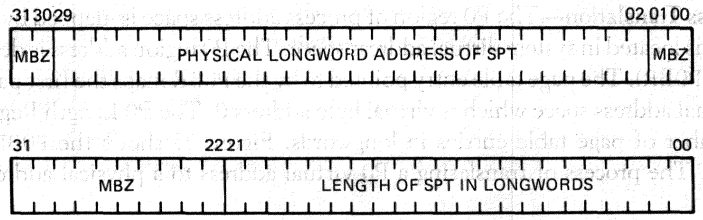


Figure 21 • CVAX 78034 System Mapping Registers Format

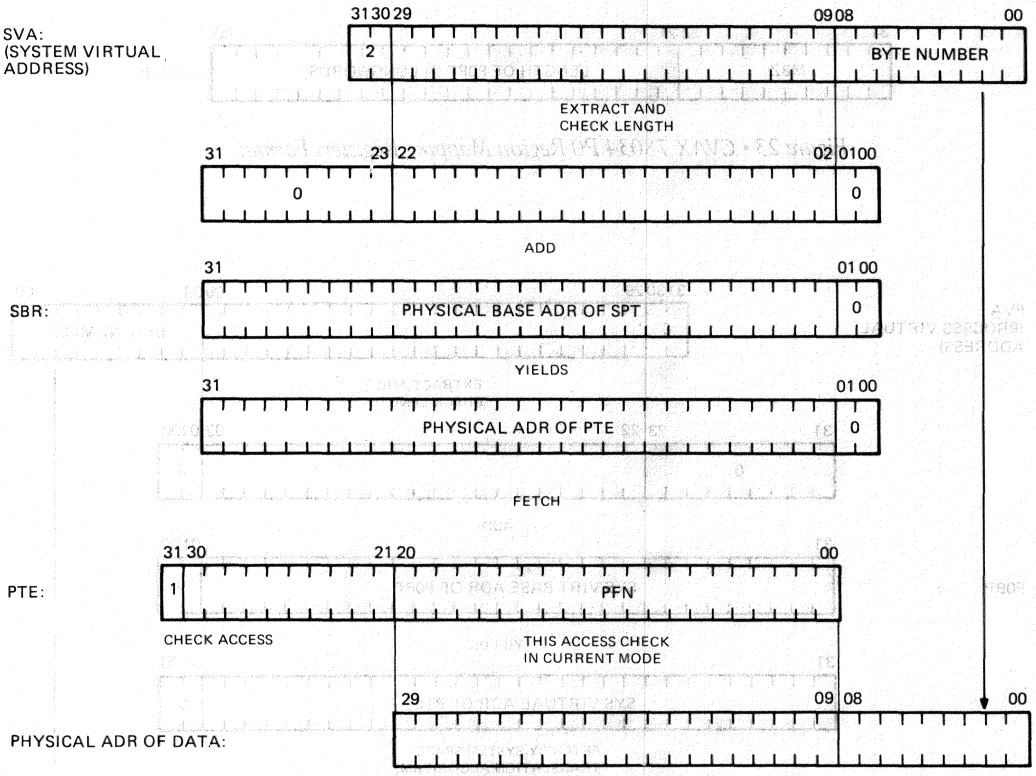


Figure 22 • CVAX 78034 System Virtual-to-physical Address Translation

Process Space Address Translation

A virtual address with bit 31 equal to 0 is identified as an address in the process virtual address space. Process space is divided into two equal sized separately mapped regions. When bit 30 is equal to 0, the address is in region P0. When bit 30 is equal to 1, the address is in region P1.

P0 Region Address Translation—The P0 region of process address space is mapped by the P0 Page Table (P0PT) that is located in system virtual address space. The P0 region address is defined by the P0 Base Register (P0BR). The page table entry pointed to by the P0BR maps the first page of the P0 region of the virtual address space which is virtual byte address 0. The P0 Length Register (P0LR) contains the number of page table entries in longwords. Figure 23 shows the P0BR and P0LR mapping formats. The process of translating a P0 virtual address to a physical address is shown in Figure 24.

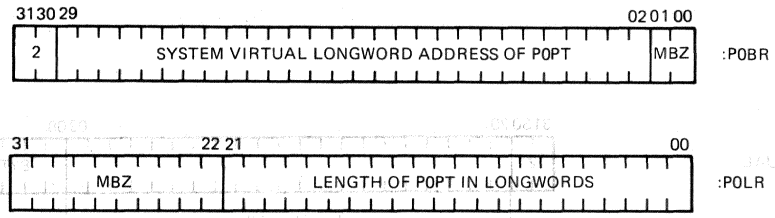


Figure 23 • CVAX 78034 P0 Region Mapping Registers Format

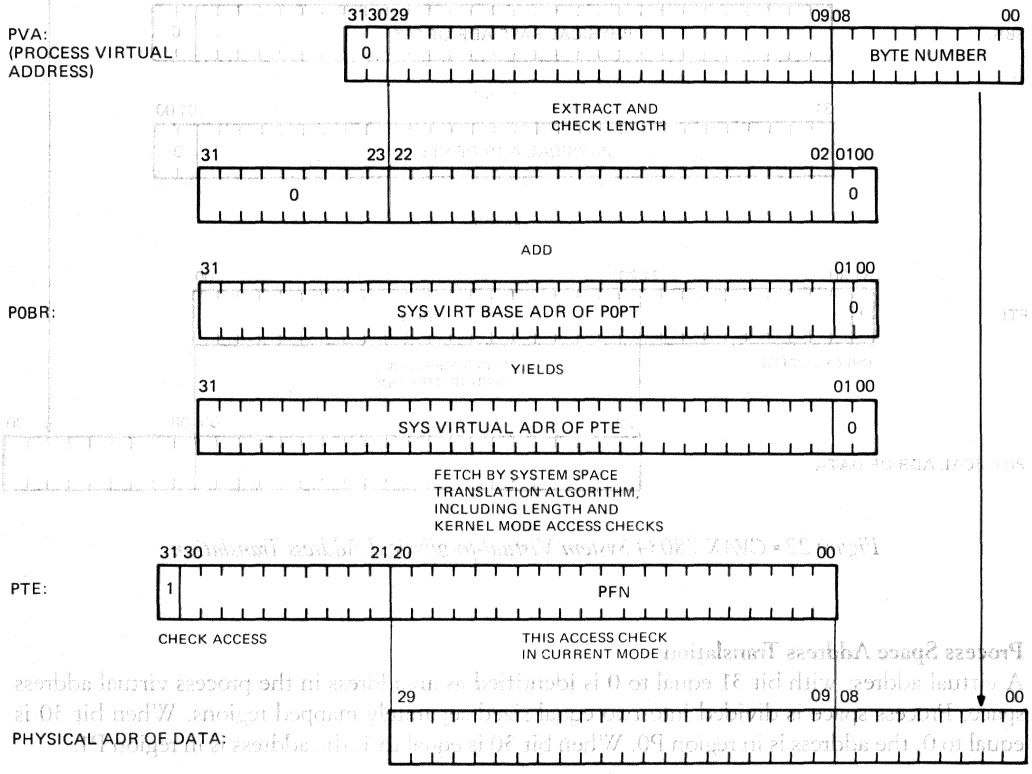


Figure 24 • CVAX 78034 P0 Virtual-to-physical Address Translation

P1 Region Address Translation—The P1 region of the address space is mapped by the P1 Page Table (P1PT) that is located in system virtual address space. The P1 region is defined by the P1 Base Register (P1BR) and the P1 Length Register (P1LR). Because the P1 space advances toward smaller addresses and because a consistent hardware interpretation of the base and length registers is not desirable, P1BR and P1LR define the portion of P1 space that is not accessible. The P1LR contains the number of nonexistent PTEs. P1BR contains the system virtual address of what would be the PTE for the first page of P1 which is virtual byte address 40000000 (hexadecimal). The address in P1BR may not be a valid system virtual address but all addresses of PTEs must be valid system virtual addresses. The P1BR and P1LR mapping format is shown in Figure 25. The process of virtual address to physical address translation is shown in Figure 26.

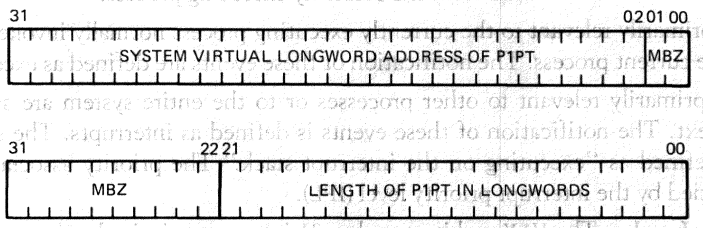


Figure 25 • CVAX 78034 P1 Region Mapping Registers Format

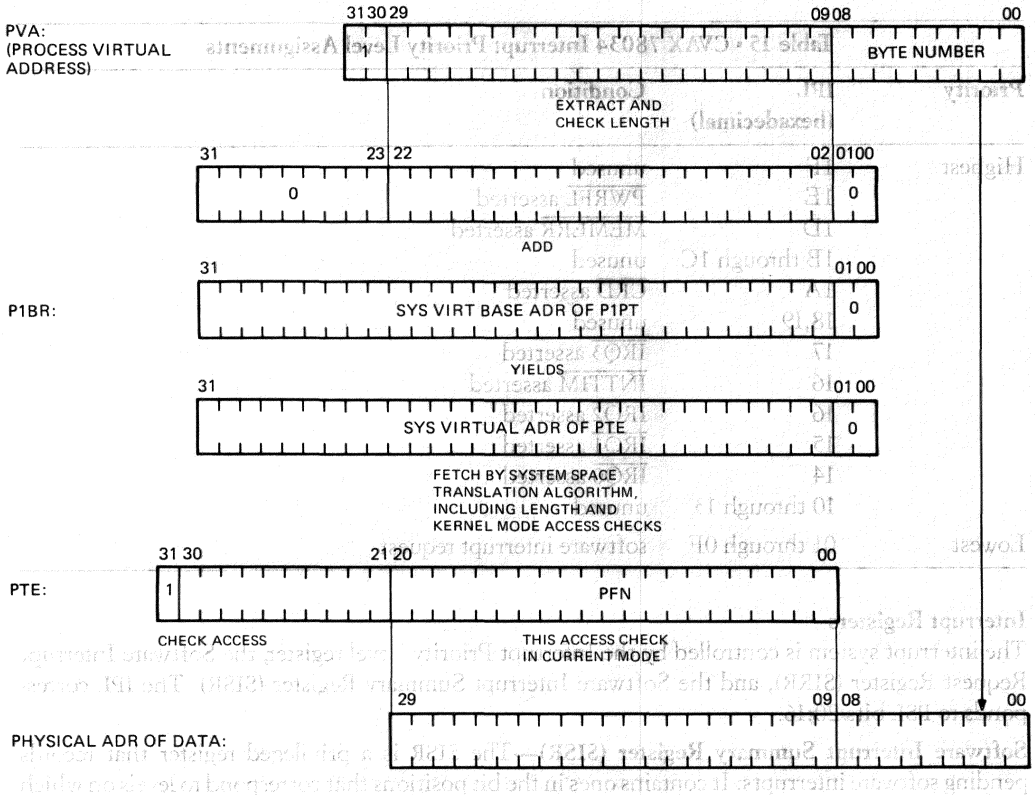


Figure 26 • CVAX 78034 P1 Virtual-to-physical Address Translation

Memory Management Faults

The two types of faults associated with memory mapping and protection are Translation Not Valid (TNV) and Access Control Violation (ACV). An ACV fault exists when the protection field of the page table entry (PTE) indicates that the intended page reference in the specified access mode is illegal. A TNV fault exists when a read or write reference is attempted through an invalid PTE (PTE bit 31 is set). The ACV fault takes precedence when both an ACV and TNV fault occurs.

• Exceptions and Interrupts

During the operation of a system, events within the system may occur that require the execution of software beyond the software required for normal control. The processor transfers control by forcing a change in the flow of control from the currently executing process.

Events that are primarily relevant to the currently executing process normally invoke software in the context of the current process. The notification of these events are defined as exceptions.

Events that are primarily relevant to other processes or to the entire system are serviced in a systemwide context. The notification of these events is defined as interrupts. The system wide context is also defined as "executing on the interrupt stack." The priority associated with an interrupt is specified by the interrupt priority level (IPL).

Interrupt Priority Levels—The VAX architecture has 31 interrupt priority levels grouped into 15 software levels (1 to F hexadecimal) and 16 hardware levels (10 to 1F hexadecimal). Table 15 lists the CVAX 78034 IPL, priority, and the conditions causing the interrupt.

Table 15 • CVAX 78034 Interrupt Priority Level Assignments

Priority	IPL (hexadecimal)	Condition
Highest	1F	unused
	1E	$\overline{\text{PWRFL}}$ asserted
	1D	$\overline{\text{MEMERR}}$ asserted
	1B through 1C	unused
	1A	$\overline{\text{CRD}}$ asserted
	18,19	unused
	17	$\overline{\text{IRQ3}}$ asserted
	16	$\overline{\text{INTTIM}}$ asserted
	16	$\overline{\text{IRQ2}}$ asserted
	15	$\overline{\text{IRQ1}}$ asserted
	14	$\overline{\text{IRQ0}}$ asserted
	10 through 13	unused
Lowest	01 through 0F	software interrupt request

Interrupt Registers

The interrupt system is controlled by the Interrupt Priority Level register, the Software Interrupt Request Register (SIRR), and the Software Interrupt Summary Register (SISR). The IPL corresponds to PSL bits 20:16.

Software Interrupt Summary Register (SISR)—The SISR is a privileged register that records pending software interrupts. It contains ones in the bit positions that correspond to levels on which software interrupts are pending. Figure 27 shows the SISR format.

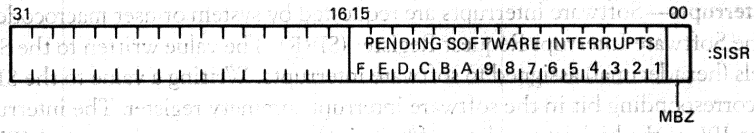


Figure 27 • CVAX 78034 Software Interrupt Summary Register Format

Software Interrupt Request Register (SIRR)—The SIRR is a write-only 4-bit privileged register used for initiating a software request. The software requests an interrupt by writing the appropriate level to the SIRR. Once a software request is made, the corresponding bit in the SISR is set. The processor will clear the bit in the SISR when the interrupt has been taken. Figure 28 shows the SIRR format.

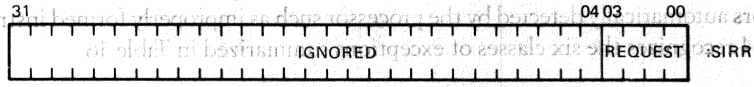


Figure 28 • CVAX 78034 Software Interrupt Request Register Format

Interrupt Priority Level Register (IPL)—Writing to the IPL register loads the processor priority field in the processor status longword. Figure 29 shows the IPL register format.

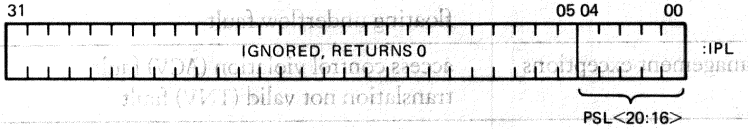


Figure 29 • CVAX 78034 Interrupt Priority Level Register Format

Interrupts

Hardware and software interrupts are initiated by the following conditions.

Hardware interrupts—Hardware interrupts are initiated by $\overline{\text{HALT}}$ (nonmaskable interrupt), $\overline{\text{PWRFL}}$, $\overline{\text{MEMERR}}$, $\overline{\text{CRD}}$, $\overline{\text{INTTIM}}$, and $\overline{\text{IRQ}}<3:0>$ signals. These signals are sampled once each microcycle by the interrupt controller of the CPU. The interrupt controller compares the IPL associated with any signal that is asserted to the current IPL of the CPU. If any of the asserted signals have an IPL higher than the CPU, an interrupt will be taken. For interrupts generated by the $\overline{\text{HALT}}$, $\overline{\text{PWRFL}}$, $\overline{\text{MEMERR}}$, $\overline{\text{CRD}}$, and $\overline{\text{INTTIM}}$, the CPU internally generates a vector which is an offset into the SCB. For interrupts generated by $\overline{\text{IRQ}}<3:0>$, the CPU executes an interrupt acknowledge cycle to fetch the vector from the device requesting the interrupt.

An interrupt is serviced at its priority level except for interrupts requested by $\overline{\text{IRQ}}<3:0>$ that are serviced at either their associated IPL or IPL 17 (hexadecimal). The level at which an interrupt requested by an $\overline{\text{IRQ}}3:0>$ is serviced is determined by DAL00 when the device sends the vector to the CPU. When DAL00 is a 0, the interrupt is serviced at the IPL associated with the asserted signal. When DAL00 is a 1, the interrupt is serviced at IPL 17 (hexadecimal).

Software interrupts—Software interrupts are requested by system or user macrocode by writing a value into the Software Interrupt Request Register (SIRR). The value written to the SIRR is one of the IPL levels (hexadecimal) assigned to software interrupts. Writing a value to the SIRR results in setting the corresponding bit in the software interrupt summary register. The interrupt controller compares the IPL of the highest pending software interrupt request to the current IPL of the CPU. If no outstanding hardware interrupt exist and the IPL of the software interrupt is higher than the current IPL of the CPU, the interrupt will be granted. The CPU internally generates the interrupt vector in the SCB.

The software interrupt system is affected by an REI instruction or other event that changes the IPL of the CPU. If the IPL is changed to a value lower than the highest pending software interrupt request and no hardware interrupts are pending, the interrupt controller grants the software interrupt.

Exceptions

An exception is an event resulting from the execution of a specific instruction. Exceptions also include errors automatically detected by the processor such as improperly formed instructions. The CVAX 78034 recognizes the six classes of exceptions summarized in Table 16.

Table 16 • CVAX 78034 CPU Summary of Exceptions

Exception class	Cause
arithmetic traps/faults	integer overflow trap integer divide by zero trap subscript range trap floating overflow fault floating divide by zero fault floating underflow fault
memory management exceptions	access control violation (ACV) fault translation not valid (TNV) fault
operand reference exceptions	reserved addressing mode fault reserved operand fault or abort
instruction execution exceptions	reserved/privileged instruction fault emulated instruction fault customer reserved instruction fault breakpoint fault
tracing exception	trace fault
system failure exceptions	machine check abort including read/write bus and parity errors, cache parity errors, and FPA protocol errors kernel stack not valid abort interrupt stack not valid abort

System Control Block

The System Control Block (SCB) is a page aligned table in physical memory that contains the vectors for servicing interrupts and exceptions. Table 17 lists the system control block vectors. The SCB is pointed to by the system control block base register (SCBB). The register format is shown in Figure 30.

Table 17 • CVAX 78034 System Control Block Vectors

Vector (hexadecimal)	Name	Type
00	passive release	interrupt
04	machine check	abort
08	kernel stack not valid	abort
0C	powerfail	interrupt
10	reserved/privileged instruction	fault
14	ustomer reserved instruction	fault
18	reserved operand	fault/abort
1C	Creserved addressing mode	fault
20	access control violation	fault
24	translation not valid	fault
28	trace pending (TP)	fault
2C	breakpoint instruction	fault
30	unused	—
34	arithmetic	trap/fault
38–3C	unused	—
40	CHMK	trap
44	CHME	trap
48	CHMS	trap
4C	CHMU	trap
50	unused	—
54	corrected read data	interrupt
58–5C	unused	—
60	memory error	interrupt
64–80	unused	—
84	software level 1	interrupt
88	software level 2	interrupt
8C	software level 3	interrupt
90–BC	software levels 4 through 15	interrupt
C0	interval time	interrupt
C4	unused	—
C8	emulation start	fault
CC	emulation continue	fault
D0–FC	unused	—
100–1FC	adapter vectors	interrupt
200–FFFC	device vectors	interrupt

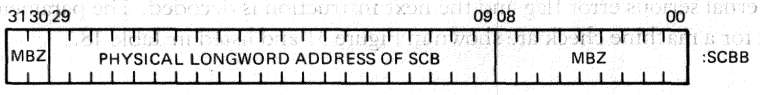


Figure 30 • CVAX 78034 System Control Block Base Register Format

SCB vectors from 100 through FFFC (hexadecimal) are used to directly vector interrupts from the external bus. The SCBB vector index is determined by bits 15:02 of the value supplied by external hardware. The new PSL priority level is determined either by the external interrupt request level that caused the interrupt or by bit 0 of the value supplied by external hardware. If bit 0 is cleared, the new IPL level is determined by the interrupt request level being serviced. If bit 0 is set, the new IPL is forced to 17 (hexadecimal). The ability to force the IPL to 17 supports an external bus, such as the Q-bus, that cannot guarantee that the device generating the SCBB vector index is the device that originally requested the interrupt. For example, the Q-bus has four separate interrupt request signals that correspond to $\overline{IRQ<3:0>}$ but only one interrupt grant is daisychained. Devices on the Q-bus are also arranged so that higher-priority devices are electrically closer to the bus master. If an $\overline{IRQ1}$ is being serviced, a device with a higher priority may intercept the grant. Software must determine the level of the device that was serviced and set the IPL to the correct value.

External devices, except devices that emulate the console storage and terminal hardware, should use only the vectors in the range of 100 to FFFC (hexadecimal).

Machine Check

A machine check occurs as a result of serious internal CPU errors or external CPU errors such as memory subsystem errors. These errors and conditions include

- FPA protocol errors.
- Impossible situations in memory management.
- Unused IPL requests.
- Impossible situations in the CPU microcode.
- Bus memory errors.
- Multiple errors.

Machine Check Processing—The CPU processes a machine check as follows:

- If an exception is in progress and a machine check occurs, a processor restart is executed by the CPU. Refer to the *Processor Restart* description that follows.
- If the current instruction can be suspended (MOVC3, MOVC5), the state of the processor should be saved and the machine check handled.
- If the instruction cannot be suspended, the state of the processor should be returned to the beginning of the instruction, if possible, and then the machine check should occur.

An instruction that cannot be restarted after the machine check is considered nonrecoverable and the current process or the operating system must be terminated.

When a machine check is generated, the CPU sets an internal serious error flag and performs machine check exception processing through SCB vector 4. A machine check exception is always processed on the interrupt stack. When machine check exception processing is complete, the CPU clears its internal serious error flag and the next instruction is decoded. The parameters recorded on the stack for a machine check are shown in Figure 31 and listed in Table 18.

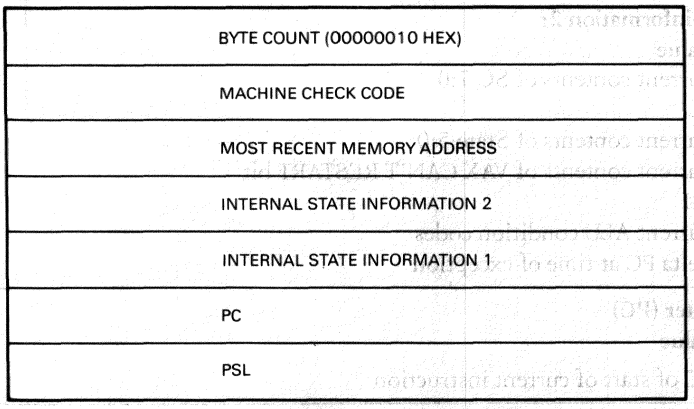


Figure 31 • CVAX 78034 Machine Check Stack

Table 18 • CVAX 78034 Machine Check Parameters

Machine check code (hexadecimal):

code	definition
1	FPA protocol error
2	FPA reserved instruction
3	FPA unknown error
4	FPA unknown error
5	process PTE in P0 space (TB miss)
6	process PTE in P1 space (TB miss)
7	process PTE in P0 space (M = 0)
8	process PTE in P1 space (M = 0)
9	undefined interrupt ID code
A	impossible microcode state (MOVCx)
80	read bus error, normal read
81	read bus error, SPTE, PCB, or SCB read
82	write bus error, normal write
83	write bus error, SPTE or PCB write

Most recent memory address:

address	value
31:00	current contents of VAP register

Internal state information 1:

bits	value
31:24	current contents of opcode 7:0
23:20	1111
19:16	current contents of HSIR 3:0
15:08	current contents of CADR 07:00
07:00	current contents of MSER 07:00

Internal state information 2:	
bits	**value**
31:24	current contents of SC 7:0
23:22	11
21:16	current contents of State 5:0
15	current contents of VAX CAN'T RESTART bit
14:12	111
11:08	current ALU condition codes
07:00	delta PC at time of exception

Program counter (PC)	
bits	**value**
31:00	PC of start of current instruction

Processor status longword (PSL)	
bits	**value**
31:00	current contents of PSL

Machine Check Errors

Machine check errors include protocol errors, memory management and microcode impossible situations, bus memory errors, and multiple errors.

Protocol—CVAX 78134 FPA checks for the proper order of requests from the CPU. If a protocol violation is detected, a machine check occurs. All FPA protocol error machine checks are nonrecoverable. The error should be logged and the currently running process or the operating system should be terminated. The hexadecimal codes generated for a FPA protocol error are

Code	Error
1	FPA protocol
2	FPA reserved instruction
3 and 4	FPA unknown

Impossible situations (memory management)—The CVAX CPU checks for some impossible conditions in the memory management unit. If an impossible situation is detected, a machine check occurs. All impossible memory management machine checks are nonrecoverable. The error should be logged and the currently running process or operating system should be terminated. The current memory management registers (P0BR, P1BR, SBR, P0LR, P1LR, and SLR) should also be logged. The hexadecimal codes generated are

Code	Machine check error
5	The calculated virtual address for a process PTE is in P0 space (TB miss flows)
6	The calculated virtual address for a Process PTE is in P1 space (TB miss flows)
7	The calculated virtual address for a Process PTE is in P0 space (M = 0 flows)
8	The calculated virtual address for a Process PTE is in P1 space (M = 0 flows)

Unused IPL request—The CVAX CPU uses 13 of the 16 hardware interrupt priority levels as defined in the VAX architecture. If an interrupt at an unused hardware IPL is requested, a hexadecimal code and machine check occurs. The unused IPL machine check is nonrecoverable. The error should be logged. A nonvectored interrupt representing a serious error (corrected read data, memory error, powerfail, or processor halt) has probably been lost. The operating system should be terminated. The hexadecimal code and error is

Code	Machine check error
9	The interrupt controller returned an interrupting IPL of 18, 19, or 1B (hexadecimal)

Impossible situations (microcode)—Because of size constraints, erroneous branches in microcode will usually result in the execution of random microinstructions. However, if the microcode detects an impossible situation, a machine check occurs. The impossible microcode machine check is nonrecoverable. The error should be logged, and the currently running process or the operating system should be terminated. The following hexadecimal code and error is generated.

Code	Machine check error
A	MOVC3 or MOVC5 in impossible state

Bus memory errors—If external logic asserts ERR in response to a memory cycle other than an instruction prefetch or interrupt acknowledge, a machine check occurs and the following hexadecimal code is generated.

Code	Machine check error
80	read bus error, normal read
81	read bus error, SPTE, PCB, or SCB read
82	rite bus error, normal write
83	write bus error, SPTE or PCB write

Bus memory error machine checks may be recoverable depending on the error code, the VAX Can't Restart flag, and FPD flags in the machine check stack frame. Bus memory error machine checks that are recognized by the CPU as restartable may be nonrecoverable for system reasons (e.g., a read lock may be outstanding). On a nonrecoverable error, the error should be logged, and the currently running process or the operating system should be terminated. The code and relationship is

Code	VAX can't restart*	FPD*	Action
80,81	0	X	restartable
	1	0	nonrecoverable
	1	1	restartable
82,83	X	X	nonrecoverable

*X is either 1 or 0.

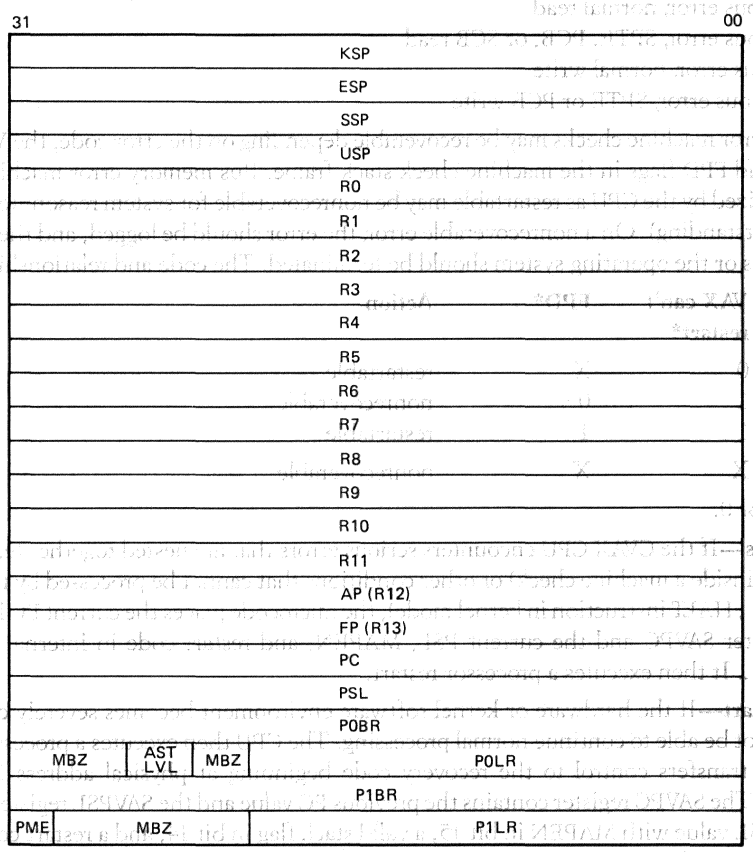
Multiple Errors—If the CVAX CPU encounters serious errors that are nested together (e.g., kernel stack not valid inside a machine check) or other conditions that cannot be processed by the system macrocode (e.g., HALT instruction in kernel mode), the microcode places the current PC in internal processor register SAVPC and the current PSL, MAPEN, and restart code in internal processor register SAVPSL. It then executes a processor restart.

Processor Restart—If the hardware or kernel software environment becomes severely corrupted, the CPU may not be able to continue normal processing. The CPU then executes a processor restart operation and transfers control to the recovery code beginning at physical address 20040000 (hexadecimal). The SAVPC register contains the previous PC value and the SAVPSL register contains the previous PSL value with MAPEN in bit 15, a valid stack flag in bit 14, and a restart code in bits 13:08. The restart codes are summarized in Table 9. The state of the CPU for a processor restart is as follows. All other registers are not defined.

Register	Condition		
SAVPC	saved PC		
SAVPSL	saved PSL bits 31:16. 07:00 in bits 31:16 and 07:00, saved MAPEN0 in bit 15, valid stack flag in bit 14, and saved restart code in bits 13:08		
SP	interrupt stack pointer	ASTLVL	4 (powerup only)
PSL	041F0000 (hexadecimal)	ICCS	cleared (powerup only)
PC	20040000 (hexadecimal)	MSER	cleared (powerup only)
MAPEN	cleared	CADR	cleared (powerup only)
SISR	cleared (powerup only)		

Process Structure

A process is a single thread of execution. The context of the current process is contained in the Process Control Block (PCB). The PCB, as defined by the CVAX 78034 CPU, is shown in Figure 32. The PCB is located in physical memory and is pointed to by the Process Control Block Base register (PCBB) shown in Figure 33.



NOTE: THE PME FIELD IS UNUSED.

Figure 32 ▪ CVAX 78034 Process Control Block Format

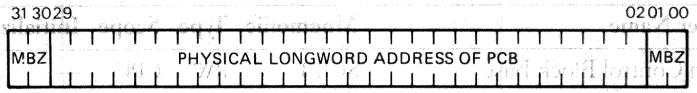


Figure 33 • CVAX 78034 Process Control Block Base Register

Processor Registers

The VAX architecture defines the Internal Processor Registers (IPRs). Some of these registers are implemented in the CVAX 78034 CPU and some can be implemented in external logic and accessed by the CVAX CPU. These registers are explicitly accessed by the Move To Processor Register (MTPR) and Move From Processor Register (MFPR) instructions. Table 19 lists the internal processor registers and their categories that are defined as follows:

1 Implemented by the CVAX CPU as specified in the VAX Architecture Standard (DEC Standard 032).

2 Implemented only by the CVAX CPU.

3 Passed to external logic via an external processor register cycle. If not externally implemented, they are read as zero and perform no function during write operations.

4 Access not allowed (reserved operand fault).

Table 19 • CVAX 78034 Internal Processor Registers

Number	Register Name	Mnemonic	Type	Scope	Initialize	Category*
0	Kernel Stack Pointer	KSP	RW	PROC	—	1
1	Executive Stack Pointer	ESP	RW	PROC	—	1
2	Supervisor Stack Pointer	SSP	RW	PROC	—	1
3	User Stack Pointer	USP	RW	PROC	—	1
4	Interrupt Stack Pointer	ISP	RW	CPU	—	1
5	reserved	—	—	—	—	4
6	reserved	—	—	—	—	4
7	reserved	—	—	—	—	4
8	P0 Base Register	P0BR	RW	PROC	—	1
9	P0 Length Register	P0LR	RW	PROC	—	1
10	P1 Base Register	P1BR	RW	PROC	—	1
11	P1 Length Register	P1LR	RW	PROC	—	1
12	System Base Register	SBR	RW	CPU	—	1
13	System Length Register	SLR	RW	CPU	—	1
14	reserved	—	—	—	—	4
15	reserved	—	—	—	—	4
16	Process Control Block Base	PCBB	RW	PROC	—	1

*Refer to Processor Register description.

Number	Register Name	Mnemonic	Type	Scope	Initialize	Category*
17	System Control Block Base	SCBB	RW	CPU	—	1
18	Interrupt Priority Level	IPL	RW	CPU	yes	1
19	AST Level	ASTLVL	RW	PROC	yes	1
20	Software Interrupt Request	SIRR	W	CPU	—	1
21	Software Interrupt Summary	SISR	RW	CPU	yes	1
22	Interprocessor Interrupt	IPIR	RW	CPU	—	4
23	CMI Error Register	CMIERR	R	CPU	—	4
24	Interval Clock Control	ICCS	RW	CPU	yes	2
25	Next Interval Count	NICR	W	CPU	—	3
26	Interval Count	ICR	R	CPU	—	3
27	Time Of Year	TODR	RW	CPU	—	3
28	Console Storage Receiver Status	CSRS	RW	CPU	—	3
29	Console Storage Receiver Data	CSRD	R	CPU	—	3
30	Console Storage Transmitter Status	CSTS	RW	CPU	—	3
31	Console Storage Transmitter Data	CSTD	W	CPU	—	3
32	Console Receiver Status	RXCS	RW	CPU	—	3
33	Console Receiver Data	RXDB	R	CPU	—	3
34	Console Transmitter Status	TXCS	RW	CPU	—	3
35	Console Transmitter Data	TXDB	W	CPU	—	3
36	Translation Buffer Disable	TBDR	RW	CPU	—	3
37	Cache Disable	CADR	RW	CPU	—	3
38	Machine Check Error Summary	MCESR	RW	CPU	—	3
39	Cache Error	CAER	RW	CPU	—	3
40	Accelerator Control/Status	ACCS	RW	CPU	—	4
41	Console Saved Interrupt Stack Pointer	SAVISP	R	CPU	—	2
42	Console Saved PC	SAVPC	R	CPU	—	2
43	Console Saved PSL	SAVPSL	R	CPU	—	2
44	WCS Address	WCSA	RW	CPU	—	4
45	WCS Data	WCSD	RW	CPU	—	4
46	reserved	—	—	—	—	4
47	reserved	—	—	—	—	4
48	SBI Fault/Status	SBIFS	RW	CPU	—	3

*Refer to Processor Register description.

Number	Register Name	Mnemonic	Type	Scope	Initialize	Category*
49	SBI Silo	SBIS	R	CPU	—	3
50	SBI Silo Comparator	SBISC	RW	CPU	—	3
51	SBI Maintenance	SBIMT	RW	CPU	—	3
52	SBI Error Register	SBIER	RW	CPU	—	3
53	SBI Timeout Address	SBITA	R	CPU	—	3
54	SBI Quadword Clear	SBIQC	W	CPU	—	3
55	IO Bus Reset	IORESET	W	CPU	—	3
56	Memory Management Enable	MAPEN	RW	CPU	yes	1
57	Trans. Buf. Invalidate All	TBIA	W	CPU	—	1
58	Trans. Buf. Invalidate Single	TBIS	W	CPU	—	1
59	Translation Buffer Data	TBDATA	RW	CPU	—	3
60	Microprogram Break	MBRK	RW	CPU	—	3
61	Performance Monitor Enable	PMR	RW	PROC	—	3
62	System Identification	SID	R	CPU	—	1
63	Translation Buffer Check	TBCHK	W	CPU	—	1
64:127	reserved	—	—	—	—	4

*Refer to Processor Register description.

Data and Bus Cycle Classification

Data cycles and read/write bus cycles, in the CVAX CPU, are grouped according to classes. The classes are determined by the type of data to be transferred and if the data is required immediately by the CPU. Status information, related to the type and class of bus cycle, is transferred onto the CSDP<2:0> lines during the address part of a bus cycle.

Data Class—The data class includes I-stream (instruction stream) and D-stream (data stream). I-stream references are generated by the CPU when prefetching instructions in the instruction stream. D-stream references are generated by the CPU when data is required by the executing instruction, when resolving a failed I-stream reference, or when filling a cache memory location.

Bus Cycle Class—There are three basic classes of read cycles and one write cycle. The cycles are request I-stream read, request D-stream read, and demand D-stream read and write. Each class of bus cycle is also grouped according to the type of bus or memory operation performed. These are read, read lock, read modify intent, read no lock or modify, write unlock, and write no unlock.

Request read cycles are generated when data is not immediately required by the CPU. For example, prefetching the I-stream (request I-stream read) and filling the second cache longword during a D-stream read (request D-stream read) generate request reads.

Demand read cycles are generated when data is immediately required by the CPU. For example, when an operand, PTE, SCB and PCB references all generate demand D-stream reads.

Write cycles are generated when data is to be written to cache and external memory.

Request and demand read cycles respond differently to errors reported during the reference. Request read errors usually do not affect program flow, and demand read errors cause a machine check abort. The effects of errors on the operation of the CPU during these cycles are described in the *Error Handling* section.

Cache Memory

To optimize the performance of the memory subsystem, the CVAX CPU contains a 1 KByte, two-way associative, 8-byte block cache memory. Cache memory can be configured to store I-stream only, I-stream and D-stream, or D-stream only (diagnostic use) references.

Organization—The CVAX CPU cache memory is organized into two sets of 64 rows as shown in Figure 34. Each row in a set is made up of a Valid (V) bit, a 20-bit tag with parity, and 8-byte data block with byte parity shown in Figure 35.

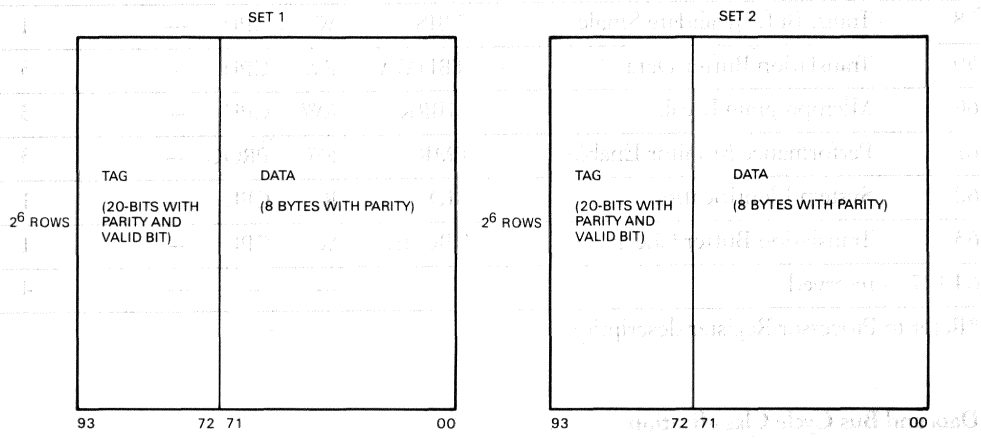


Figure 34 • CVAX 78034 Cache Memory Organization

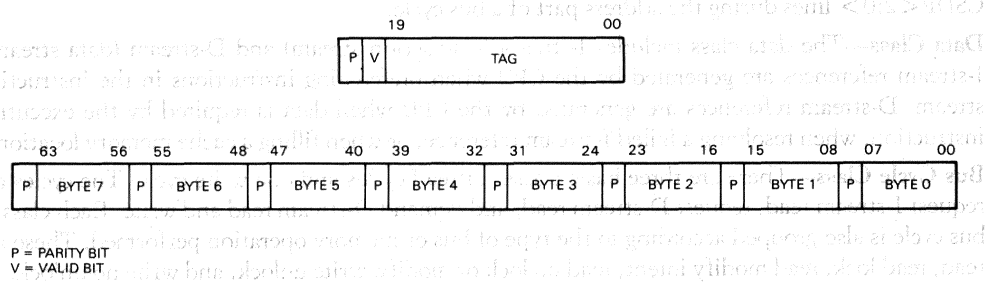


Figure 35 • CVAX 78034 Cache Tag and Data Format

Control

Operation of cache memory is controlled by the Cache Disable Register (CADR) and the $\overline{\text{CCTL}}$ signal. Status information is reported by the Memory Error Register (MEMER) and $\overline{\text{CSDP3}}$. The CADR register determines the operating mode of the cache and selects the set(s) to be enabled.

External logic can use the $\overline{\text{CCTL}}$ signal to prevent the storing of data in cache during CPU read cycles and to invalidate cache entries during DMA cycles that write to a memory location stored in cache. $\overline{\text{CSDP3}}$ allows external logic to track the set in the internal cache that has been allocated. This allows a coherent external cache memory system to be constructed.

Access—A cache memory location is accessed by a physical address generated by the CPU. The cache physical addresses are shown in Figure 36. The function of each field of the physical address is described in Table 20.

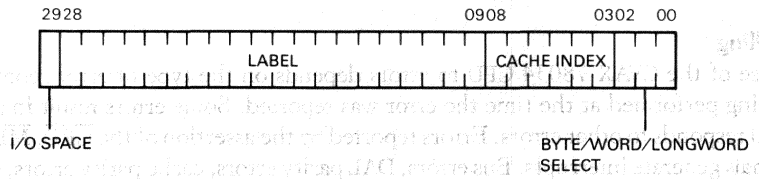


Figure 36 • CVAX 78034 Physical Address for Cache Access

Table 20 • CVAX 78034 Physical Address Description

Bit	Description
29	I/O (input/output)—This bit indicates whether the physical address is in I/O space. When set, the physical address is in I/O space. I/O space references are never stored in cache.
28:09	Label—These bits are compared to the TAG field(s) of the row selected by the Cache Index bits 08:03.
08:03	Cache Index—These bits select the row in cache memory to be accessed.
02:00	Byte/Word/Longword Select—These bits select the bytes to be accessed in the data block when there is a cache hit.

Cachable reference—A cachable reference has the following characteristics:

- The reference matches the type selected by bits 05:04 of the CADR. These are I-stream only, I-stream and D-stream, or D-stream only (diagnostic use).
- The reference is not a read lock reference.
- The reference is not in I/O space, bit 29 of the physical address is 0.

Cache hit—A cache hit occurs when the requested data is present and valid in cache memory. A hit is recognized when the label field of the physical address is the same as a tag in the selected set(s) and the entry is valid. During a CPU read operation, the data is from cache memory and no external bus cycle is performed. During a CPU write operation, cache memory and external memory are updated. This is defined as a write-through.

Cache miss—A cache miss occurs when the requested data is not in cache memory or is not valid. A cache miss during a CPU read operation results in a cache allocation if the reference is a cachable. A cache location cannot be allocated on a write-miss.

Cache allocation—The CVAX CPU allocates a cache memory location when a CPU read operation to a cacheable reference results in a cache miss. When the CVAX allocates a cache memory location,

it initiates a multiple transfer CPU read cycle. This bus cycle will read two longwords from memory to fill the allocated 8-byte row in cache. The first longword read is the one that contains the data requested by the CPU (preferred longword). The second longword read completes the quadword in the row.

Random set selection is used when both sets in cache memory are selected. The CPU does not differentiate between valid and invalid entries when selecting the set for a cache allocation. When the CPU allocates a row in cache, it clears the valid bit for the row in the selected set, fetches the preferred longword, fills the row with the second longword, and sets the valid bit if no errors occur. Refer to the *Multiple Transfer CPU Read Cycles section.*

Error Handling

The response of the CVAX 78034 CPU to errors depends on the type of error reported and the function being performed at the time the error was reported. Some errors result in an interrupt, and the CPU responds to other errors. Errors reported by the assertion of the $\overline{\text{CRD}}$, $\overline{\text{MEMERR}}$, and $\overline{\text{PWRFL}}$ signals generate interrupts. Bus errors, DAL parity errors, cache parity errors, and memory management errors have a defined response from the CPU.

Bus errors—External logic notifies the CPU of a bus error by asserting the $\overline{\text{ERR}}$ signal during a bus cycle. The response of the CPU to a bus errors is summarized in Table 21. External logic can also request a retry of some bus cycles by asserting the $\overline{\text{ERR}}$ and $\overline{\text{RDY}}$ signals.

Table 21 • CVAX 78034 Response to Bus Errors and DAL Parity Errors

Cycle type	Prefetch	Cache[1]	Error status[2]	Results
demand D-stream (read)	—	entry is invalidated	logged in MESRbits 06:05	machine check abort
write	—	—	—	machine check abort
request D-stream (read)	—	entry is invalidated	logged in MESR bit 06	—
request I-stream (read)	prefetch halted	entry is invalidated	logged in MSER bit 06	—

[1]The entire row in cache memory selected by the faulting address is invalidated whether the reference is cachable or not cachable. The entries from both sets are invalidated.
[2]Only DAL parity errors will log the status.

DAL parity errors—External logic enables DAL parity checking by asserting the $\overline{\text{DPE}}$ signal. Each 8-bit byte of DAL data is conditionally checked by a parity bit. Odd data bytes have odd parity and even data bytes have even parity. The parity sense is alternated in order to detect stuck-at-one faults and stuck-at-zero faults. DAL parity checking can be disabled, reference by reference, by deasserting the $\overline{\text{DPE}}$ signal.

The action following the detection of a DAL parity error depends on the type of reference. During a demand D-stream reference, the cache entry is invalidated, the cause of the error is logged in the MSER bits 06:05, and a machine check abort is initiated. During request D-stream and I-stream references, the cache entry is invalidated, the cause of the error is logged in MSER bit 06, and no abort occurs. Table 21 lists responses of the CPU to DAL parity errors.

Cache parity—The CVAX CPU protects the internal cache with parity. Each 8-bit byte of cache data and the 20-bit tag field is checked by a parity bit. Odd data bytes record odd parity and even data bytes record even parity. The tag field records odd parity. The stored parity is valid only when the valid bit associated with the cache entry is set. Cache parity is checked on all cachable read and write references that can be stored in cache and on DMA invalidate cycles. Read cycles report cache parity errors when a valid tag matches bits 28:09 of the physical address and either the stored tag or the longword selected by address bit 02 generate a parity error. Write and DMA invalidate cycles report cache parity errors when a valid tag matches bits 28:09 of the physical address and the stored tag generates a parity error.

The results of detecting a cache parity error depend on the reference type. During a demand D-stream reference, the entire cache is cleared and disabled (CADR is cleared), the cause of the error is logged in MSER bits 04:00, and a machine check abort is initiated. During a DMA invalidate cycle, the cache remains unchanged, the cause of the error is logged in MSER bits 3:0, and an abort does not occur. During a request I-stream reference, the entire cache is cleared but it remains enabled, the cause of the error is logged in MSER bits 3:0, prefetching is halted, and an abort does not occurs.

The responses of the CPU to cache parity errors is listed in Table 22.

Table 22 • CVAX 78034 Response to Cache Parity Error

Cycle type	Prefetch	Cache	Error status	Results
demand D-stream (read)	—	clear cache and disabled[1]	logged in MSER bits 04:00	machine check abort
write cache hit	—	clear cache[1]	logged in MSER bits 03:00[2]	—
DMA invalidate cache hit	—	no cache change	logged in MSER bits 03:00[2]	—
write cache miss		(not possible)		
request D-stream (read)		(not possible)		
request I-stream (read)	prefetch halted	clear cache[1]	logged in MSER bits 03:00	—

[1]The cache is cleared only if CADR bit 00 is cleared.
[2]A parity error is detected only in the tags.

Memory Management Error—The CPU response to memory management faults is listed in Table 23. Refer to *Memory-management Faults* for a description of memory management faults.

Table 23 • CVAX 78034 Response to Memory-management Faults

Cycle type	Prefetch	Results
demand D-stream (read)	—	memory-management fault (ACV, TNV, etc.)

Cycle type	Prefetch	Results
write	—	memory-management fault (ACV, TNV, etc.)
request D-stream (read)	(not possible)	
request I-stream (read)	prefetch halted	

Interfacing Requirements

The power supply, clock timing, and bus connections to the CVAX CPU chip are described in the following paragraphs.

Power and Ground Connections

The CVAX 78034 requires a single 5-volt power supply. Six V_{DD} pins and six V_{SS} pins connect to the power supply and ground. The TEST/V_{SS} pin connects to the supply ground or can be used for test purposes. Figure 37 shows the power and ground connection and decoupling. Table 24 lists the CVAX CPU pin and associated power and ground requirements.

Note

Care must be taken when connecting the V_{DD} and V_{SS} pins. The V_{DD} pins should be connected together and to the 5-volt power plane using short wires. The V_{SS} pins should also connect together and to the ground plane using short leads. The power supply should be decoupled by connecting a 0.33 f and a 0.047 f ceramic or equivalent capacitor between each V_{DD} pin and its associated V_{SS} pin.

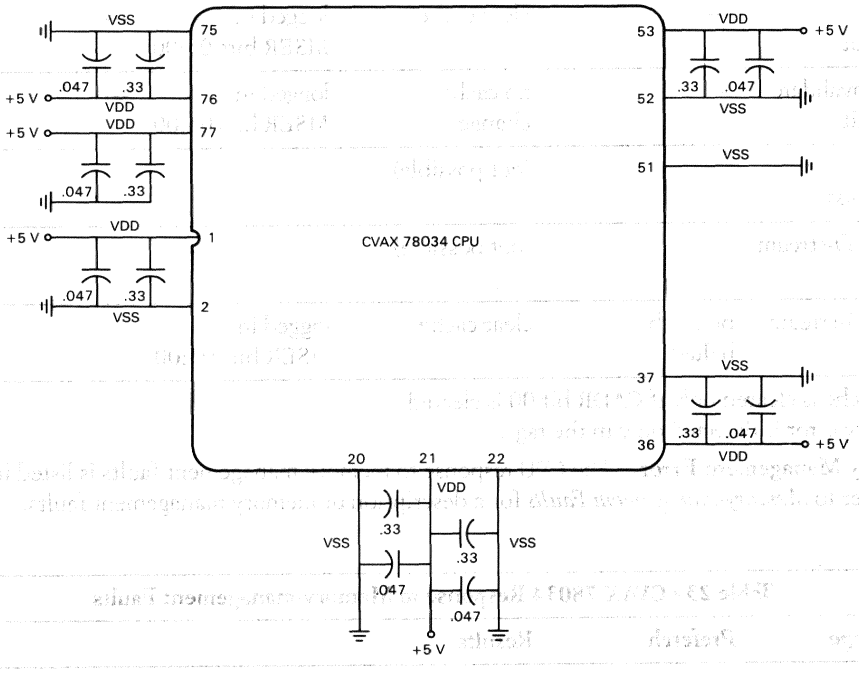


Figure 37 • CVAX 78034 CPU Power and Ground Connections

Table 24 • CVAX 78034 CPU Power Distribution

Pin	Type	Output signals powered
76,53	V_{DD}	DAL<31:00>, $\overline{BM}$<3:0>
75,52	V_{SS}	DAL<31:00>, $\overline{BM}$<3:0>
77	V_{DD}	cache and internal IDAL drivers
51	V_{SS}	cache and internal IDAL drivers
1,36	V_{DD}	internal logic
2,37	V_{SS}	internal logic
21	V_{DD}	CPDAT<5:0>, CPSAT<1:0>, TEST, $\overline{CWB}$, $\overline{CCTL}$, $\overline{DMG}$, $\overline{DS}$, $\overline{AS}$, $\overline{DBE}$, $\overline{WR}$, $\overline{CSDP}$<3:0>
22	V_{SS}	CPDAT<5:0>, CPSAT<1:0>, TEST, $\overline{CWB}$, $\overline{CCTL}$, $\overline{DMG}$, $\overline{DS}$, $\overline{WR}$, $\overline{CSDP}$<3:0>
20	V_{SS}	$\overline{AS}$

Clocks and Synchronization

The CVAX CPU uses two precision MOS clock inputs to generate its internal timing and control signals. These clocks are provided by the CVAX 78135 clock generator. The TTL level oscillator input provides the two 180-degree, phase shifted, precision MOS clock signals required by the CPU.

The $\overline{RESET}$, $\overline{RDY}$, and $\overline{ERR}$ signals to the CPU must be asserted synchronously with respect to the CLKA and CLKB inputs. To aid the system designer, the CVAX 78135 clock (CCLOCK) generator provides a common synchronization point for these signals. This allows peripheral support chips and other devices to operate asynchronously with the CCLOCK and to synchronize these inputs to meet the timing requirements of the CPU. Figure 38 shows the CVAX 78135 CCLOCK in a CVAX 78034 CPU system. Care must be taken during board layout to limit the amount of skew between the CLKA and CLKB inputs of the CVAX CPU so that the timing parameters are met.

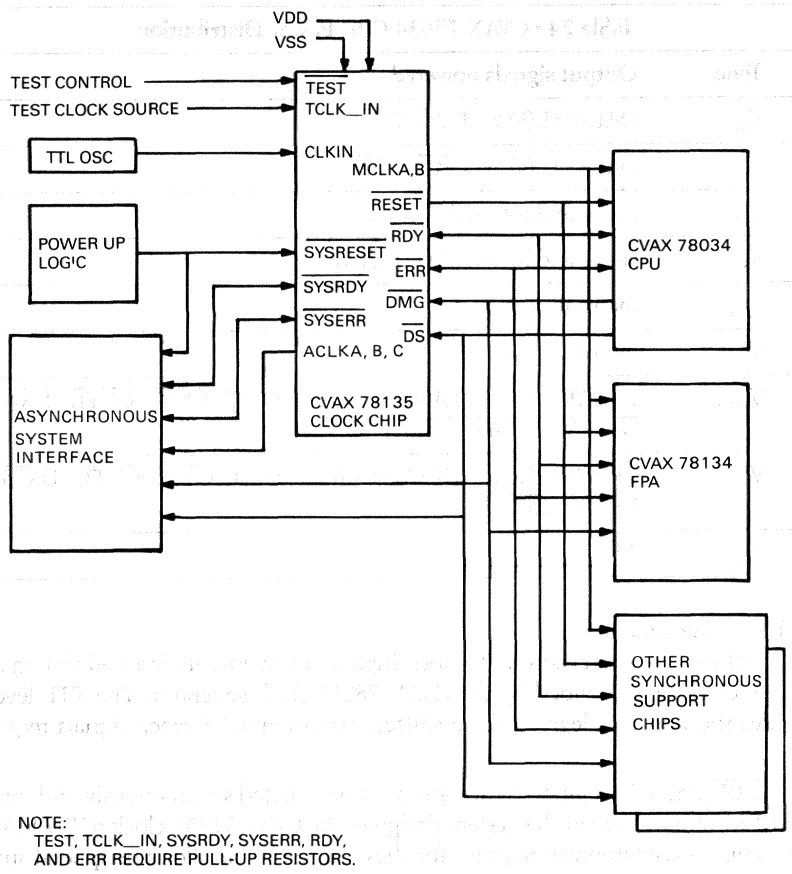


Figure 38 • CVAX 78034 CPU System with CVAX 78135 Clock Generator

Strobe Termination

To eliminate interactions between the output strobes of the CVAX 78034 CPU, each strobe output must be terminated with a series resistor. The strobe ouputs that requiring resistors are $\overline{AS}$, $\overline{DS}$, $\overline{DBE}$, $\overline{WR}$, $\overline{DPE}$, $\overline{CSDP<3:0>}$, and $\overline{CWB}$. The resistor value should be from 20Ω to 47Ω, however, the value depends on the layout and loading of each strobe. The resistor value selected should dampen the transmission line reflections. A 10Ω series resistor reduces a glitch by approximately 1.0 volt. The terminating resistors should be connected as close to the signal pin as possible.

Bus Cycles

The CVAX CPU performs a bus cycle when

- Reading or writing information to or from memory, a peripheral device, or an externally implemented processor register.
- Acknowledging an interrupt and reading a device interrupt vector.
- Transferring information from or to the CVAX 78134 FPA.

Figure 39 shows the bus connections used by the CVAX CPU.

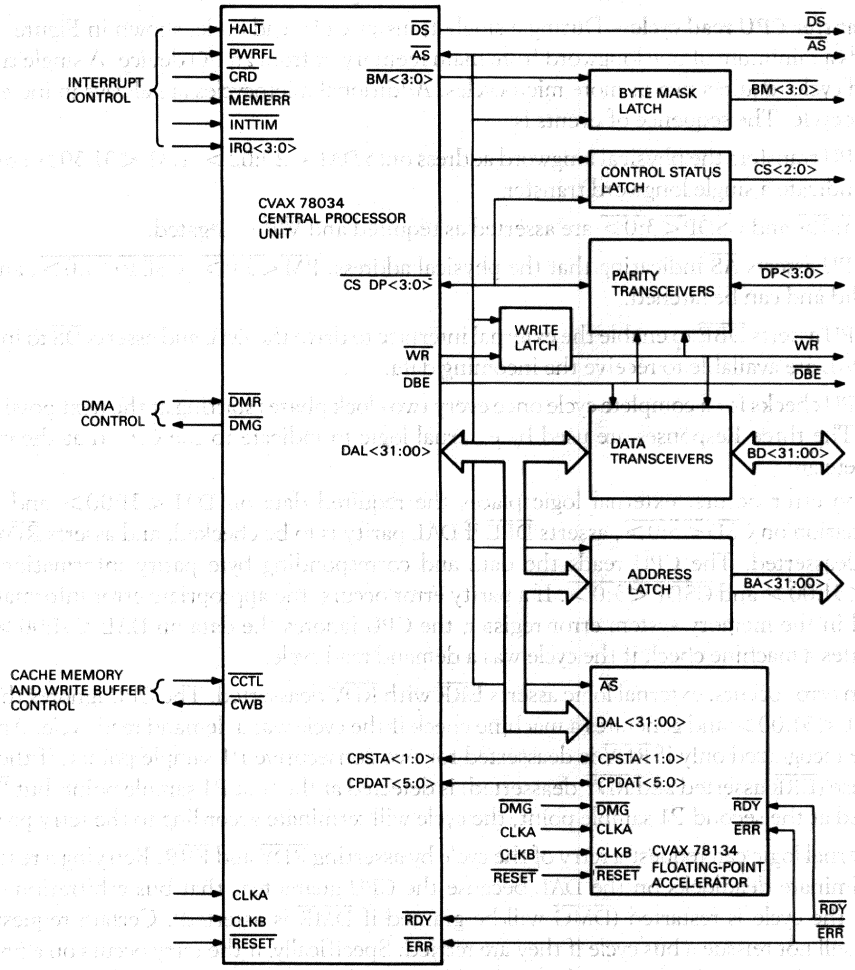


Figure 39 • CVAX 78034 CPU Bus Connections

A microcycle is the basic timing unit for a bus cycle. A microcycle is defined as four clock phases (P1 through P4) as shown in Figure 40. Detailed timing information for the following bus cycles is contained in the *ac Electrical Characteristics*.

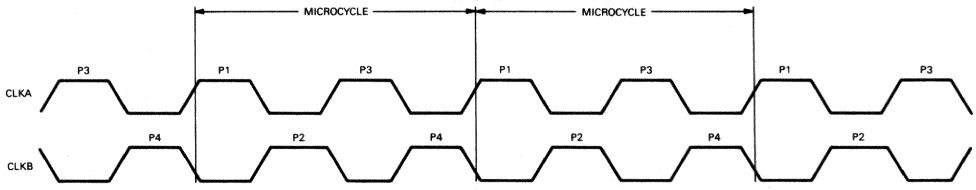


Figure 40 • CVAX 78034 Microcycle

Idle cycle—An idle cycle requires one microcycle. During an idle cycle, DAL<31:00> are undefined and the bus control signals are not asserted.

Single transfer CPU read cycle—During a single transfer CPU read cycle, shown in Figure 41, the CPU reads a minimum of one longword from main memory or from an I/O device. A single transfer CPU read cycle requires two or more microcycles. Additional microcycles are always in increments of a microcycle. The sequence of events is

1. The CPU transfers the physical longword address onto DAL<29:02>. DAL<31:30> are set to 01 to indicate a single longword transfer.
2. $\overline{\text{BM}}$<3:0> and $\overline{\text{CSDP}}$<3:0> are asserted as required and $\overline{\text{WR}}$ is negated.
3. The CPU asserts $\overline{\text{AS}}$ indicating that the physical address, $\overline{\text{BM}}$<3:0>, $\overline{\text{CSDP}}$<3:0>, and $\overline{\text{WR}}$ are valid and can be latched.
4. The CPU asserts $\overline{\text{DBE}}$ to enable the external interface to drive the DAL and asserts $\overline{\text{DS}}$ to indicate that DAL are available to receive the incoming data.
5. The CPU checks for a complete cycle once every two clock phases starting at the next possible P1 edge. The three Responses are used by external logic to indicate to the CPU that the cycle is complete are

   a. If no error occurs, external logic places the required data on DAL<31:00> and parity information on $\overline{\text{CSD}}$<3:0>, asserts $\overline{\text{DPE}}$ if DAL parity is to be checked, and asserts $\overline{\text{RDY}}$ with $\overline{\text{ERR}}$ deasserted. The CPU reads the data and corresponding byte parity information from DAL<31:00> and $\overline{\text{CSDP}}$<3:0>. If a parity error occurs, the appropriate error information is logged in the memory system error register, the CPU ignores the data on DAL<31:00>, and generates a machine check if the cycle was a demand read cycle.

   b. If an error occurs, external logic asserts $\overline{\text{ERR}}$ with $\overline{\text{RDY}}$ deasserted. The CPU ignores the data on DAL<31:00> and generates a machine check if the cycle was a demand read cycle. An error will be recognized only if $\overline{\text{RDY}}$ is deasserted for two consecutive P1 sample points. If the error response ($\overline{\text{ERR}}$ asserted and $\overline{\text{RDY}}$ deasserted) is detected at the first P1 sample point, but $\overline{\text{RDY}}$ is asserted at the second P1 sample point, the cycle will terminate according to the retry protocol.

   c. External logic can request a retry of the cycle by asserting $\overline{\text{RDY}}$ and $\overline{\text{ERR}}$. Retrying a read cycle can eliminate deadlocks on the DAL because the CPU guarantees that bus arbitration occurs before the cycle is restarted ($\overline{\text{DMG}}$ will be granted if $\overline{\text{DMR}}$ is asserted). Certain request read cycles will not reissue a bus cycle if they are retried. Specifically, if the retry occurs on a prefetch reference, the operation may not be reissued because the CPU may execute a branch operation before the prefetch can be retried.

6 The CPU completes the cycle by deasserting $\overline{\text{DS}}$, $\overline{\text{DBE}}$ and $\overline{\text{AS}}$.

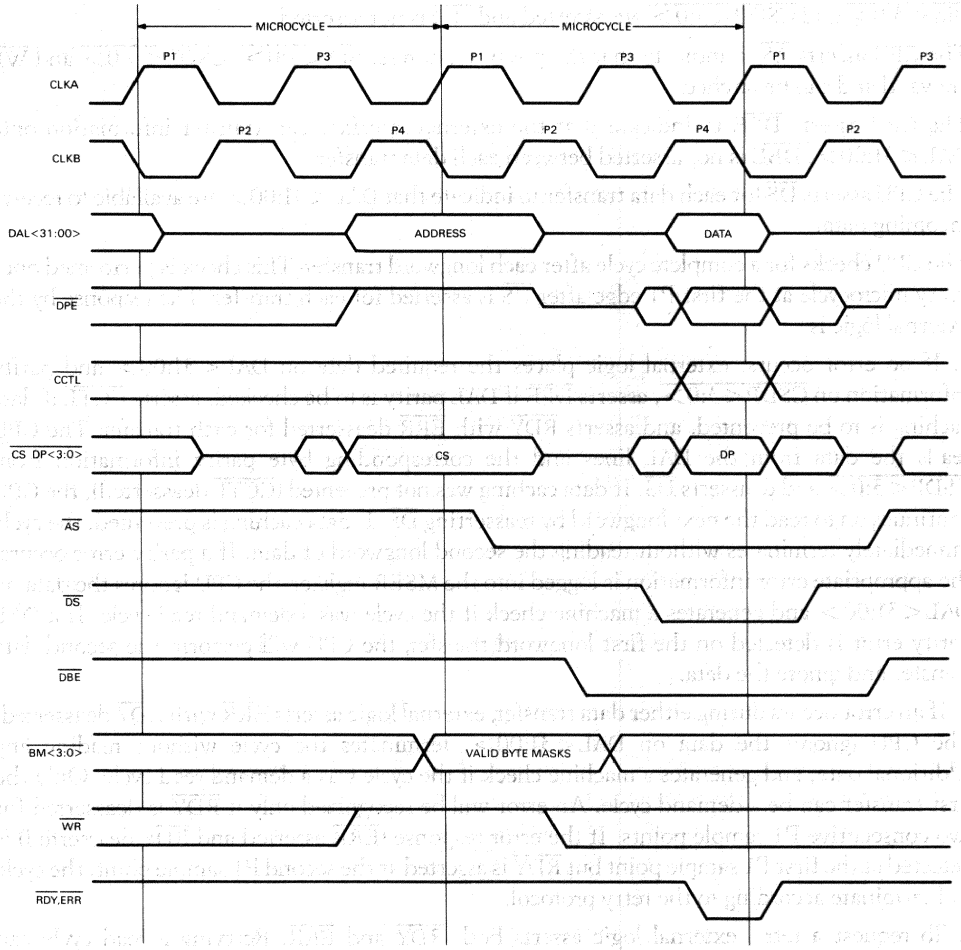


Figure 41 • CVAX 78034 Single Transfer CPU Read Cycle

Multiple transfer CPU read cycle—During multiple transfer CPU read cycles shown in Figure 42, the CPU reads two longwords (one quadword) from main memory. A multiple transfer CPU read cycle requires a minimum of three microcycles. Each longword transfer may be increased in increments of one microcycle. I/O space read references always occur as single transfer read cycles. The sequence of events for a multiple transfer CPU read cycle is

1. The CPU transfers the physical address of the preferred longword that is to be accessed onto DAL<29:02>. This address can be aligned with either of the longword addresses within the quadword block. DAL<31:30> are set to 10 to indicate a quadword transfer. The CPU sends an address only on the initial longword (preferred) transfer of a multiple transfer read cycle. The address associated with the second (cache fill) transfer is implied and therefore is not transferred by the CPU. External logic can generate the implied address by inverting address bit 02 of the preferred address. All references, therefore, remain within a quadword block. For example, if the initial longword address in a quadword transfer is 0007FB36 (hexadecimal), the implied address is 0007FB32.

2. $\overline{BM<3:0>}$ and $\overline{CSDP<3:0>}$ are asserted and $\overline{WR}$ is not asserted.
3. The CPU asserts $\overline{AS}$ to indicate that the physical address, $\overline{BM<3:0>}$, $\overline{CSDP<3:0>}$ and $\overline{WR}$ are valid and can be latched.
4. The CPU asserts $\overline{DBE}$ to indicate that the external interface can transfer information onto DAL<31:00>. $\overline{DBE}$ is not asserted between each data transfer.
5. The CPU asserts $\overline{DS}$ for each data transfer to indicate that DAL<31:00> are available to receive incoming data.
6. The CPU checks for a complete cycle after each longword transfer. This check is performed once every microcycle at the first P1 edge after $\overline{DS}$ is asserted for each transfer. The response by the external logic is

   a. If no error occurs, external logic places the required data on DAL<31:00> and parity information on $\overline{CSDP<3:0>}$, asserts $\overline{DPE}$ if DAL parity is to be checked, asserts $\overline{CCTL}$ if data caching is to be prevented, and asserts $\overline{RDY}$ with $\overline{ERR}$ deasserted for each transfer. The CPU reads the data from the DAL lines and the corresponding byte parity information from $\overline{CSDP<3:0>}$ and deasserts $\overline{DS}$. If data caching was not prevented ($\overline{CCTL}$ deasserted), the CPU continues on to read the next longword by reasserting $\overline{DS}$. If data caching is prevented, the cycle immediately terminates without reading the second longword of data. If a parity error occurs, the appropriate error information is logged into the MSER register, the CPU ignores the data on DAL<31:00> and generates a machine check if the cycle was a demand read cycle. If a DAL parity error is detected on the first longword transfer, the CPU will perform the second data transfer and ignore the data.

   b. If an error occurs during either data transfer, external logic asserts $\overline{ERR}$ with $\overline{RDY}$ deasserted. The CPU ignores the data on DAL<31:00>, terminates the cycle without reading any additional data, and generates a machine check if the cycle was a demand read cycle. Only the first transfer can be a demand cycle. An error will be recognized only if $\overline{RDY}$ is deasserted for two consecutive P1 sample points. If the error response ($\overline{ERR}$ asserted and $\overline{RDY}$ deasserted) is detected at the first P1 sample point but $\overline{RDY}$ is asserted at the second P1 sample point, the cycle will terminate according to the retry protocol.

   c. To request a retry, external logic asserts both $\overline{RDY}$ and $\overline{ERR}$. Retrying a read cycle can eliminate DAL deadlocks because the CPU guarantees that bus arbitration occurs before the cycle is restarted ($\overline{DMG}$ will be granted if $\overline{DMR}$ is asserted). If the retry occurs during the second longword transfer, the read cycle will not be reissued.
7. The CPU completes the cycle by deasserting $\overline{AS}$, $\overline{DBE}$, and $\overline{DS}$.

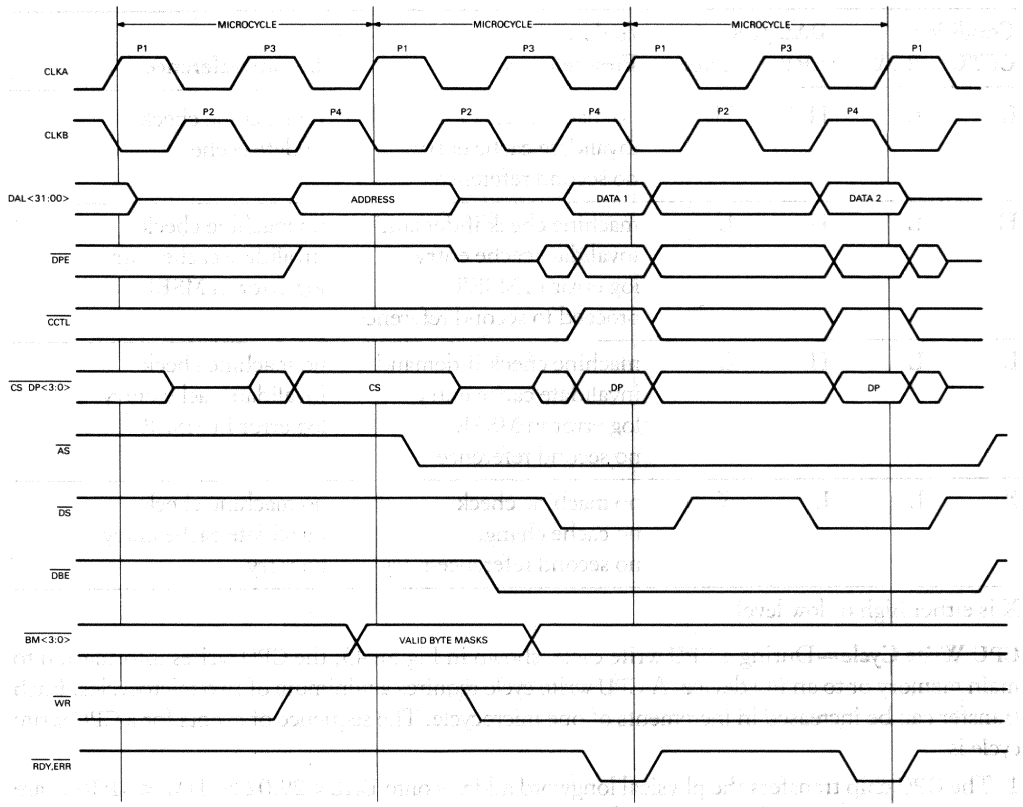


Figure 42 • CVAX 78034 Multiple Transfer CPU Read Cycle

Normally, a multiple transfer CPU read cycle reads two longwords of data. However, the cycle terminates after the first data transfer if $\overline{ERR}$ is asserted and $\overline{RDY}$ is deasserted (memory error), or if $\overline{CCTL}$ is asserted to prevent data caching. The cycle does not terminate early if a DAL parity error is detected on the first transfer. Table 25 lists the possible multiple transfer cycle responses.

Table 25 • CVAX 78034 Responses to a Multiple Transfer CPU Read Cycle

Condition		DAL parity		Action	
CCTL	RDY	ERR	error	First reference	Second reference
X	H	H	X	wait for data	wait for data
X	H	L	X	machine check if demand invalidate cache entry no second reference	no machine check invalidate cache entry
H	L	H	H	no machine check update cache proceed to second reference	no machine check update cache

Condition CCTL	RDY	DAL parity ERR	error	Action First reference	Second reference
L	L	H	H	no machine check invalidate cache entry no second reference	no machine check update cache
H	L	H	L	machine check if demand invalidate cache entry log error in MSER proceed to second reference	no machine check invalidate cache entry log error in MSER
L	L	H	L	machine check if demand invalidate cache entry log error in MSER no second reference	no machine check invalidate cache entry log error in MSER
X	L	L	X	no machine check no cache change no second reference-retry	no machine check invalidate cache entry no retry

X is either high or low level

CPU Write Cycle—During a CPU write cycle, shown in Figure 43, the CPU writes information to main memory or to an I/O device. A CPU write cycle requires a minimum of two microcycles. Each transfer can be increased in increments of one microcycle. The sequence of events for a CPU write cycle is

1. The CPU chip transfers the physical longword address onto DAL<29:02>. DAL<31:30> are set to 01 to indicate a longword transfer.
2. $\overline{\text{BM}}$<3:0> and $\overline{\text{CSDP}}$<3:0> are asserted as required and $\overline{\text{WR}}$ is asserted.
3. The CPU asserts $\overline{\text{AS}}$ to indicate that the physical address, $\overline{\text{BM}}$<3:0>, $\overline{\text{CSDP}}$<3:0>, and $\overline{\text{WR}}$ are valid and can be latched.
4. The CPU asserts $\overline{\text{DBE}}$ to indicate the write data can be transferred onto an external bus.
5. The CPU transfers the output data onto DAL<31:00> and byte parity information onto $\overline{\text{CSDP}}$<3:0>, asserts $\overline{\text{DPE}}$ to indicate that valid parity information is available, and asserts $\overline{\text{DS}}$ to indicate that the DAL contains valid data.
6. The CPU checks for a complete cycle once every two clock phases starting at the next possible P1. The response of the external logic is

   a. If no error occurs, external logic reads the data from the DAL<31:00> and asserts $\overline{\text{RDY}}$ with $\overline{\text{ERR}}$ deasserted.

   b. If an error occurs, external logic asserts $\overline{\text{ERR}}$ with $\overline{\text{RDY}}$ deasserted. Aborting a write cycle generates a machine check. External logic can report a DAL parity error by asserting $\overline{\text{ERR}}$ and deasserting $\overline{\text{RDY}}$. An error will be recognized only if $\overline{\text{RDY}}$ is deasserted for two consecutive P1 sample points. If the error response $\overline{\text{ERR}}$ asserted and $\overline{\text{RDY}}$ deasserted) is detected at the first P1 sample point but $\overline{\text{RDY}}$ is asserted at the second P1 sample point, the cycle will terminate according to the retry protocol.

   c. To request a retry, external logic asserts both $\overline{\text{RDY}}$ and $\overline{\text{ERR}}$. DAL arbitration occurs after the write operation is terminated.
7. The CPU completes the cycle by deasserting $\overline{\text{AS}}$, $\overline{\text{DBE}}$, and $\overline{\text{DS}}$.

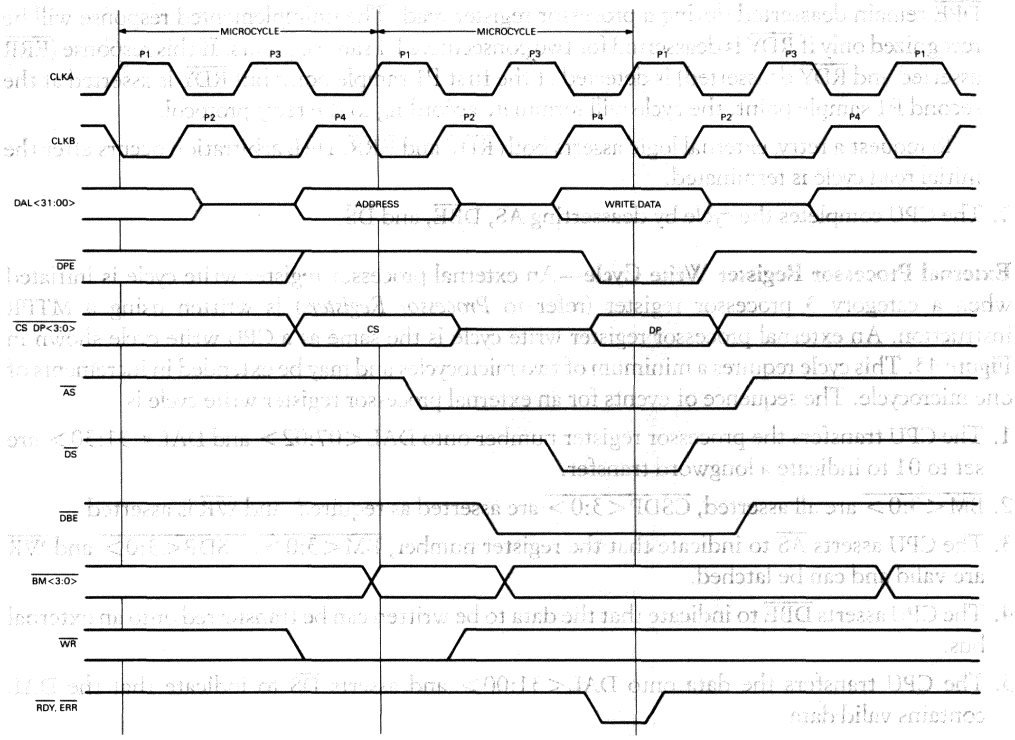


Figure 43 • CVAX 78034 CPU Write Cycle

External Processor Register Read Cycle—An external processor register read cycle is initiated when a category 3 processor register (refer to *Processor Registers*) is read using a MFPR instruction. The external processor register read cycle is the same as a single transfer CPU read cycle shown in Figure 41. This cycle requires a minimum of two microcycles and can be extended in increments of one microcycle. The sequence of events for an external processor register read cycle is

1. The CPU transfers the processor register number onto DAL<07:02>, and DAL<31:30> are set to 01 to indicate longword transfer.
2. $\overline{BM<3:0>}$ are all asserted, $\overline{CSDP<3:0>}$ are asserted as required and $\overline{WR}$ is unasserted.
3. The CPU asserts $\overline{AS}$ indicating that the register number, $\overline{BM<3:0>}$, $\overline{CSDP<3:0>}$, and $\overline{WR}$ are valid and can be latched.
4. The CPU asserts $\overline{DBE}$ to indicate that read data can be transferred onto the DAL.
5. The CPU asserts $\overline{DS}$ to indicate that DAL are available to receive incoming data.
6. The CPU checks for a complete cycle once every two clock phases at the next possible P1. The response of external logic is

   a. If the processor register is implemented, external logic transfers the required data on DAL<31:00>, deasserts $\overline{DPE}$, and asserts $\overline{RDY}$ with $\overline{ERR}$ deasserted. The CPU reads the data from DAL<31:00>.

   b. If the processor register is not implemented, external logic asserts $\overline{ERR}$ with $\overline{RDY}$ deasserted. The CPU ignores the data on DAL<31:00> and internally forces the result to zero. A detected parity error will force the result to zero and is not reported. Therefore, it is recommended that

$\overline{DPE}$ remain deasserted during a processor register read. The unimplemented response will be recognized only if $\overline{RDY}$ is deasserted for two consecutive P1 sample points. If this response ($\overline{ERR}$ asserted and $\overline{RDY}$ deasserted) is detected at the first P1 sample point but $\overline{RDY}$ is asserted at the second P1 sample point, the cycle will terminate according to the retry protocol.

c. To request a retry, external logic asserts both $\overline{RDY}$ and $\overline{ERR}$. DAL arbitration occurs after the initial read cycle is terminated.

7. The CPU completes the cycle by deasserting $\overline{AS}$, $\overline{DBE}$, and $\overline{DS}$.

External Processor Register Write Cycle—An external processor register write cycle is initiated when a category 3 processor register (refer to *Processor Registers*) is written using a MTPR instruction. An external processor register write cycle is the same as a CPU write cycle shown in Figure 43. This cycle requires a minimum of two microcycles and may be extended in increments of one microcycle. The sequence of events for an external processor register write cycle is

1. The CPU transfers the processor register number onto DAL<07:02> and DAL<31:30> are set to 01 to indicate a longword transfer.
2. $\overline{BM<3:0>}$ are all asserted, $\overline{CSDP<3:0>}$ are asserted as required, and $\overline{WR}$ is asserted.
3. The CPU asserts $\overline{AS}$ to indicate that the register number, $\overline{BM<3:0>}$, $\overline{CSDP<3:0>}$ and $\overline{WR}$ are valid and can be latched.
4. The CPU asserts $\overline{DBE}$ to indicate that the data to be written can be transferred onto an external bus.
5. The CPU transfers the data onto DAL<31:00> and asserts $\overline{DS}$ to indicate that the DAL contains valid data.
6. The CPU checks for a complete cycle once every two clock phases, starting at the next possible P1. The response of the external logic is

   a. If the processor register is implemented, external logic reads the data from DAL and asserts $\overline{RDY}$ while $\overline{ERR}$ is deasserted.

   b. If the processor register is not implemented, external logic either responds as if the register is implemented by asserting $\overline{ERR}$ when $\overline{RDY}$ is deasserted. Both responses have the same effect and no special action is taken. The unimplemented response initiates no special action only if $\overline{RDY}$ is deasserted for two consecutive P1 sample points. If this response is detected at the first P1 sample point, but $\overline{RDY}$ is Asserted at the second P1 sample point, the cycle will terminate according to the retry protocol.

   c. To request a retry, external logic asserts both $\overline{RDY}$ and $\overline{ERR}$. DAL arbitration occurs after the initial write cycle is terminated.
7. The CPU completes the cycle by deasserting $\overline{AS}$, $\overline{DBE}$, and $\overline{DS}$.

Interrupt Acknowledge Cycle—An interrupt acknowledge cycle sequence is similar to a single transfer CPU read cycle shown in Figure 41. The sequence of events is

1. DAL<06:02> transfers the IPL of the interrupt being acknowledged with IPL 17, IPL 16, IPL 15 and IPL 14 as $\overline{IRQ3}$, $\overline{IRQ2}$, $\overline{IRQ1}$, and $\overline{IRQ0}$, respectively. DAL<31:30> are set to 01, and DAL<29:07> and DAL<01:00> are set to zeros.
2. The data read is used to generate the vector and new IPL for the interrupt sequence. Bits 15:02 of the incoming data are used to create the vector offset within the system control block. The new processor status longword priority level is determined either by the external interrupt request level that caused the interrupt or by bit 00 of the value supplied by external hardware. If bit 00 is 0, the new IPL is determined by the interrupt request level being serviced. $\overline{IRQ3}$ sets the

IPL to 17 (hexadecimal) and IRQ0 to IPL 14 (hexadecimal). If bit 00 of the value supplied by external hardware is 1, the new IPL is forced to 17 (hexadecimal). Bits <31:16> and bit 01 of the incoming data are ignored.

3. Assertion of $\overline{ERR}$ in the proper order with $\overline{RDY}$ causes the bus cycle to be reissued or aborted. An abort causes the DAL data to be ignored and the CPU continues as if the interrupt request never occurred (passive release of the interrupt request). A detected DAL parity error also causes a passive release and is not reported. Therefore, it is recommended that $\overline{DPE}$ remain deasserted during an interrupt acknowledge cycle.

DMA Grant Cycle—The CPU can relinquish its control of the DAL bus and related control signals upon request from a DMA device or another CPU. Figure 44 shows the sequence of the DMA grant cycle. The sequence is

1. The external device requests control of the bus by asserting $\overline{DMR}$.
2. At the conclusion of the current bus cycle, the CPU responds by causing DAL<31:00>, $\overline{AS}$, $\overline{DS}$, $\overline{WR}$, $\overline{DBE}$, $\overline{BM}$<3:0>, and $\overline{CSDP}$<3:0> to become a high impedance and asserts $\overline{DMG}$.
3. The external device may now use the DAL to transfer data.
4. To return control of DAL to the CPU, the external device deasserts $\overline{DMR}$. The CPU responds by deasserting $\overline{DMG}$ and starting the next bus cycle.

The CPU ensures that successive DMA requests ($\overline{DMR}$ asserted) cannot prevent all CPU activity. As an example, one CVAX cycle can occur between two successive assertions of $\overline{DMR}$.

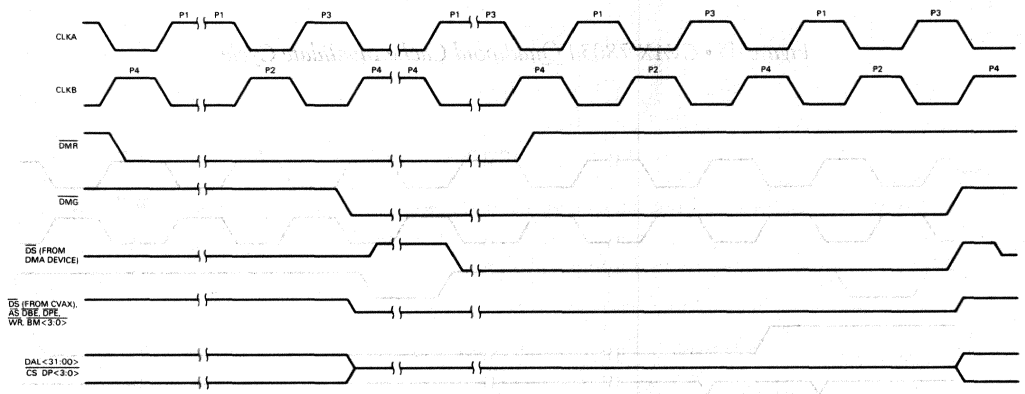


Figure 44 • CVAX 78034 DMA Grant Cycle

Cache Invalidate Cycles—External logic initiates a conditional cache invalidate cycle, shown in Figures 45 and 46, to allow the CPU to detect and invalidate stale data that is stored in the cache. A conditional invalidate cycle uses a minimum of three microcycles. The sequence of events for a cache invalidate cycle is

1. After $\overline{DMG}$ is asserted by the CPU, external logic asynchronously transfers the physical address onto DAL<31:00>, asynchronously asserts $\overline{AS}$ to latch the address into the CPU, and asynchronously asserts $\overline{CCTL}$ to start a conditional invalidate cycle.
2. The CPU invalidates the quadword cache entry selected by the DMA address if the location is stored in the cache.

3. External logic deasserts $\overline{CCTL}$ and optionally reasserts $\overline{CCTL}$ to conditionally invalidate the alternate quadword formed by inverting address bit 03 of the physical address. This allows external logic to detect and invalidate stale data stored in any naturally aligned octaword.
4. The cycle ends when external logic deasserts both $\overline{AS}$ and $\overline{CCTL}$.

If a cache parity error is detected during the conditional invalidate operation, no machine check is generated, no invalidate occurs, and the error is logged in the MSER.

The CPU detects and invalidates quadword stale data in three microcycles. Therefore, the maximum cache invalidate rate cannot exceed 8-byte or three microcycles (nominally 26.6 Mbytes per second).

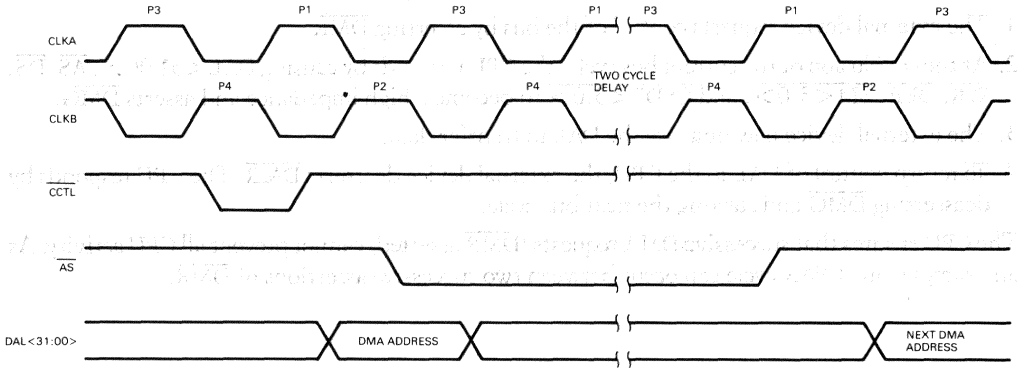


Figure 45 • CVAX 78034 Quadword Cache Invalidate Cycle

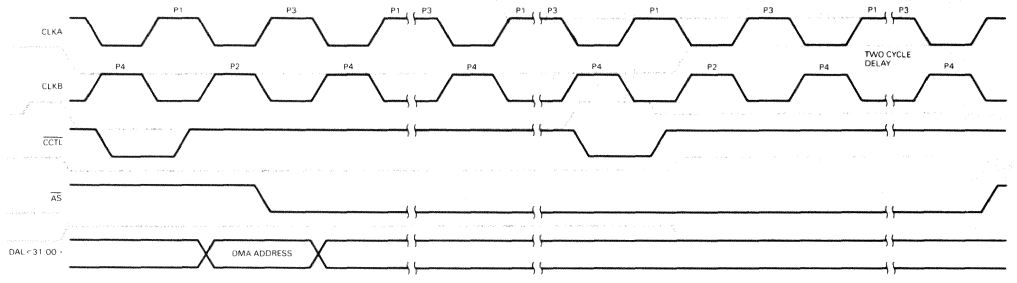


Figure 46 • CVAX 78034 Octaword Cache Invalidate Cycle

Coprocessor Protocols

Coprocessor protocols are used by the CVAX CPU when communicating with the optional CVAX 78134 FPA (CFPA). These devices communicate with each other through the CPSTA<1:0> and CPDAT<5:0> lines and DAL<31:00>. CPSTA<1:0> inform the CPU or CFPA on the method of interpretation of the CPDAT<5:0> information. The CPDAT<5:0> lines transfer opcode and control information to the CFPA and return condition code and exception status to the CPU. DAL<31:00> are used to transfer operands and results.

The protocol for the transfer of information between the two devices is

1. The CPU sends the opcode for the instruction to be executed and the operand(s) to the CFPA.
2. The CPU waits for the CFPA to complete the instruction. DMA devices may be granted use of DAL<31:00> and its associated control signals while the CPU waits.
3. The CFPA notifies the CPU that the result is ready. Condition codes and error information are transferred on CPDAT<5:0> lines by the FPA.
4 The CFPA transfers any results of the computation through DAL<31:00> during consecutive microcycles.
5. When the operation is complete, the CPU can send another opcode to the CFPA.

Opcode Transfer—The CPU transfers opcode information to the CFPA when the CFPA is ready to execute an instruction. The transfer cycle is shown in Figure 47. The CPU transfers six low-order bits of the opcode onto the CPDAT<5:0> lines and the opcode type (F, D, G floating or integer), onto the CPSTA<1:0> lines.

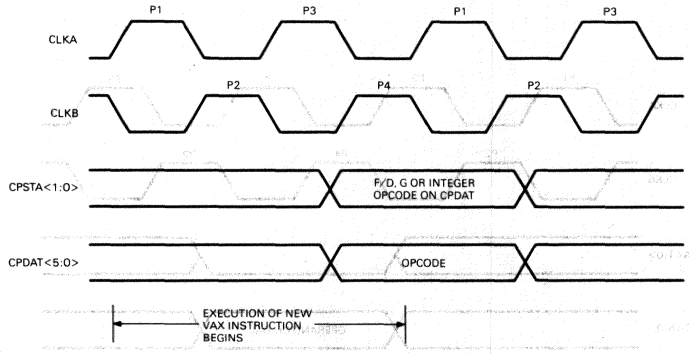


Figure 47 • CVAX 78034 Opcode Transfer Cycle

Operand Transfer—After sending the opcode to the CFPA, the CPU transfers the necessary operands to the CFPA as shown in Figures 48 and 49. The operand(s) can originate from the general registers or the internal cache memory of the CPU or from external memory. The CFPA monitors the $\overline{AS}$ signal to determine if the source of the operand. When $\overline{AS}$ is deasserted, the operand is from the CPU. When $\overline{AS}$ is asserted, the operand is from external memory. The protocol used for an operand transfer is

1. The CPU sets the CPSTA<1:0> lines to 00 to indicate that the operation is encoded on CPDAT<5:0>.
2. The CPU transfers information related to the operand transfer onto CPDAT<5:0>. The line information is

CPDAT Line	Description
<5:4>	Address alignment code. These are zeros when the operand originates from general registers. They transfer the two low-order address bits of the reference when the operand originates from cache or external memory.
3	0 for an operand transfer.

CPDAT Line	Description
2	0 when PSL 06 is cleared 1 when PSL6 is set
1	0 for no action 1 when DAL<31:00> contains the operand
0	0 for no action 1 when DAL<05:00> is a short literal (DAL<31:06> are zeros.

3. The operand is transferred to the CFPA on DAL<31:00>. The CFPA aligns all unaligned data. When the operand originates from external memory ($\overline{AS}$ asserted), the CFPA reads DAL<31:00> according to the full memory read protocol ($\overline{RDY}$ and/or $\overline{ERR}$ asserted). When the operand originates from the general registers or internal cache memory of the CPU, the data is transferred onto DAL<31:00> at P3 of the cycle and sampled by the FPA at the next P1.
4. If a parity error is detected by the CPU when the source of the operand is either the internal cache memory or external memory, it aborts the FPA operation. The CPU aborts the operation by not informing the CFPA of the current result. The CFPA is reset when the CPU sends a new opcode.

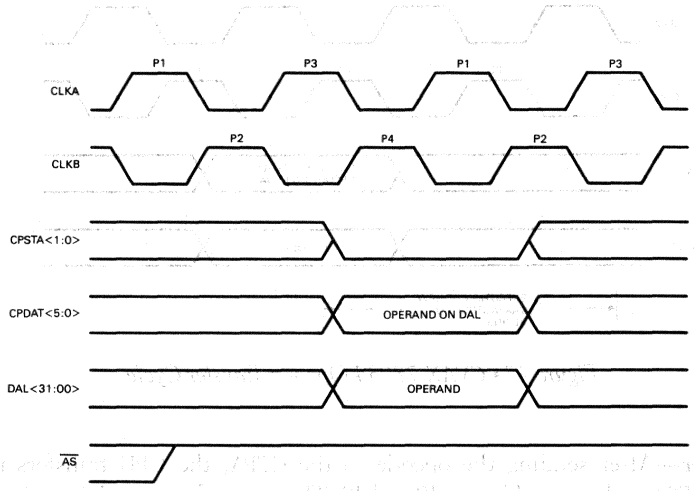


Figure 48 • CVAX 78034 Single-precision CPU to CFPA Transfer

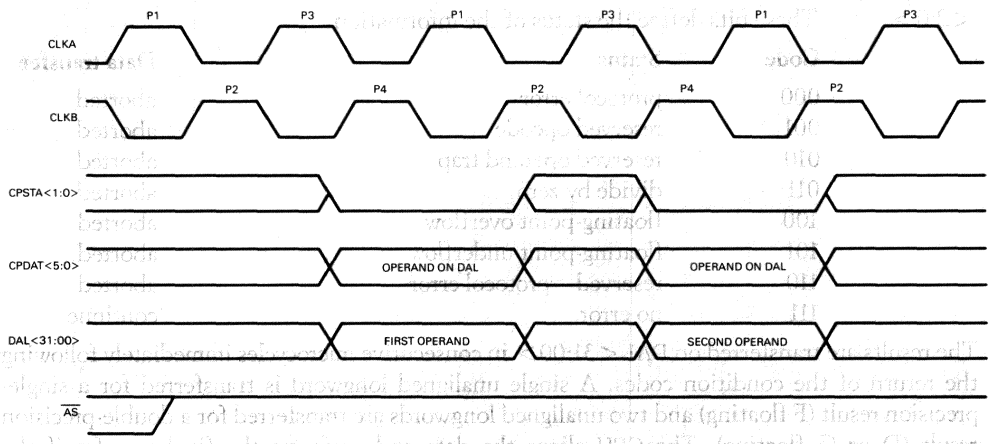


Figure 49 • CVAX 78034 Double-precision CPU to CFPA Transfer

CFPA Result Transfer—After receiving the opcode and operands, the CFPA executes the instruction and transfers condition codes, status information, and result of the computation to the CPU. Figure 50 showsv a single-precision CFPA to CPU transfer, and Figure 51 shows a double-precision CFPA to CPU transfer. The protocol for the transfer is

1. When the CPU is ready for a result it set the CPSTA<1:0> lines to zero and the CPDAT3 line to 1. Ownership of CPSTA<1:0> and CPDAT<5:0> lines is then transferred to the CFPA. The DAL<31:00> are set to a high-impedance state at the next P2 edge.
2. The CFPA gains ownership of the CPSTA<1:0> and CPDAT<5:0> lines by transferring zeros on lines CPSTA<1:0> indicating that the result is not ready and undefined data on CPDAT<5:0> during the next P3 edge. The CFPA continues to transfer zeros on CPSTA<1:0> at each P3 edge. The CPU continuously monitors the CPSTA<1:0> lines until a 11 is present indicating that the result is ready. While waiting for the CFPA to return the result ready condition, the CPU can grant use of the DAL and its associated control signals ($\overline{DMG}$ asserted) to a DMA device. The CPU asserts $\overline{DMG}$ on a P4 edge and stops sampling CPSTA<1:0> until it deasserts $\overline{DMG}$.
3. The CFPA sets the CPSTA<1:0> to 11 and transfers condition codes and status information on lines CPDAT<5:0> on the next P3 edge. If a DMA cycle is in progress or is granted on the following P4 edge, the CFPA repeats the response until $\overline{DMG}$ is deasserted.
4. The CPU reads the CPDAT<5:0> information to determine the response of the FPA, and a DMA request is not granted until the end of the operation. The CPDAT<5:0> lines are encoded as follows:

CPDAT Line	Description
5	0 if the result clears the N bit of the PSL 1 if the result sets the N bit of the PSL
4	0 if the result clears the Z bit of the PSL 1 if the result sets the Z bit of the PSL
3	0 if the result clears the V bit of the PSL 1 if the result sets the V bit of the PSL (integer overflow/ACB condition met)

<2:0> These bits define the status of the information

Code	Status	Data transfer
000	protocol error	aborted
001	reserved opcode	aborted
010	reserved operand trap	aborted
011	divide by zero	aborted
100	floating-point overflow	aborted
101	floating-point underflow	aborted
110	reserved—protocol error	aborted
111	no error	continue

The results are transferred on DAL<31:00> in consecutive microcycles immediately following the return of the condition codes. A single unaligned longword is transferred for a single-precision result (F floating) and two unaligned longwords are transferred for a double-precision result (D or G floating). The CPU aligns the data and performs the final transfer if the destination of the data is memory.

If CPDAT<2:0> indicate a protocol error, reserved opcode, reserved operand trap, divide by zero, floating-point overflow or underflow, no data is transferred. The CFPA will not return a floating-point underflow error if PSL6 is clear.

5. The CFPA sets the CPSTA<1:0> and CPDAT<5:0> lines to a high-impedance state on the next P2 edge. The CPU gains control of CPSTA<1:0> and CPDAT<5:0> on the following P3 edge to complete the transfer.

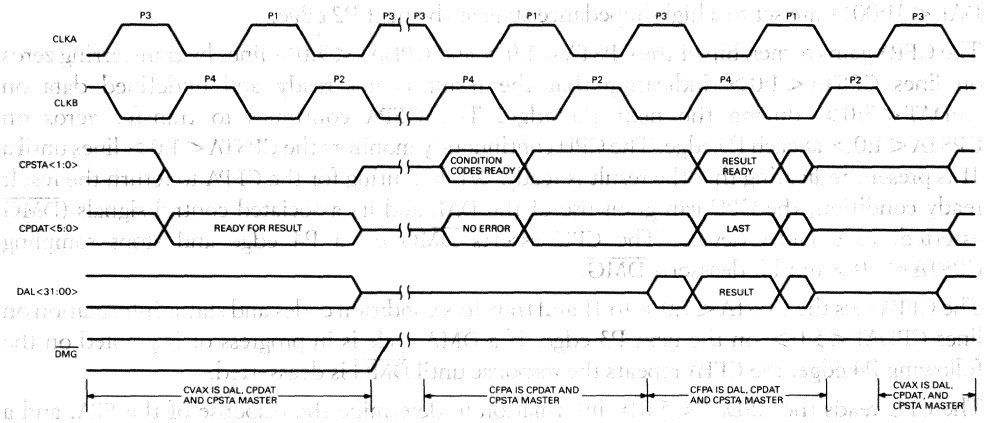


Figure 50 • CVAX 78034 Single-precision CFPA to CPU Transfer

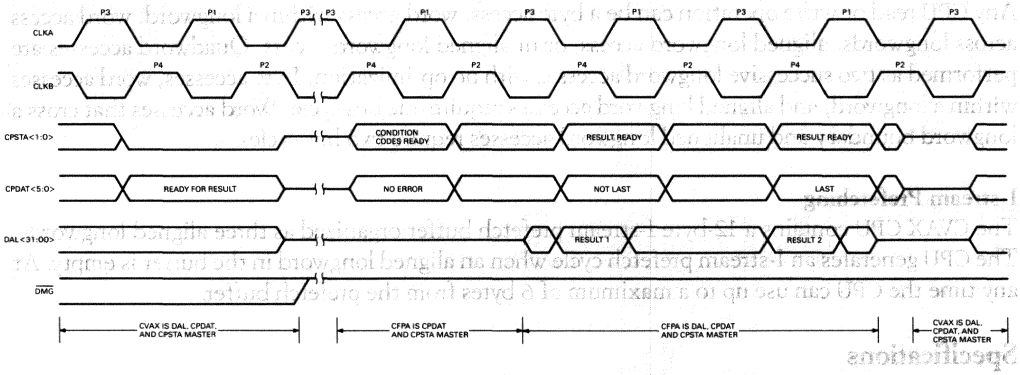


Figure 51 • CVAX 78034 Double-precision CFPA to CPU Transfer

Memory Access Protocol

The 28-bit address provided by the CVAX CPU on DAL<29:02> is a longword address that uniquely identifies one of up to 268,435,456 32-bit memory locations. The CPU provides four byte masks on lines $\overline{BM<3:0>}$ to facilitate byte accesses within 32-bit memory locations. The CPU imposes no restrictions on data alignment. Any data item regardless of size may start at any memory address except for the aligned operands of ADAWI and the interlocked queue instructions.

Memory is viewed as four parallel 8-bit banks each of which receives the longword address DAL<29:02> in parallel. Each bank reads or writes one byte of the data from DAL<31:00> when its byte mask signal is asserted. Figure 52 shows the memory organization.

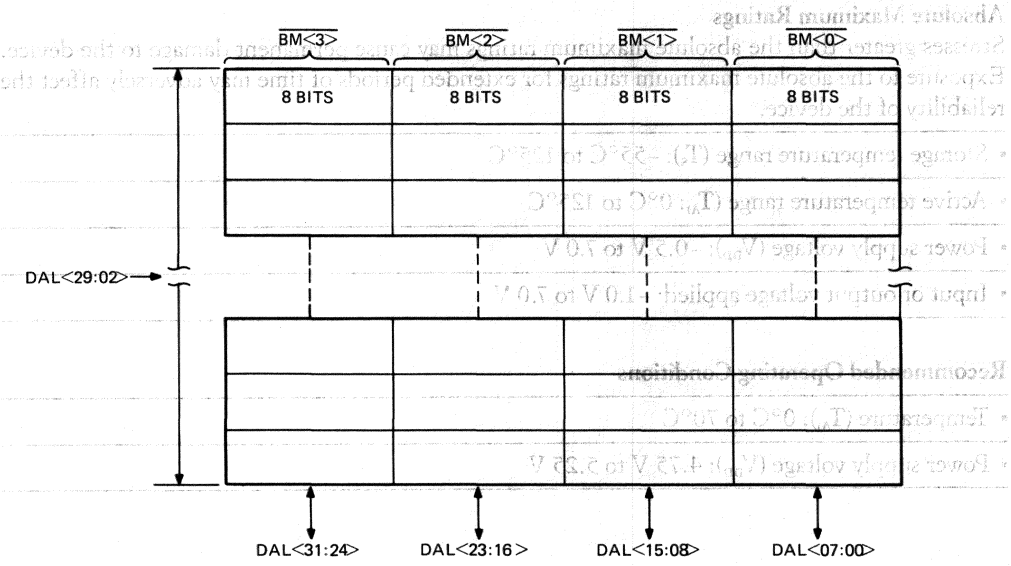


Figure 52 • CVAX 78034 Memory Organization

Any CPU read or write operation can be a byte access, word access within a longword, word access across longwords, aligned longword access, or unaligned longword access. Quadword accesses are performed as two successive longword accesses with no optimization. Byte accesses, word accesses within a longword, and aligned longword accesses require one bus cycle. Word accesses that cross a longword boundary and unaligned longword accesses require two bus cycles.

I-stream Prefetching

The CVAX CPU contains a 12-byte I-stream prefetch buffer organized as three aligned longwords. The CPU generates an I-stream prefetch cycle when an aligned longword in the buffer is empty. At any time the CPU can use up to a maximum of 6 bytes from the prefetch buffer.

Specifications

The mechanical, electrical, and environmental specifications of the CVAX 78034 are contained in the following paragraphs. The test conditions for the values specified are listed as follows unless indicated otherwise.

- Temperature (T_A): 70°C
- Power supply voltage (V_{DD}): 4.75 V
- Ground (V_{SS}): 0

Mechanical Configuration

The physical dimensions of the CVAX 78034 CPU 84-pin surfacemount package are contained in the Appendix.

Absolute Maximum Ratings

Stresses greater than the absolute maximum ratings may cause permanent damage to the device. Exposure to the absolute maximum ratings for extended periods of time may adversely affect the reliability of the device.

- Storage temperature range (T_S): −55°C to 125°C
- Active temperature range (T_A): 0°C to 125°C
- Power supply voltage (V_{DD}): −0.5 V to 7.0 V
- Input or output voltage applied: −1.0 V to 7.0 V

Recommended Operating Conditions

- Temperature (T_A): 0°C to 70°C
- Power supply voltage (V_{DD}): 4.75 V to 5.25 V

dc Electrical Characteristics

The dc input and output parameters are listed in Table 26.

Table 26 • CVAX 78034 dc Input and Output Parameters

Symbol	Parameter	Requirements Min.	Max.	Units	Test Condition
V_{IH}	High-level input voltage (TTL)	2.0	—	V	
V_{IL}	Low-level input voltage (TTL)	—	0.8	V	
V_{OHM}	High-level output voltage (MOS)	90% V_{DD}	—	V	
V_{OLM}	Low-level output voltage (MOS)	—	10% V_{DD}	V	
V_{IHM}	High-level input voltage (MOS)	70% V_{DD}	—	V	
V_{ILM}	Low-level input voltage (MOS)	—	30% V_{DD}	V	
V_{OH}	High-level output voltage	2.4	—	V	I_{OH} = –400 µA
V_{OL}	Low-level output voltage (all pins except $\overline{DBE}$)	—	0.4	V	I_{OL} = 2.0 mA
	$\overline{DBE}$ pin				I_{IL} = 3.0 mA
I_{IL}	Input leakage current	–10	10	µA	$0 < V_{in} < 5.25$ V
I_{OL}	Output leakage current	–10	10	µA	$0 < V_{in} < 5.25$ V
I_{CC}	Active supply current	—	*	mA	I_{out} = 0, T_A = 0°C
C_{in}	Input capacitance	—	*	pF	
C_{out}	Output capacitance	—	*	pF	

*To be determined.

ac Electrical Characteristics

The following notes apply to Figures 53 through 67 and their associated timing tables.

- All times are in nanoseconds (ns) except where noted.
- C_{load} = 130 pF (except for CPDAT<5:0> and CPSTA<1:0>)
- ac highs for MOS inputs are measured at V_{IHM} and lows are measured at V_{ILM}.
- ac highs for MOS outputs are measured at V_{OHM} and lows are measured at V_{OLM}.
- ac highs for TTL inputs are measured at V_{IH} and lows are measured at V_{IL}.

- ac highs for TTL outputs are measured at V_{OH} and lows are measured at V_{OL}.

- MOS inputs are driven to V_{OLM} or V_{OHM} and TTL inputs are driven to V_{OL} or V_{OH}.
  - —$\overline{RDY}$ and $\overline{ERR}$ sampling is performed by the CPU to determine if one of the following bus cycles is to be completed: CPU read cycle, interrupt acknowledge cycle, multiple transfer CPU read cycle, and a CPU write cycle. If $\overline{RDY}$ or $\overline{ERR}$ is not asserted during the sampling window, the bus cycle is extended in increments of one microcycle until $\overline{RDY}$ or $\overline{ERR}$ are asserted. The following restrictions apply to the assertion and deassertion of $\overline{RDY}$ or $\overline{ERR}$ with respect to the sampling window.
  - —Only a high to low transition (assertion) is allowed on $\overline{RDY}$ or $\overline{ERR}$ during a P4 that is part of the sampling window. If the assertion of $\overline{RDY}$ or $\overline{ERR}$ meets setup time t_{SWS}, the CPU recognizes the assertion of the signal. The result of a low-to-high transition (deassertion) of either of these signals during P4 is unpredictable.
  - —If $\overline{RDY}$ or $\overline{ERR}$ is to be recognized by the CPU as deasserted, the signal must be deasserted prior to the P4 that starts the sampling window and held deasserted through the sampling window.
  - —$\overline{RDY}$ and $\overline{ERR}$ can be asserted prior to P4 that starts the sampling window and remain asserted through the sampling window if they are to be recognized by the CPU as asserted.

Clock Timing—Figure 53 shows the timing symbols used to define the CLKA and CLKB clock inputs. The timing parameters are defined in Table 27.

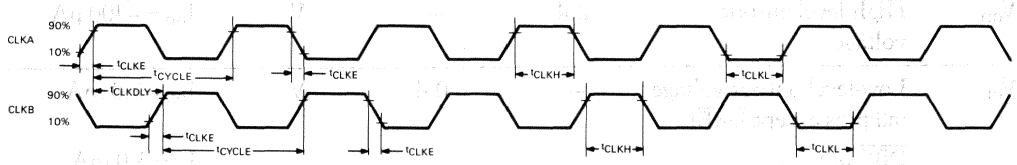


Figure 53 • CVAX 78034 Clock Input Timing

Table 27 • CVAX 78034 Clock Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{CLKDLY}	CLKA to CLKB delay	$t_{CYCLE}/2-2$	$t_{CYCLE}/2+2$
t_{CLKE}	External clock edge rate	0	10
t_{CLKH}	External clock high	5.0	25
t_{CLKL}	External clock low	5.0	25
t_{CYCLE}	External clock cycle	50	*

*To be determined.

Initialization and Reset Timing

The initialization and reset timing sequence is shown in Figure 54. The timing parameters are listed in Table 28.

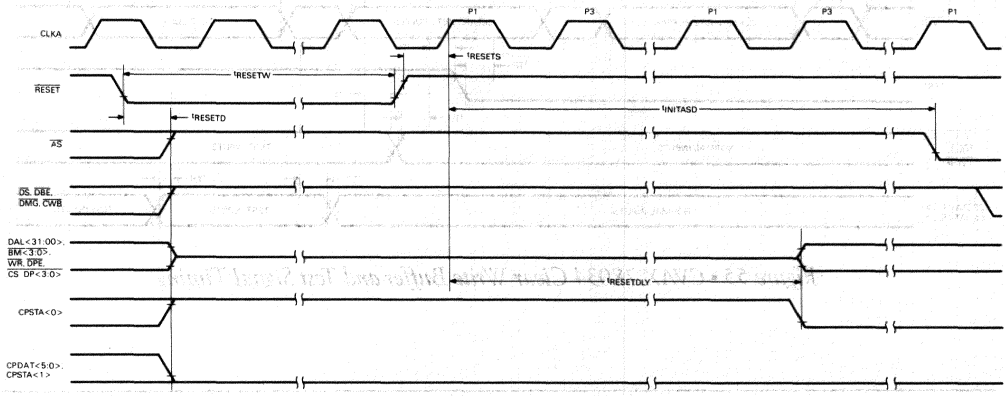


Figure 54 • CVAX 78034 Initialization and Reset Timing

Table 28 • CVAX 78034 Initialization and Reset Timing

Symbol	Definition	Requirements (ns) Min.	Max.
$t_{INITASD}$	First assertion of $\overline{AS}$ after $\overline{RESET}$	$20 \times t_{CYCLE}$	—
t_{RESETD}	Strobe inactive delay from $\overline{RESET}$	0	25
$t_{RESETDLY}$	Output drive from $\overline{RESET}$ deassertion	$7 \times t_{CYCLE}$	$7 \times t_{CYCLE}+20$
t_{RESETS}	$\overline{RESET}$ input setup prior to P1	20	$t_{CYCLE}-10$
t_{RESETW}	$\overline{RESET}$ input width	$10 \times t_{CYCLE}$	—
t_{RESETZ}	Bus high-impedance time from $\overline{RESET}$	0	25

Clear Write Buffer and Test Signal Timing

The Clear Write Buffer (CWB) and TEST signal timing are shown in Figure 55. The signal parameters are listed in Table 29.

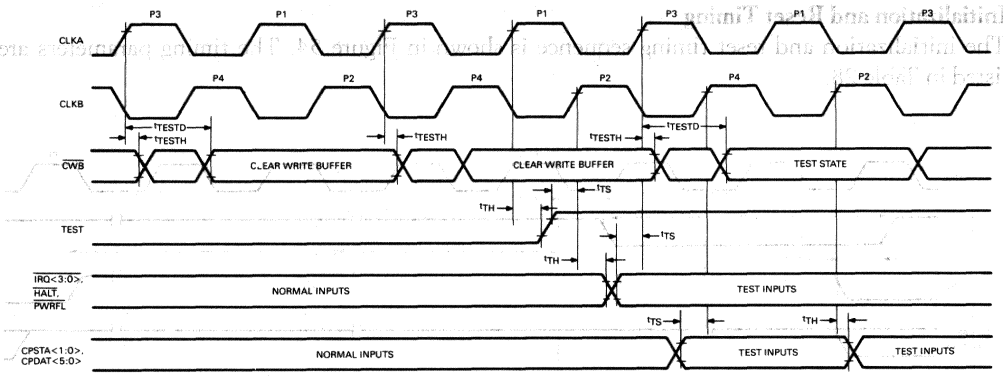


Figure 55 • CVAX 78034 Clear Write Buffer and Test Signal Timing

Table 29 • CVAX 78034 Clear Write Buffer and TEST Signal Parameters

Symbol	Definition*	Requirements (ns) Min.	Max.
t_{TESTD}	$\overline{CWB}$ drive	0	32
t_{TESTH}	$\overline{CWB}$ hold	0	—
t_{TH}	Test input hold	5.0	—
t_{TS}	Test input setup	10	—

External Interrupt Timing

Figure 56 shows external interrupt timing sequence, and Table 30 lists the timing parameters.

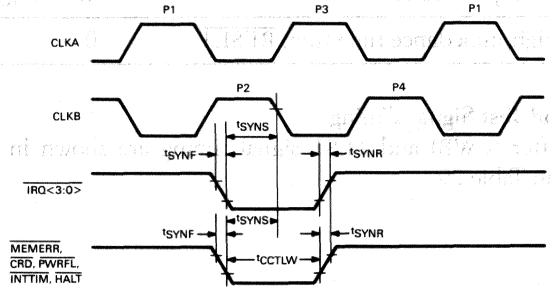


Figure 56 • CVAX 78034 External Interrupt Input Timing

Table 30 • CVAX 78034 External Interrupt Input Timing Parameters

Symbol	Definition*	Requirements (ns) Min.	Max.
t_{CCTLW}	$\overline{CCTL}$ width during cache invalidates	$t_{SYNS} + t_{SYNH}$	—
t_{SYNF}	Asynchronous input fall time	—	15
t_{SYNH}	Asynchronous input hold	15	—
t_{SYNR}	Asynchronous input rise time	—	15
t_{SYNS}	Asynchronous input setup	15	—

*$\overline{IRQ<3:0>}$ are level sensitive and must be asserted for a setup (t_{SYNS}) and hold time (t_{SYNH}) near the end of P2 to assure recognition.

Low going pulses that occur outside the setup and hold window are not recognized.

$\overline{MEMERR}$, $\overline{CRD}$, $\overline{PWRFL}$, $\overline{INTTIM}$, and $\overline{HALT}$ are edge sensitive. The transition from deasserted to asserted must occur one setup time (t_{SYNS}) before the end of P2 to assure recognition; otherwise, recognition is delayed one microcycle.

External DMA Timing

Figure 57 shows the timing sequence for the external DMA signals. Table 31 lists the timing parameters.

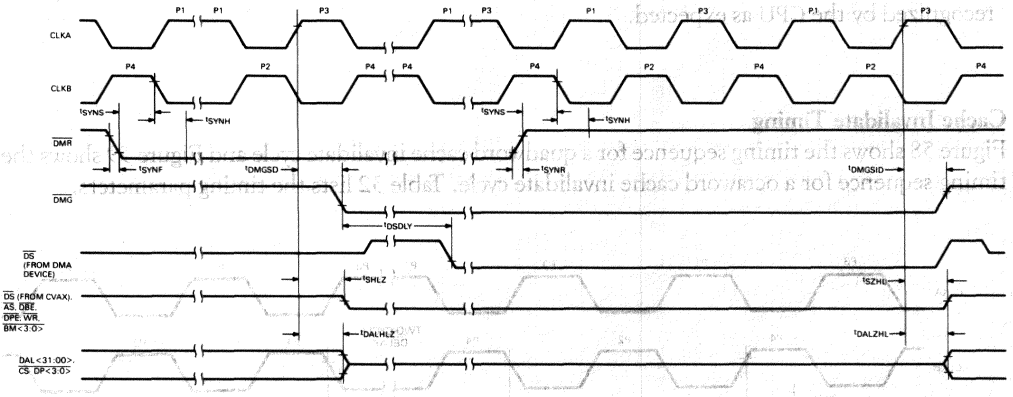


Figure 57 • CVAX 78034 External DMA Timing

Table 31 • CVAX 78034 External DMA Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{DALHLZ}	DAL high-impedance delay	0	20
t_{DALZHL}	DAL active drive delay	0	20
t_{DMGSD}[1]	$\overline{DMG}$ strobe assertion delay	0	20
t_{DMGSID}[2]	$\overline{DMG}$ strobe deassertion delay	0	20
t_{DSDLY}	$\overline{DS}$ delay from receiving $\overline{DMG}$	$3 \times t_{CYCLE}$	—
t_{SHLZ}	Strobe high-impedance delay	0	20
t_{SYNF}	Asynchronous input fall time	—	15
t_{SYNH}[3]	Asynchronous input hold	15	—
t_{SYNR}	Asynchronous input rise time	—	15
t_{SYNS}[3]	Asynchronous input setup	15	—
t_{SZHL}	Strobe active drive delay	0	20

[1] $\overline{DMG}$ is asserted at P4 when $\overline{DMR}$ is asserted eight phases earlier and no CPU I/O cycle has started.

[2] $\overline{DMG}$ is deasserted at P3 when $\overline{DMR}$ is deasserted seven phases earlier.

[3] t_{SYNS} and t_{SYNH} are the setup and hold times needed at a synchronizer to ensure that a signal is recognized by the CPU as expected.

Cache Invalidate Timing

Figure 58 shows the timing sequence for a quadword cache invalidate cycle and Figure 59 shows the timing sequence for a octaword cache invalidate cycle. Table 32 lists the timing parameters.

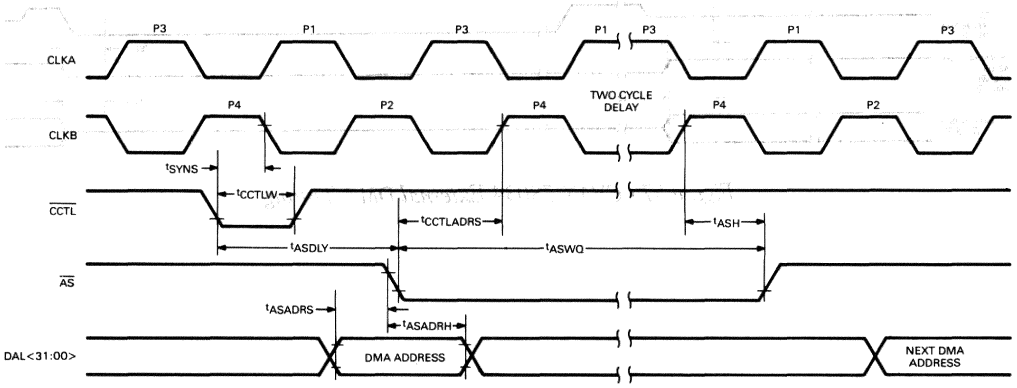


Figure 58 • CVAX 78034 Quadword Cache Invalidate Cycle Timing

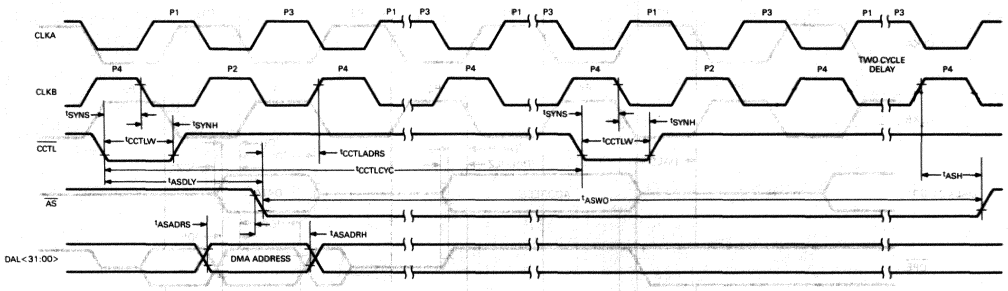


Figure 59 • CVAX 78034 Octaword Cache Invalidate Cycle Timing

Table 32 • CVAX 78034 Cache Invalidate Timing Parameters

Symbol	Definition*	Requirements (ns) Min.	Max.
t_{ASADRH}	DAL hold during cache invalidates	20	—
t_{ASADRS}	DAL setup during cache invalidates	20	—
t_{ASDLY}	$\overline{AS}$ delay from asserting $\overline{CCTL}$ during invalidates	—	[1]
t_{ASH} [2]	$\overline{AS}$ hold during cache invalidates	$t_{CYCLE}/2+5$	—
t_{ASWO}	$\overline{AS}$ width during octaword invalidates	[3]	—
t_{ASWQ}	$\overline{AS}$ width during quadword invalidates	[4]	—
$t_{CCTLADRS}$ [5]	$\overline{AS}$ set up during cache invalidates	20	—
$t_{CCTLCYC}$	$\overline{CCTL}$ cycle time during octaword invalidates	6	—
t_{CCTLW}	$\overline{CCTL}$ width during cache invalidates	$t_{SYNS}+t_{SYNH}$	—
t_{SYNH} [7]	Asynchronous input hold	15	—
t_{SYNS} [7]	Asynchronous input setup	15	—

[1] $2 \times t_{CYCLE} - t_{CLKH}$ (max.) $+ t_{SYNS} - t_{CCTLADRS}$.

[2] t_{ASH} is measured from the third P4 that follows recognition of $\overline{CCTL}$. On octaword invalidate cycles, it is measured from the third P4 that follows the second $\overline{CCTL}$.

[3] $t_{CCTLCYC} - 2 \times t_{ASDLY} + t_{ASWQ} + t_{ASDLY}$ (max.).

[4] $4 \times t_{CYCLE} + t_{CCTLADRS} + t_{ASH} + t_{ASDLY}$ (max.) $- t_{ASDLY}$.

[5] $t_{CCTLADRS}$ is measured from the P4 that follows the recognition of $\overline{CCTL}$. On octaword invalidate cycles, it is measured from the first recognition of $\overline{CCTL}$.

[6] $6 \times t_{CYCLE} + t_{SYNS} + t_{SYNH}$.

[7] t_{SYNS} and t_{SYNH} are the setup and hold times needed at a synchronizer to ensure a signal is recognized by the CPU as expected.

Read and Write Timing

Figure 60 shows the timing sequence for a single-transfer read and interrupt cycle, Figure 61 shows the timing sequence for a multiple-transfer read bus cycle, and Figure 62 shows the timing sequence for CPU write bus cycle. Table 33 lists the read and write cycle timing parameters.

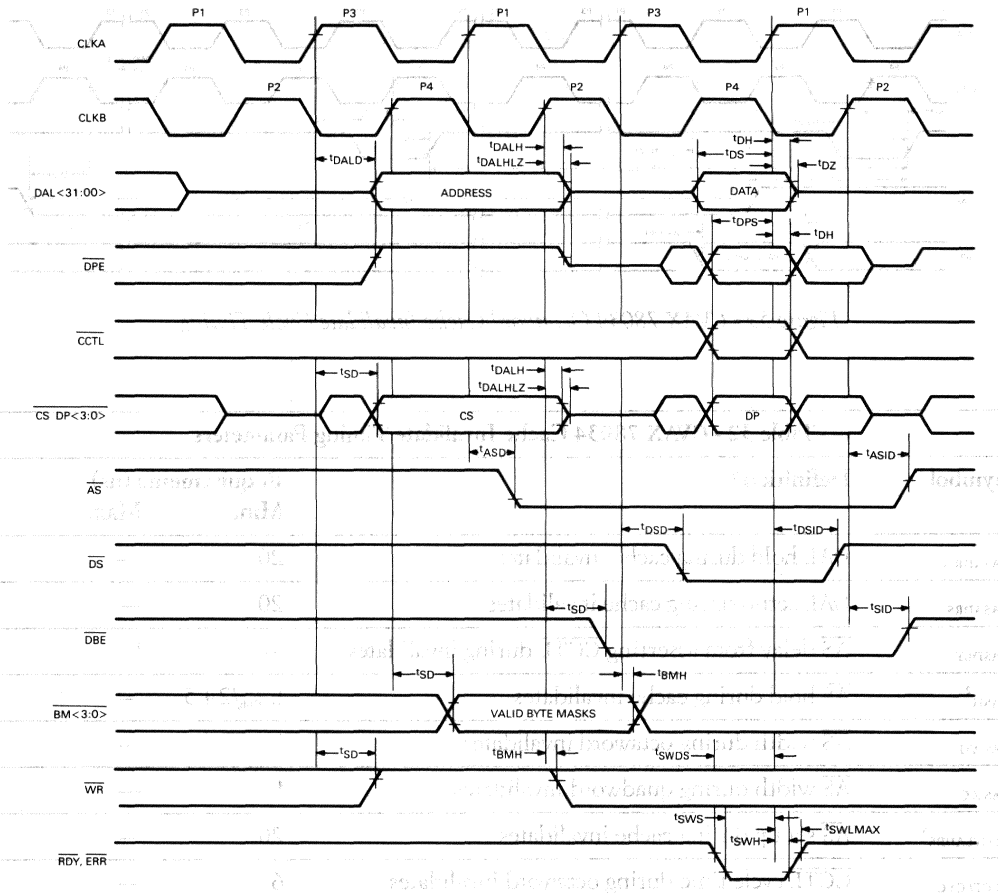


Figure 60 ▪ CVAX 78034 Single-transfer Read and Interrupt Bus Cycle Timing

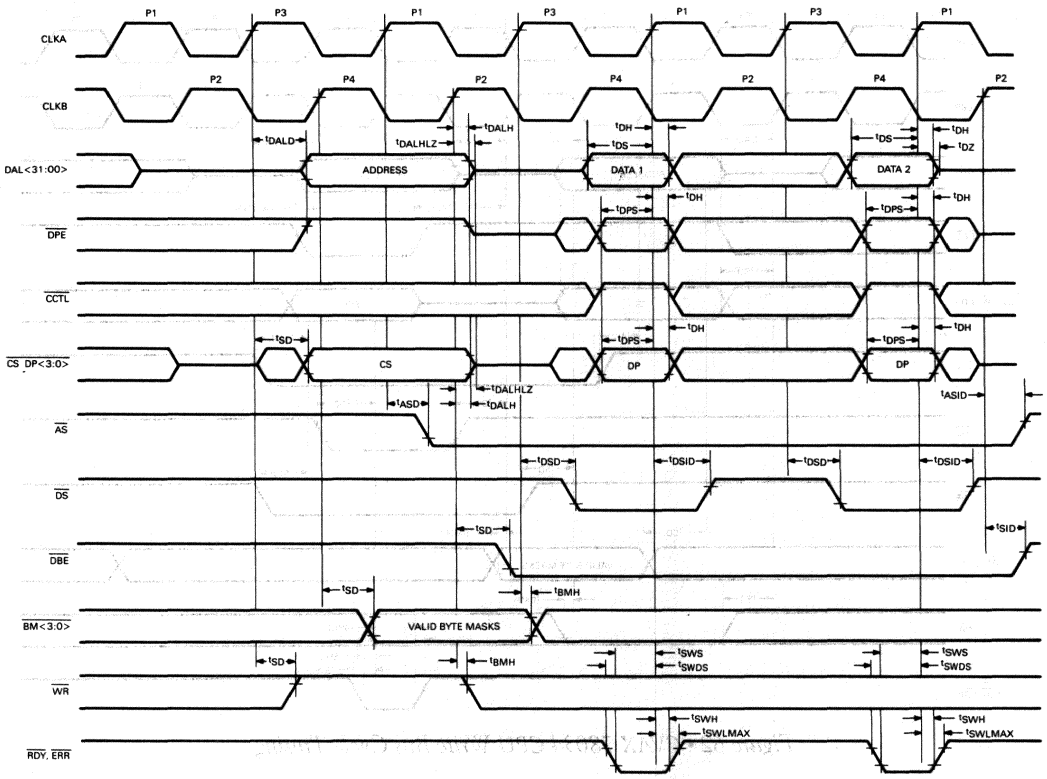


Figure 61 ▪ CVAX 78034 Multiple-transfer Read Bus Cycle Timing

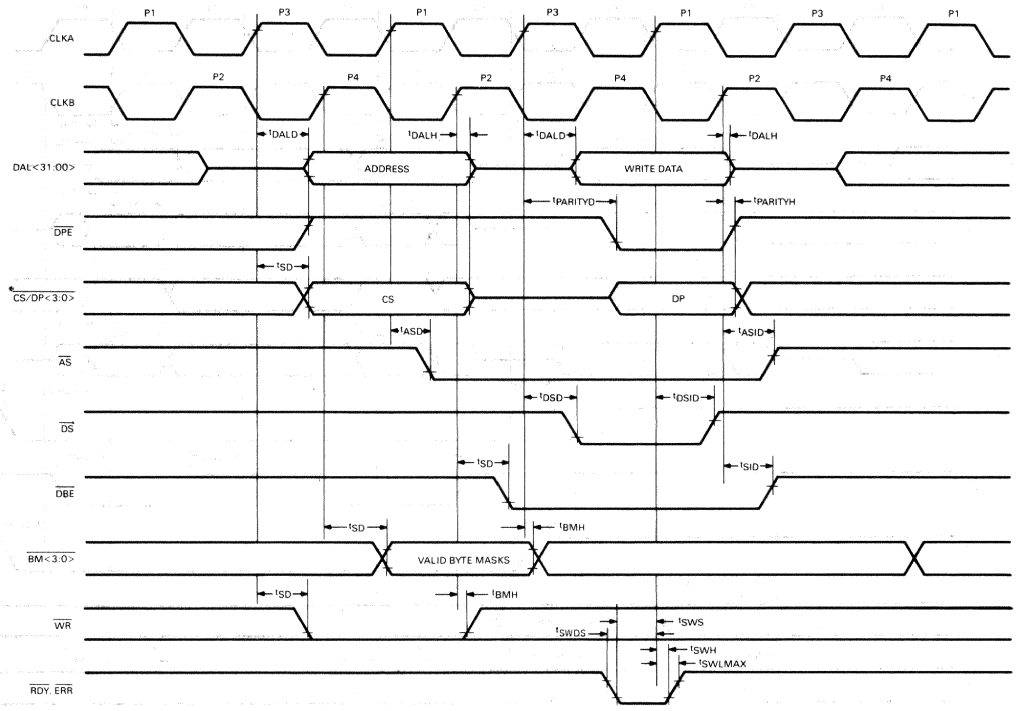


Figure 62 • CVAX 78034 CPU Write Bus Cycle Timing

Table 33 • CVAX 78034 Read and Write Bus Cycle Timing Parameters

Symbol	Definition	Requirements (ns)	
		Min.	Max.
t_{ASD}	$\overline{AS}$ strobe assertion delay	0	15
t_{ASID}	$\overline{AS}$ strobe deassertion delay	0	20
t_{BMH}	$\overline{BM}$ and $\overline{WR}$ hold	0	—
t_{DALD}	DAL drive	0	20
t_{DALH}	DAL hold	5.0	—
t_{DALHLZ}	DAL high-impedance delay	0	20
t_{DH}	DAL hold	5.0	—
t_{DPS}	Parity setup	20	—
t_{DS}	DAL setup	25	—
t_{DSD}	$\overline{DS}$ strobe assertion delay	0	20
t_{DSID}	$\overline{DS}$ strobe deassertion delay	0	18

Symbol	Definition	Requirements (ns) Min.	Max.
t_{DZ}	DAL high impedance	t_{CYCLE}	—
$t_{PARITYD}$	DP drive	0	35
$t_{PARITYH}$	DP hold	0	—
t_{SD}	General strobe assertion delay	0	20
t_{SID}	Strobe deassertion delat	0	20
t_{SWDS}	$\overline{RDY}$ and $\overline{ERR}$ deassertion setup	t_{CLKH}	—
t_{SWH}	$\overline{RDY}$ and $\overline{ERR}$ sample-window hold	5.0	—
t_{SWLMAX}	$\overline{RDY}$ and $\overline{ERR}$ maximum assertion time	—	40
t_{SWS}	$\overline{RDY}$ and $\overline{ERR}$ sample-window setup	15	—

Coprocessor Timing

These following specifications are in effect when the CVAX 78034 CPU is operating with the CVAX 78134 floating-point accelerator (CFPA) coprocessor. Figure 63 shows the timing sequence for the operand transfer cycle. Figure 64 shows the CPU to CFPA timing sequence for single-precision transfers, and Figure 65 shows the CPU to CFPA timing sequence for double-precision transfers. Figures 66 and 67 show the CFPA to CVAX CPU single- and double-precision transfers, respectively. Table 34 lists the coprocessor timing parameters.

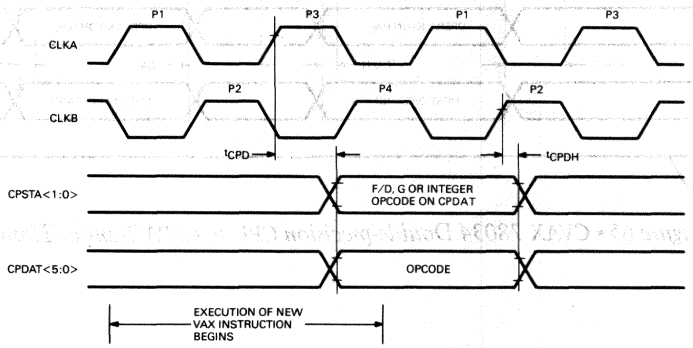


Figure 63 ▪ CVAX 78034 Operand Transfer Cycle Timing

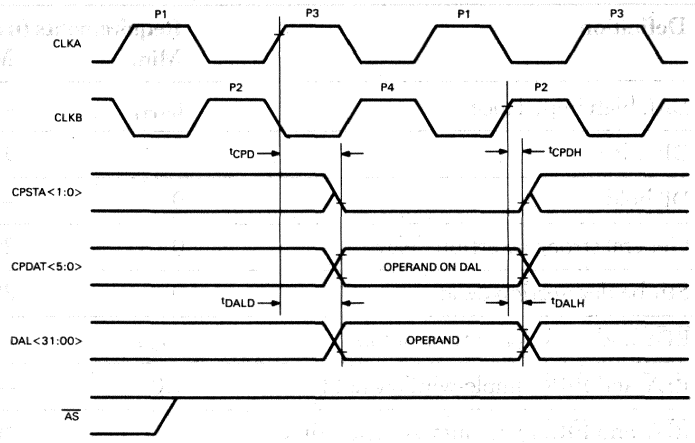


Figure 64 • CVAX 78034 Single-precision CPU to CFPA Transfer Timing

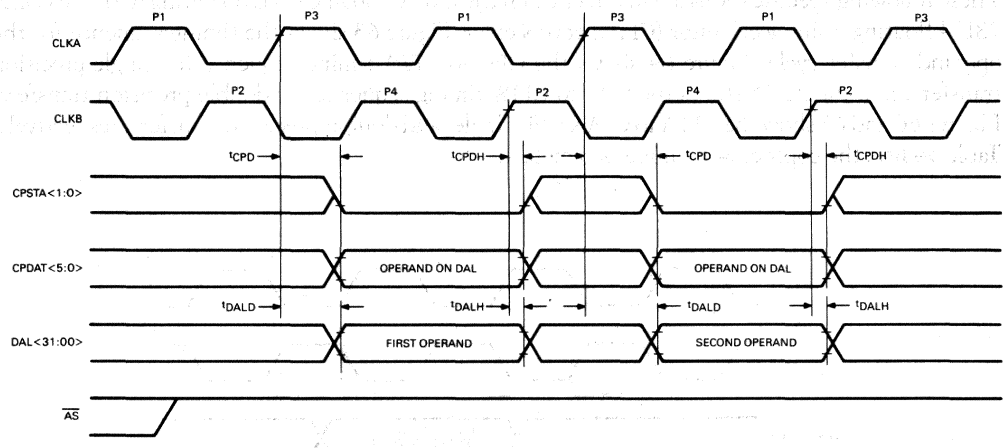


Figure 65 • CVAX 78034 Double-precision CPU to CFPA Transfer Timing

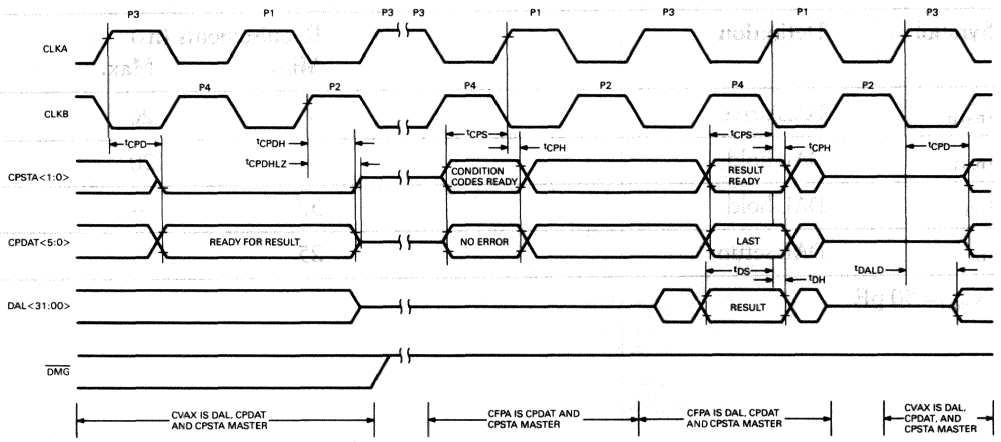


Figure 66 • CVAX 78034 Single-precision CFPA to CPU Transfer Timing

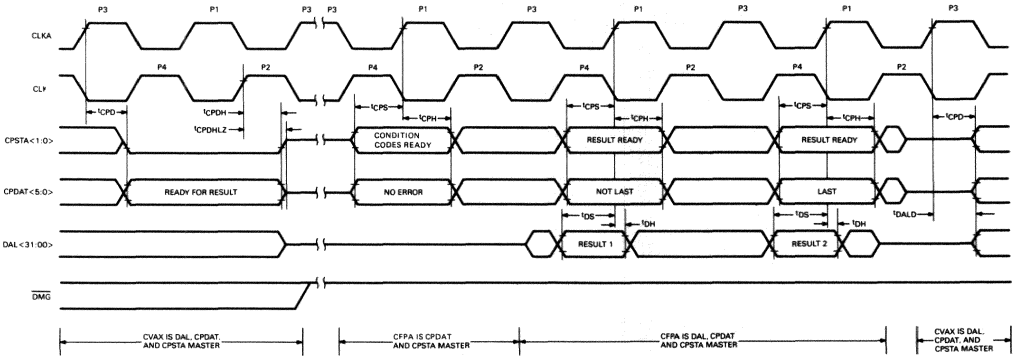


Figure 67 • CVAX 78034 Double-precision CFPA to CPU Transfer Timing

Table 34 • CVAX 78034 Coprocessor Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{CPD}*	Coprocessor line drive	0	20
t_{CPDH}*	Coprocessor line hold	0	—
t_{CPH}*	Coprocessor line hold	23	—
t_{CPHLZ}	Coprocessor high-impedance delay	0	20
t_{CPS}*	Coprocessor line setup	23	—

Symbol	Definition	Requirements (ns) Min.	Max.
t_{DALD}	DAL drive	0	20
t_{DALH}	DAL hold	0	20
t_{DH}	DAL hold	5.0	—
t_{DS}	DAL setup	25	—

*C_{load} = 50 pF.

Features

- High-performance, floating-point processor for use with the CVAX 78034 CPU
- VAX floating-point instruction set (70 instructions)
- Processes VAX integer data types
  —byte
  —word
  —longword
- Processes standard VAX floating-point data types
  —single-precision (F_floating)
  —double-precision (D_floating)
  —extended range double precision (G_floating)
- Enhanced CPU interface
- Single 5-volt power supply

Description

The CVAX 78134 Floating-point Accelerator (CFPA) is a high-performance coprocessor for use with the CVAX 78034 Central Processing Unit (CVAX CPU). The primary purpose of the FPA is to accelerate the execution of floating-point instructions by eliminating the need to emulate them in software. The CFPA handles single-precision, double-precision, and extended-range, double-precision, floating-point data types. The CFPA supports floating-point add, subtract, multiply, divide, convert, and other floating-point operations. It also accelerates the execution of integer multiply and divide operations for longwords only. Figure 1 is a block diagram of the CFPA.

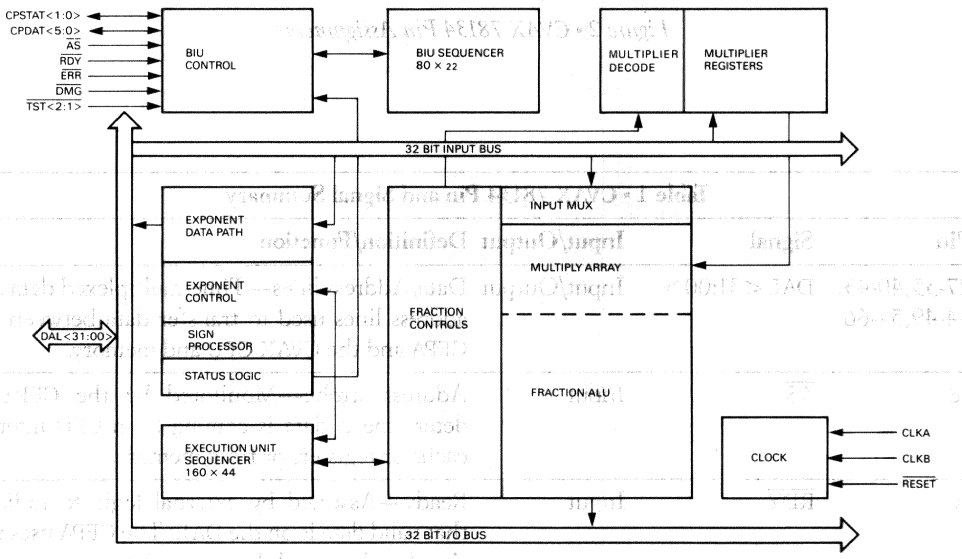


Figure 1 • CVAX 78134 Floating-point Accelerator Block Diagram

• Pin and Signal Descriptions

This section provides a description of the input and output signals and power and ground connections used by the CFPA. The signal pin assignments are identified in Figure 2 and summarized in Table 1.

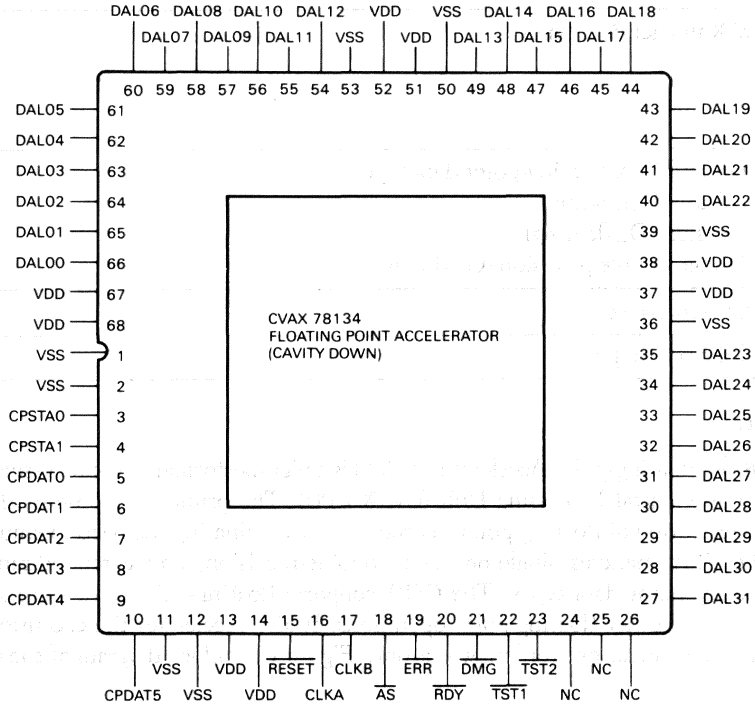


Figure 2 • CVAX 78134 Pin Assignments

Table 1 • CVAX 78134 Pin and Signal Summary

Pin	Signal	Input/Output	Definition/Function
27-35,40-43, 44-49,54-66	DAL<31:00>	Input/Output	Data/Address lines—Time-multiplexed data and address lines used to transfer data between the CFPA and the CVAX CPU and memory.
18	$\overline{AS}$	Input	Address strobe—Monitored by the CFPA to determine if data is coming from CPU internal cache or registers or from memory.
20	$\overline{RDY}$	Input	Ready—Asserted by external logic to indicate that valid data is on the DAL. The CFPA uses this signal to detect valid memory data.

Pin	Signal	Input/Output	Definition/Function
19	$\overline{ERR}$	Input	Error—Asserted by external logic to indicate abnormal termination of the current bus cycle. The CFPA uses this signal to detect faulty memory cycles.
15	$\overline{RESET}$	Input	Reset—Asserted to force the CFPA to its initial powerup state.
21	$\overline{DMG}$	Input	DMA grant—This signal is monitored by the CFPA to determine if a DMA cycle is progress.
10-5	DPDAT<5:0>	Input/Output	Coprocessor data lines—Used to transfer opcode, control information, condition codes, and exception status between the CFPA and the CVAX CPU.
4,3	CPSTA<1:0>	Input/Output	Coprocessor status lines—Used to notify the CFPA or CVAX CPU of the type of information present on CPDAT<5:0>.
16,17	CLKA,CLKB	Input	Clocks—Supply basic clock timing to the CFPA. CLKA and CLKB are nominal 20-MHz, MOS level, square-wave signals that are phase shifted from each other by 180 degrees.
13,14,37,38, 51,52,67,68	V_{DD}	Input	Voltage—5 volt power supply.
1,2,11,12 36,39,50,53	V_{SS}	Input	Ground—Ground reference.
23,22	TST<2:1>	Input	Test 2 and Test 1—Reserved for CFPA manufacturing test.

CVAX Bus and Control

Data And Address Lines (DAL<31:00>)—These are bidirectional time-multiplexed lines used by the CFPA to exchange data with the CVAX CPU. The CFPA receives operands from the CPU or memory over DAL<31:00> and returns the results over these lines.

Address Strobe (AS)—This signal is used by the CFPA to determine if an operand is from the internal registers or cache memory of the CVAX CPU or from external memory. When the operand is from the internal registers or cache memory, the CPU does not assert $\overline{AS}$ during the operand transfer. When the operand is from external memory, the CPU asserts $\overline{AS}$ and the CFPA reads the operand following the normal protocol for a CPU read bus cycle.

Ready (RDY)—This signal is asserted by external logic to indicate that valid data is on DAL<31:00>. The CFPA monitors this signal when an operand comes from external memory to determine if valid data is on DAL<31:00>.

Error (ERR)—This signal is asserted by external logic to indicate abnormal termination of the current bus cycle. The CFPA monitors this signal to detect bad memory references when an operand comes from external memory.

Note

$\overline{RDY}$ and $\overline{ERR}$ must be asserted synchronously with respect to the timing sampling point of the CFPA and must not change during the sample window.

DMA Grant (DMG)—This signal is asserted by the CPU to grant control of the DAL and its associated control signals to external logic. The CFPA monitors this signal to determine if a DMA cycle is in progress.

Coprocessor Signals

Coprocessor Data Lines (CPDAT<5:0>)—The CFPA uses these lines to receive opcode and control information from the CPU and to return condition codes and exception status to the CPU. The CVAX CPU drives these lines when it is not waiting for data to be returned from the CFPA. The CFPA drives these lines after the CVAX CPU indicates it is ready for the result and until the CFPA indicates status ready and DMG is not asserted. The CPDAT<5:0> lines are sampled synchronously by the destination at the beginning of P1.

Coprocessor Status lines (CPSTA<1:0>)—These bidirectional lines are used by the FPA and CPU to determine the interpretation of the contents of CPDAT<5:0>. CPSTA<1:0> are sampled synchronously at the beginning of P1 and indicate the contents of CPDAT<5:0> to the destination processor.

Table 2 lists the function of the coprocessor status information from the CPU to the CFPA. Table 3 lists the function of the coprocessor status information from the CFPA to the CPU.

Table 2 • CVAX 78134 CPU to CFPA Status Line Information

CPSTA <1:0>	Function	CPDAT <5:0>	Description
00	Operation encoded on CPDAT<5:0>*	<5:4>	Address alignment code
		3	CPU ready for result
		2	Floating underflow (PSL6)
		1	Next floating-point operand is on DAL<31:00>
		0	Next floating-point operand is a short literal on DAL<05:00>; DAL<31:06> are zeros.
01	Integer opcode on CPDAT<5:0>	<5:0>	Integer opcode
10	F_/D_floating-point opcode on CPDAT<5:0>	<5:0>	Floating-point opcode
11	G_floating-point opcode on CPDAT<5:0>	<5:0>	Floating-point opcode

*Operand on bus (CPDAT1 = 1) and short literal (CPDAT0 = 1) are mutually exclusive.

Table 3 • CVAX 78134 CFPA to CPU Status Line Information

CPSTA <1:0>	Function	CPDAT <5:0>	Description
00	Result not ready		reserved
01	illegal		
10	illegal		
11	Condition codes ready	5	N condition code
		4	Z condition code
		3	V condition code or ACBf branch
		<2:0>	Status code as follows:
			000 protocol error
			001 illegal opcode
			010 reserved operand trap
			011 divide by zero
			100 floating-point overflow
			101 floating-point underflow
			110 reserved (error)
			111 no fatal error

Power and Clocks

Care must be taken to connect the power and ground pins with the shortest wires or power plane possible.

Voltage (V_{DD})—5-volt power supply.

Ground (V_{SS})—Ground reference.

Clock A In and Clock B In (CLKA,CLKB)—These inputs provide the basic clock timing to the CFPA. CLKA and CLKB are nominally 20-MHz, MOS-level, square-wave signals that are phase shifted from each other by 180 degrees.

Miscellaneous

Reset ($\overline{\text{RESET}}$)—This signal is asserted by external logic to force the CFPA to its initial powerup state. Deassertion of $\overline{\text{RESET}}$ is internally synchronized so that the first rising edge of CLKA that follows the deassertion of $\overline{\text{RESET}}$ corresponds to P1.

Test 2 and Test 1 (TST<2:1>)—These signals are reserved for CFPA manufacturing use.

• Architecture Summary

The following is a brief description of the CFPA architecture. The CFPA has no user-accessible registers or mode bits. The general registers and condition codes are contained in the CPU. Round or truncate operational modes and F_floating or D_floating data types are determined by commands sent from the CPU to the CFPA.

Data Types

The architecture of the CVAX 78134 FPA supports seven data types—byte, word, longword, quadword, F_floating, D_floating, and G_floating. Figures 3 and 4 show the organization of these data types.

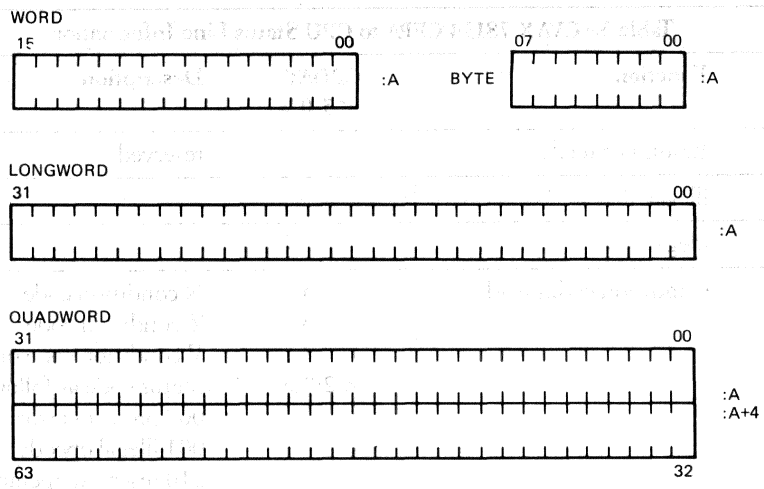


Figure 3 • CVAX 78134 Integer Data Types

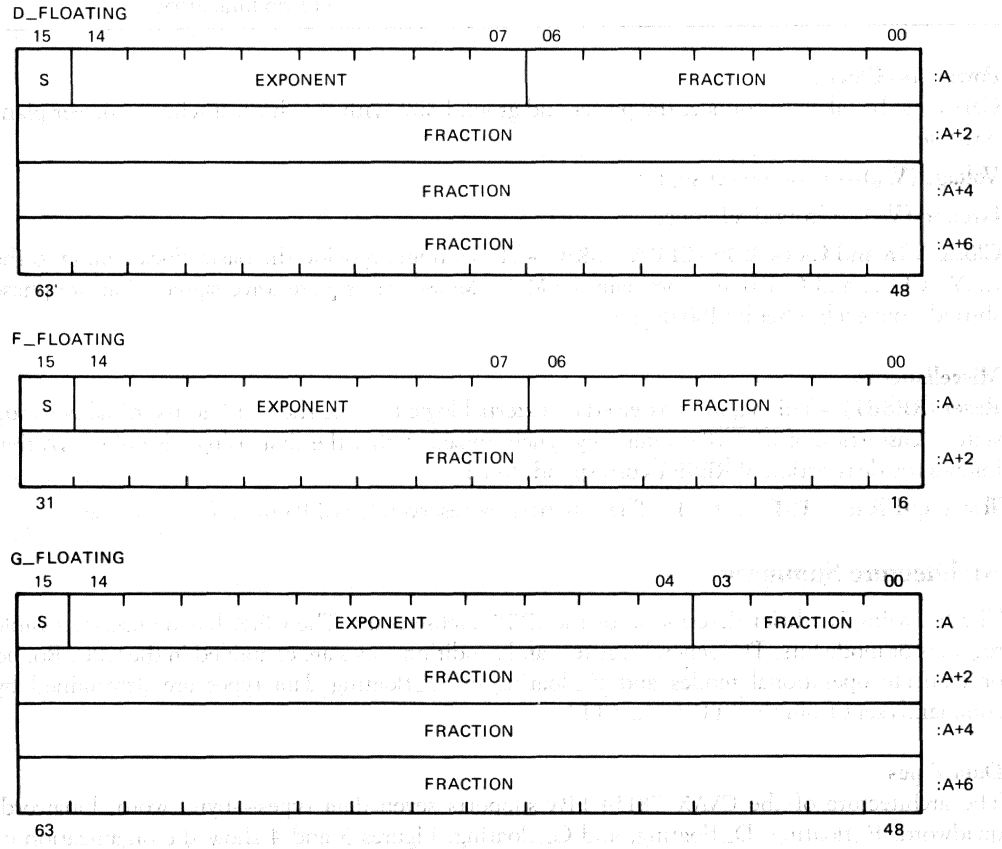


Figure 4 • CVAX 78134 Floating-point Data Types

Instruction Set

The CFPA instruction set consists of 70 floating-point instructions and five integer instructions that provide the following operations.

Addition and Subtraction—For single-precision and double-precision floating-point numbers.

Multiplication and Division—For single-precision and double-precision floating-point numbers and for integers (longwords only).

Conversion—The CFPA performs floating-to-integer and integer-to-floating conversions and double-precision to or from single-precision floating-point conversions.

Comparison—The CFPA has a compare (CMP) and test (TST) instruction associated with each of the three floating-point data types.

Add Compare and Branch—The CFPA assists the CPU in executing an Add Compare and Branch (ACB) instruction by performing the add and compare portions of the instruction. There is an ACB instruction associated with each of the three floating-point data types.

Polynomial Evaluation—The CFPA assists the CPU in executing the polynomial evaluation (POLY) instruction. The CFPA performs the floating-point addition and multiplication operations associated with the polynomial evaluation. There is a POLY instruction for each floating-point data type.

Extended Multiply and Integerize—The CFPA has an extended multiply and integerize (EMOD) instruction associated with each floating-point data type for accurate range reductions of math function arguments.

Instruction Set Notation

The standard notation for operand specifiers is

<name>.<access type><data type>

1. Name is a suggestive name for the operand in the context of the instruction. It is the capitalized name of a register or block for implied operands.
2. Access type is a letter denoting the operand specifier access type.
   a = address operand
   b = branch displacement
   m = modified operand (both read and written)
   r = read-only operand
   w = write-only operand
3. Data type is a letter denoting the data type of the operand.
   b = byte
   d = D_floating
   f = F_floating
   g = G_floating
   l = longword
   q = quadword
   w = word

The abbreviations for condition codes are

* = conditionally set/cleared
\- = not affected
0 = cleared
1 = set

The abbreviations for exceptions are

rsv = reserved operand fault
iov = integer overflow trap
idvz = integer divide by zero trap
fov = floating overflow fault
fuv = floating underflow fault
fdvz = floating divide by zero fault

Integer Instructions

The CFPA accelerates the integer instructions listed in Table 4. The table lists the VAX opcode (hexadecimal) for the instruction, code transferred by the CPSTA<1:0> and CPDAT<5:0> lines, instruction, condition codes affected, and exceptions that can be reported.

Table 4 • CVAX 78134 Integer Instructions

Opcode	CPSTA/CPDAT codes	Instruction	N	Z	V	C	Exceptions
C6	01 000110	DIVL2 divr.rl,quo.ml	*	*	*	0	iov,idvz
C7	01 000111	DIVL3 divr.rl,divd.rl,quo.wl	*	*	*	0	iov,idvz
7A	01 111010	EMUL mulr.rl,muld.rl,add.rl,prod.wq	*	*	0	0	
C4	01 000100	MULL2 mulr.rl,prod.ml	*	*	*	0	iov
C5	01 000101	MULL3 mulr.rl,muld.rl,prod.wl	*	*	*	0	iov

Floating-point Instructions

The CFPA implements all the floating-point instructions for F_floating, D_floating, and G_floating data types except for CLRF which is equivalent to CLRL, and CLRD/G which is equivalent to CLRQ. Table 5 lists the floating-point instructions implemented by the CFPA giving the VAX opcode (hexadecimal), code transferred over the CPSTA<1:0> and CPDAT<5:0> lines, instruction, condition codes affected, and exceptions that can be reported.

Table 5 • CVAX 78134 Floating-point Instructions

Opcode	CPSTA/CPDAT codes	Instruction	N	Z	V	C	Exceptions
6F	10 101111	ACBD limit.rd,add.rd,index.md,displ.bw	*	*	0	–	rsv,fov,fuv
4F	10 001111	ACBF limit.rf,add.rf,index.mf,displ.bw	*	*	0	–	rsv,fov,fuv
4FFD	11 001111	ACBG limit.rg,add.rg,index.mg,displ.bw	*	*	0	–	rsv,fov,fuv
60	10 100000	ADDD2 add.rd,sum.md	*	*	0	0	rsv,fov,fuv
40	10 000000	ADDF2 add.rf,sum.mf	*	*	0	0	rsv,fov,fuv
40FD	11 000000	ADDG2 add.rg,sum.mg	*	*	0	0	rsv,fov,fuv
61	10 100001	ADDD3 add1.rd,add2.rd,sum.wd	*	*	0	0	rsv,fov,fuv
41	10 000001	ADDF3 add1.rf,add2.rf,sum.wf	*	*	0	0	rsv,fov,fuv
41FD	11 000001	ADDG3 add1.rg,add2.rg,sum.wg	*	*	0	0	rsv,fov,fuv
71	10 110001	CMPD src1.rd,src2.rd	*	*	0	0	rsv
51	10 010001	CMPF src1.rf,src2.rf	*	*	0	0	rsv

Opcode	CPSTA/CPDAT codes	Instruction	N	Z	V	C	Exceptions
51FD	11 010001	CMPG src1.rg,src2.rg	*	*	0	0	rsv
6C	10 101100	CVTBD src.rb,dst.wd	*	*	0	0	
4C	10 001100	CVTBF src.rb,dst.wf	*	*	0	0	
4CFD	11 001100	CVTBG src.rb,dst.wg	*	*	0	0	
68	10 101000	CVTDB src.rd,dst.wb	*	*	*	0	rsv,iov
76	10 110110	CVTDF src.rd,dst.wf	*	*	0	0	rsv,fov
6A	10 101010	CVTDL src.rd,dst.wl	*	*	*	0	rsv,iov
69	10 101001	CVTDW src.rd,dst.ww	*	*	*	0	rsv,iov
48	10 001000	CVTFB src.rf,dst.wb	*	*	*	0	rsv,iov
56	10 010110	CVTFD src.rf,dst.wd	*	*	0	0	rsv
99FD	11 011001	CVTFG src.rf,dst.wg	*	*	0	0	rsv
4A	10 001010	CVTFL src.rf,dst.wl	*	*	*	0	rsv,iov
49	10 001001	CVTFW src.rf,dst.ww	*	*	*	0	rsv,iov
48FD	11 001000	CVTGB src.rg,dst.wb	*	*	*	0	rsv,iov
33FD	11 110011	CVTGF src.rg,dst.wf	*	*	0	0	rsv,fov,fuv
4AFD	11 001010	CVTGL src.rg,dst.wl	*	*	*	0	rsv,iov
49FD	11 001001	CVTGW src.rg,dst.ww	*	*	*	0	rsv,iov
6E	10 101110	CVTLD src.rl,dst.wd	*	*	0	0	
4E	10 001110	CVTLF src.rl,dst.wf	*	*	0	0	
4EFD	11 001110	CVTLG src.rl,dst.wg	*	*	0	0	
6D	10 101101	CVTWD src.rw,dst.wd	*	*	0	0	
4D	10 001101	CVTWF src.rw,dst.wf	*	*	0	0	
4DFD	11 001101	CVTWG src.rw,dst.wg	*	*	0	0	
6B	10 101011	CVTRDL src.rd,dst.wl	*	*	*	0	rsv,iov
4B	10 001011	CVTRFL src.rf,dst.wl	*	*	*	0	rsv,iov
4BFD	11 001011	CVTRGL src.rg,dst.wl	*	*	*	0	rsv,iov
66	10 100110	DIVD2 divr.rd,quo.md	*	*	0	0	rsv,fov,fuv,fdvz
46	10 000110	DIVF2 divr.rf,quo.mf	*	*	0	0	rsv,fov,fuv,fdvz
46FD	11 000110	DIVG2 divr.rg,quo.mg	*	*	0	0	rsv,fov,fuv,fdvz
67	10 100111	DIVD3 divr.rd,divd.rd,quo.wd	*	*	0	0	rsv,fov,fuv,fdvz
47	10 000111	DIVF3 divr.rf,divd.rf,quo.wf	*	*	0	0	rsv,fov,fuv,fdvz
47FD	11 000111	DIVG3 divr.rg,divd.rg,quo.wg	*	*	0	0	rsv,fov,fuv,fdvz
74	10 110100	EMODD mulr.rd,mulrx.rb,muld.rd, int.wl,fract.wd	*	*	*	0	rsv,fov,fuv,iov
54	10 010100	EMODF mulr.rf,mulrx.rb,muld.rf, int.wl,fract.wf	*	*	*	0	rsv,fov,fuv,iov
54FD	11 010100	EMODG mulr.rg,mulrx.rw, muld.rg,int.wl,fract.wg	*	*	*	0	rsv,fov,fuv,iov
72	10 110010	MNEGD src.rd,dst.wd	*	*	0	0	rsv
52	10 010010	MNEGF src.rf,dst.wf	*	*	0	0	rsv
52FD	11 010010	MNEGG src.rg,dst.wg	*	*	0	0	rsv
70	10 110000	MOVD src.rd,dst.wd	*	*	0	–	rsv
50	10 010000	MOVF src.rf,dst.wf	*	*	0	–	rsv
50FD	11 010000	MOVG src.rg,dst.wg	*	*	0	–	rsv
64	10 100100	MULD2 mulr.rd,prod.md	*	*	0	0	rsv,fov,fuv
44	10 000100	MULF2 mulr.rf,prod.mf	*	*	0	0	rsv,fov,fuv

Opcode	CPSTA/CPDAT codes	Instruction	N	Z	V	C	Exceptions
44FD	11 000100	MULG2 mulr.rg,prod.mg	*	*	0	0	rsv,fov,fuv
65	10 100101	MULD3 mulr.rd,muld.rd,prod.wd	*	*	0	0	rsv,fov,fuv
45	10 000101	MULF3 mulr.rf,muld.rf,prod.wf	*	*	0	0	rsv,fov,fuv
45FD	11 000101	MULG3 mulr.rg,muld.rg,prod.wg	*	*	0	0	rsv,fov,fuv
75	10 110101	POLYD arg.rd,degree.rw,table.ab	*	*	0	0	rsv,fov,fuv
55	10 010101	POLYF arg.rf,degree.rw,table.ab	*	*	0	0	rsv,fov,fuv
55FD	11 010101	POLYG arg.rf,degree.rw,table.ab	*	*	0	0	rsv,fov,fuv
62	10 100010	SUBD2 sub.rd,dif.md	*	*	0	0	rsv,fov,fuv
42	10 000010	SUBF2 sub.rf,dif.mf	*	*	0	0	rsv,fov,fuv
42FD	11 000010	SUBG2 sub.rg,dif.mg	*	*	0	0	rsv,fov,fuv
63	10 100011	SUBD3 sub.rd,min.rd,dif.wd	*	*	0	0	rsv,fov,fuv
43	10 000011	SUBF3 sub.rf,min.rf,dif.wf	*	*	0	0	rsv,fov,fuv
43FD	11 000011	SUBG3 sub.rg,min.rg,dif.wg	*	*	0	0	rsv,fov,fuv
73	10 110011	TSTD src.rd	*	*	0	0	rsv
53	10 010011	TSTF src.rf	*	*	0	0	rsv
53FD	11 010011	TSTG src.rg	*	*	0	0	rsv

Instruction Processing

During normal operations, the opcode and all operands associated with the instruction to be executed are transferred to the CFPA. The CFPA executes the instruction and returns the status including all errors and the results. The exceptions to this general case are described as follows.

Integer divide (DIVL2, DIVL3)—During integer divide instruction, the CPU detect and reports a divide by zero condition. It does not request a result and the CFPA will abort the integer divide operation.

Floating compare (CMPD, CMPF, CMPG)—During a floating compare instruction, the only result transferred is the status of the PSL condition codes. To maintain the normal return result protocol, the CFPA will return a longword result. The CVAX CPU should discard this longword as its contents are unpredictable.

Floating Add Compare and Branch (ACBF, ACBD, ACBG)—During a floating add compare and branch instruction, the CFPA reports, in addition to the normal result, whether the branch should be taken. This is encoded in bit 3 of the returned status. Bit 3 is normally used to report integer overflow, which cannot occur on a floating add, compare, and branch. After testing this bit, the CPU must ensure that the V bit in its PSL is cleared.

Extended Modulus (EMODF, EMODD, EMODG)—The extended modulus instructions compute two results. The CFPA returns the integer result followed by the floating result. Therefore, EMODF returns two longwords and EMODD and EMODG return three longwords.

Polynomial Evaluation (POLYF, POLYD, POLYG)—The CFPA supports the polynomial evaluation instructions by implementing a POLY step function. Given the argument and the current partial result, the CFPA reads the new coefficient, computes the partial result, and returns status and the new partial result to the CPU.

The protocol for the startup and poly step loop phase between the CVAX CPU and the CFPA for POLYF, POLYD, and POLYG instructions is described. After the setup phase, the CPU and CFPA enter the POLY STEP loop. The CPU records the loop count using the degree operand.

Startup Phase

1. The CPU sends the opcode for POLYF, POLYD, or POLYG to the CFPA.
2. The CPU sends the argument operand to the CFPA.
3. The CPU sends the degree operand to the CFPA which checks for a reserved operand (degree GTR 31). If found, the CFPA returns reserved operand status when the CPU indicates it is ready for the results.
4. The CPU does not send the table address operand to the CFPA.
5. The CPU indicates when it is ready for a result. The CFPA responds by transferring status and a result equal to the argument operand. If the argument was a short literal, the CFPA returns the argument in expanded form: one longword for POLYF, two longwords for POLYD or POLYG.
6. The CPU sends the seed partial result to the CFPA. If the instruction is being started, the seed will be zero; if the instruction is being restarted, the seed will be the last partial result.

POLY STEP Loop

1. The CPU sends the new coefficient to the CFPA. The FPA checks the new coefficient for a reserved operand. The FPA computes new partial result which is equal to the current partial result * argument) plus coefficient.
2. The CPU indicates when it is ready for the result. The CFPA responds by transferring status and the new partial result. The status includes the reserved operand check on the coefficient and any errors from the polynomial step computation.
3. The FPA executes the POLY STEP loop until a new opcode is received.

• Specifications

The mechanical, electrical and environmental specifications of the CFPA are contained in the following paragraphs. The test conditions for the values specified are listed as follows unless indicated otherwise.

- Temperature (T_A): 70°C
- Power supply voltage (V_{DD}): 4.75 V
- Ground (V_{SS}): 0 V

Mechanical Configuration

The physical dimensions of the CVAX 78135 44-pin surfacemount cerquad package are contained in the Appendix.

Absolute Maximum Ratings

Stresses greater than the absolute maximum ratings may cause permanent damage to the device. Exposure to the absolute maximum ratings for extended periods of time adversely affect the reliability of the device.

- Storage temperature range (T_S): −55°C to 125°C
- Active temperature range (T_A): 0°C to 125°C
- Power supply voltage (V_{DD}): −0.5 V to * V
- Input or output voltage applied: −1.0 V to * V

*To be determined.

Recommended Operating Conditions

- Temperature (T_A): 0°C to 70°C
- Power supply voltage (V_{DD}) 4.5 V to 5.5 V

dc Electrical Characteristics

The dc input and output parameters are listed in Table 6.

Table 6 • CVAX 78134 dc Input and Output Parameters

Symbol	Parameter	Requirements Min.	Max.	Units	Test Condition
V_{IH}	High-level input voltage (TTL)	2.0	—	V	
V_{IL}	Low-level input voltage (TTL)	—	0.8	V	
V_{OH}	High-level output voltage (TTL)	2.4	—	V	$I_{OH} = -400\ \mu A$
V_{OL}	Low-level output voltage (TTL)	—	0.4	V	$I_{OL} = 2.0$ mA
V_{IHM}	High-level input voltage (MOS)	70% V_{DD}	—	V	
V_{ILM}	Low-level input voltage (MOS)	—	30% V_{DD}	V	
V_{OHM}	High-level output voltage (MOS)	90% V_{DD}	—	V	$I_{OL} = -1.0$ mA
V_{OLM}	Low-level output voltage (MOS)	—	0.4	V	$I_{OL} = 1.0$ mA
I_{IL}	Input leakage current	-20	20	µA	$0 < V_{in} < 5.25$ V
I_{OZ}	Output leakage current	-20	20	µA	$0 < V_{in} < 5.25$ V
I_{CC}	Active supply current	—	200	mA	$I_{out} = 0$ $T_A = 0°C$
C_{in}	Input capacitance	—	*	pF	
C_{out}	Output capacitance	—	*	pF	

*To be determined.

ac Electrical Characteristics

The following notes apply to Figures 5 through 12 and their associated timing tables.

- All times are in nanoseconds (ns) except where noted.
- $C_{load} = 130$ pF (except for CPDAT<5:0> and CPSTA<1:0>)

- ac highs for MOS inputs are measured at V_{IHM} and lows are measured at V_{ILM}.
- ac highs for MOS outputs are measured at V_{OHM} and lows are measured at V_{OLM}.
- ac highs for TTL inputs are measured at V_{IH} and lows are measured at V_{IL}.
- ac highs for TTL outputs are measured at V_{OH} and lows are measured at V_{OL}.
- MOS inputs are driven to V_{OLM} or V_{OHM} and TTL inputs are driven to V_{OL} or V_{OH}.

Clock Input

Figure 5 shows the clock input timing and the parameters are listed in Table 7.

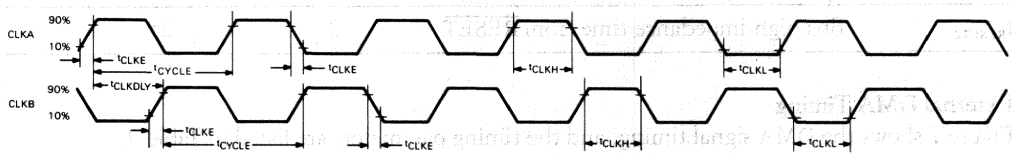


Figure 5 • CVAX 78134 Clock Input Timing

Table 7 • CVAX 78134 Clock Input Timing Parameters

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{CLKDLY}	CLKA to CLKB delay (nominal)	$t_{CYCLE}/2-2$ 23	$t_{CYCLE}/2+2$ 27
t_{CLKH}	External clock high	5.0	25
t_{CLKL}	External clock low	5.0	25
t_{CLKE}	External clock edge rate	0	10
t_{CYCLE}	External clock cycle	50	*

*To be determined.

Initialization

Figure 6 shows the initialization timing and the parameters are listed in Table 8.

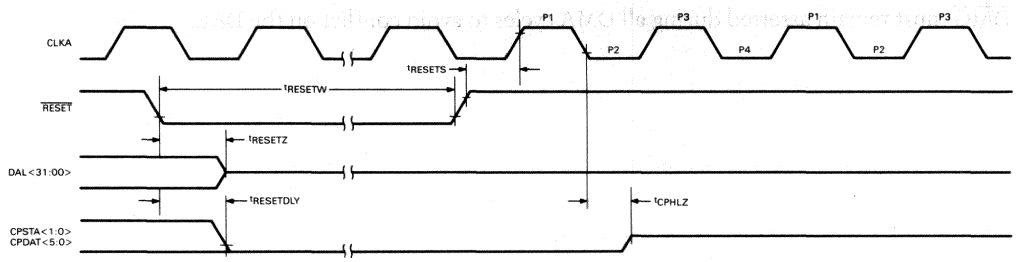


Figure 6 • CVAX 78134 Initialization Timing

Table 8 • CVAX 78134 Initialization Timing Parameters

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{CPHLZ}	Coprocessor high-impedance delay	0	20
$t_{RESETDLY}$	Output drive from $\overline{RESET}$ assertion	0	25
t_{RESETS}	$\overline{RESET}$ input setup prior to P1 (nominal)	20 —	$t_{CYCLE}-10$ 40
t_{RESETW}	$\overline{RESET}$ input width (nominal)	$10 \times t_{CYCLE}$ 500	— —
t_{RESETZ}	Bus high-impedance time from $\overline{RESET}$	0	25

External DMA Timing

Figure 7 shows the DMA signal timing, and the timing parameters are listed in Table 9.

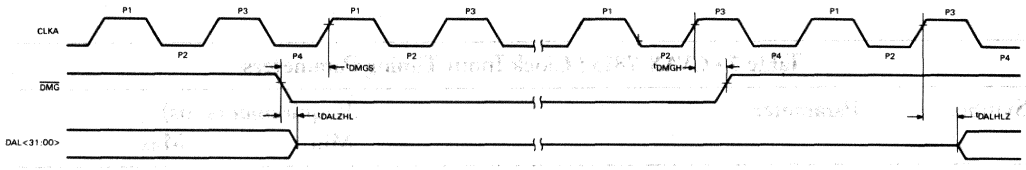


Figure 7 • CVAX 78134 External DMA Timing

Table 9 • CVAX 78134 External DMA Timing Parameters

Symbol	Parameter*	Requirements (ns) Min.	Max.
t_{DALHLZ}	DAL high-impedance delay	0	20
t_{DALZHL}	DAL active drive delay	0	20
t_{DMGH}	DMG hold	5.0	—
t_{DMGS}	DMG setup	25	—

*$\overline{DMG}$ must remain asserted during all DMA cycles to avoid conflict on the DAL.

Coprocessor Timing

Figures 8 through 12 show the timing of the transfers between the CPU and CFPA, and the timing parameters are listed in Table 10.

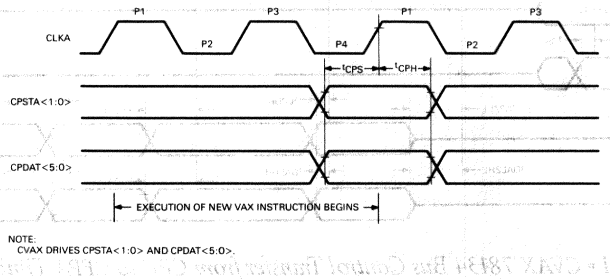


Figure 8 ▪ CVAX 78134 Opcode Transfer Timing

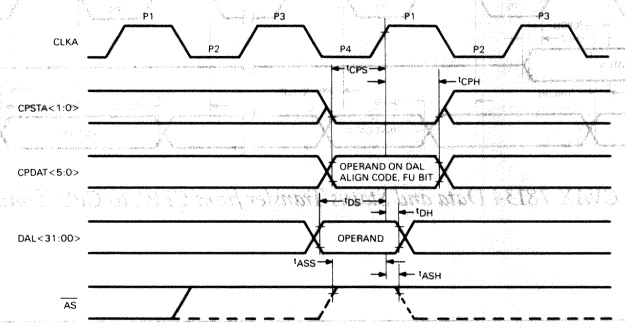


Figure 9 ▪ CVAX 78134 CPU to CFPA Operand Transfer (AS not asserted) Timing

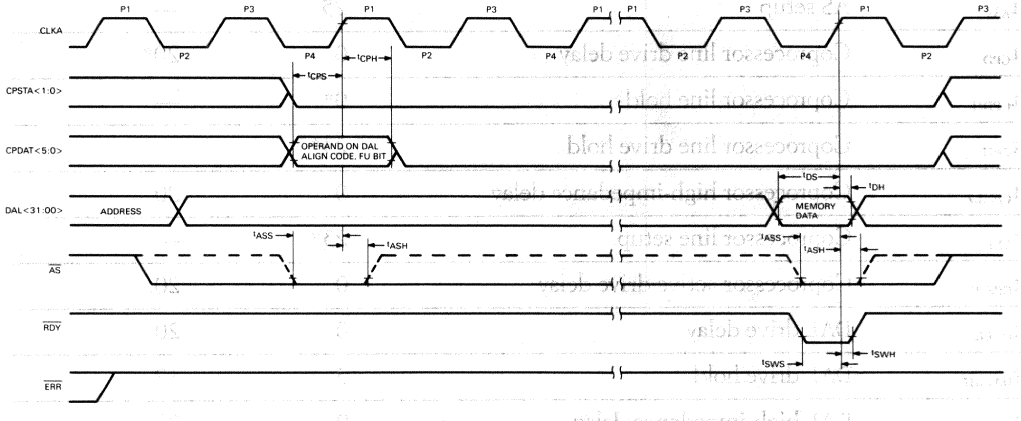


Figure 10 ▪ CVAX 78134 CPU to CFPA Operand Transfer (AS asserted) Timing

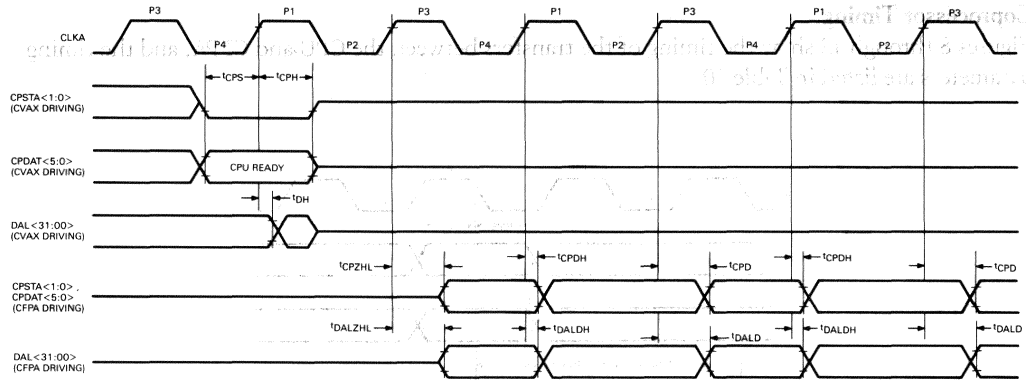


Figure 11 • CVAX 78134 Bus Control Transfer from CPU to CFPA Timing

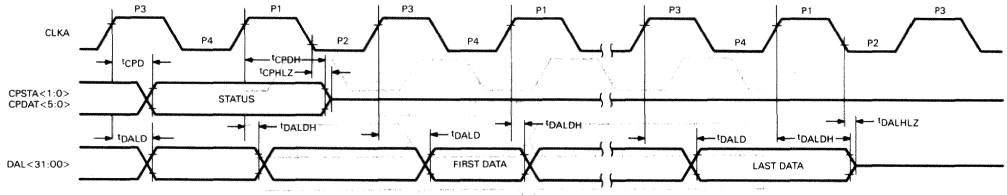


Figure 12 • CVAX 78134 Data and Status Transfer from CFPA to CPU Timing

Table 10 • CVAX 78134 Coprocessor Timing Parameters

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{ASH}	$\overline{AS}$ hold	5	—
t_{ASS}	$\overline{AS}$ setup	25	—
t_{CPD}	Coprocessor line drive delay	0	20*
t_{CPDH}	Coprocessor line hold	0*	—
t_{CPH}	Coprocessor line drive hold	23*	—
t_{CPHLZ}	Coprocessor high-impedance delay	0	20
t_{CPS}	Coprocessor line setup	23*	—
t_{CPZHL}	Coprocessor active drive delay	0	20
t_{DALD}	DAL drive delay	0	20
t_{DALDH}	DAL drive hold	5	32
t_{DALHLZ}	DAL high-impedance delay	0	20

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{DALZHL}	DAL active drive delay	0	20
t_{DH}	DAL hold	5	—
t_{DS}	DAL setup	25	—
t_{SWH}	$\overline{RDY}$ and $\overline{ERR}$ sample window hold	5	—
t_{SWS}	$\overline{RDY}$ and $\overline{ERR}$ sample window setup	15	—

*C_{load} = 150 pF.

• Features

- Standard TTL oscillator input
- Generates two 180-degree out-of-phase precision MOS clock signals for CVAX 78034 CPU, CVAX 78134 FPA, and CVAX support chips
- Generates two auxiliary 180-degree out-of-phase precision clocks for use by system interface
- Generates a single auxiliary MOS clock in phase with the CPU microcycle
- Generates proper timing and synchronization for $\overline{\text{RESET}}$ signal
- Provides common synchronization point for the $\overline{\text{RDY}}$ and $\overline{\text{ERR}}$ signal allowing the system interface to operate asynchronously
- Provides board level tester support via $\overline{\text{TEST}}$ and TCLKIN
- Requires single 5-volt supply

• Description

The CVAX 78135 Clock (CCLOCK), contained in a 44-pin surfacemount package, generates the precision MOS clock signals required by the CVAX 78034 CPU, CVAX 78134 FPA, and up to two additional support chips. The CCLOCK also generates three auxiliary clock signals that can be used by the CVAX system interface and provides timing and synchronization for the $\overline{\text{RESET}}$, $\overline{\text{RDY}}$, and $\overline{\text{ERR}}$ signals. Figure 1 is the block diagram for the CVAX 78135 clock.

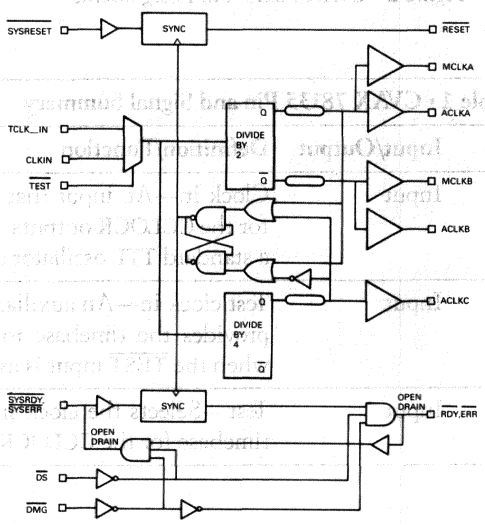


Figure 1 • CVAX 78135 Clock Generator Logic Diagram

• Pin Descriptions

This section provides a brief description of the input and output signals and power and ground connections of the CVAX 78135 clock generator. The pin assignments are identified in Figure 2 and the signals are summarized in Table 1.

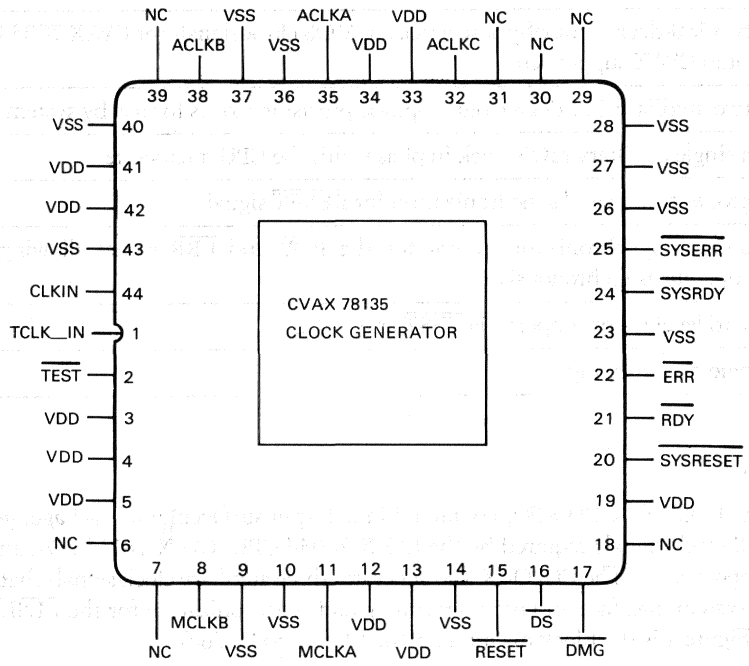


Figure 2 • CVAX 78135 Pin Assignments

Table 1 • CVAX 78135 Pin and Signal Summary

Pin	Signal	Input/Output	Definition/Function
44	CLKIN	Input	Clock in—An input that provides the timebase for the CCLOCK outputs. This input is driven by a standard TTL oscillator output.
1	TCLK IN	Input	Test clock in—An auxiliary test input clock that provides the timebase to all CCLOCK outputs when the $\overline{\text{TEST}}$ input is asserted.
2	$\overline{\text{TEST}}$	Input	Test—Selects the clock input to be used as the timebase for the CCLOCK.

Pin	Signal	Input/Output	Definition/Function
11,8	MCLKA, MCLKB	Output	Master clock A and Master clock B—These outputs provide the timebase to CVAX 78034 CPU and support chips. The frequency of both clocks is one-half of the frequency of CLKIN input. MCLKA is phase shifted by 180 degrees from MCLKB.
35,38	ACLKA, ACLKB	Output	Auxiliary clock A and auxiliary clock B—These outputs provide a general purpose timebase. ACLKA and ACLKB are the same frequency as MCLKA and are synchronized with MCLKA and MCLKB, respectively.
32	ACLKC	Output	Auxiliary clock C—This output is one-fourth of the frequency of CLKIN and is synchronized to MCLKA and ACLKA.
20	$\overline{\text{SYSRESET}}$	Input	System reset—An asynchronous system reset signal.
24	$\overline{\text{SYSRDY}}$	Input/Output	System ready—An asynchronous system ready signal.
25	$\overline{\text{SYSERR}}$	Input/Output	System error—An asynchronous system error signal.
15	$\overline{\text{RESET}}$	Output	Reset—A synchronized reset signal for the CVAX 78034 CPU and its associated support chips.
21	$\overline{\text{RDY}}$	Input/Output	Ready—A synchronized ready signal for the CVAX 78034 CPU and its associated support chips.
22	$\overline{\text{ERR}}$	Input/Output	Error—A synchronized error signal for the CVAX 78034 CPU and its associated support chips.
17	$\overline{\text{DMG}}$	Input	DMA grant—Provides bus direction control for the $\overline{\text{SYSRDY}}$, $\overline{\text{SYSERR}}$, $\overline{\text{RDY}}$, and $\overline{\text{ERR}}$ signals.
16	$\overline{\text{DS}}$	Input	Data strobe—A timing strobe to control the deassertion of the $\overline{\text{SYSRDY}}$, $\overline{\text{SYSERR}}$, $\overline{\text{RDY}}$, and $\overline{\text{ERR}}$ signals.
3,4,5,12, 13,19,33, 34,41,42	V_{DD}	Input	Voltage—5 volt power supply.
9,10,14, 23,26,27, 28,36,37, 40,43	V_{SS}	Input	Ground—Common ground reference.

Clock In (CLKIN)—This input provides the time base for all CCLOCK outputs. It is driven by a standard TTL oscillator output (nominally 40 MHz).

Master Clock A and B (MCLKA, MCLKB)—Precision MOS clock outputs that provide the timebase for the CVAX CPU and support chips. The frequency of these clocks is one-half of the oscillator input frequency (nominally 20 MHz). MCLKB is nominally delayed 180 degrees relative to MCLKA.

Auxiliary Clocks A, B, and C (ACLKA, ACLKB, ACLKC)—ACLKA and ACLKB are auxiliary precision clock outputs that provide a general purpose time base. ACLKA and ACLKB have the same frequency as MCLKA and are nominally delayed 0 degrees and 180 degrees, respectively, relative to MCLKA. The frequency of ACLKC is one-half of MCLKA and is nominally delayed 0 degrees relative to MCLKA. It can be used by the system interface and defines the starting time of a cycle.

Reset (RESET)—This output drives the $\overline{\text{RESET}}$ input of the CVAX CPU and support chips. It results in the chips being set to their initial powerup state. $\overline{\text{RESET}}$ is asynchronously asserted when the $\overline{\text{SYSRESET}}$ signal is asserted. It is deasserted after $\overline{\text{SYSRESET}}$ is deasserted and is synchronized with the rising edge of MCLKA. This defines the system phase 1 (P1).

System Reset ($\overline{\text{SYSRESET}}$)—This asynchronous input is synchronized by the CCLOCK to drive the $\overline{\text{RESET}}$ output. The assertion of $\overline{\text{SYSRESET}}$ causes the $\overline{\text{RESET}}$ output to be asynchronously asserted. The deassertion of SYSRESET is synchronized on the rising edge of MCLKA to cause $\overline{\text{RESET}}$ to deassert.

Ready ($\overline{\text{RDY}}$)—This is a bidirectional signal and is an open-drain output during a CPU cycle when $\overline{\text{DMG}}$ is deasserted. It is an input to $\overline{\text{SYSRDY}}$ during a DMA cycle when $\overline{\text{DMG}}$ is asserted. $\overline{\text{RDY}}$ connects directly to the $\overline{\text{RDY}}$ input of the CVAX CPU and synchronous CVAX support chips. $\overline{\text{RDY}}$ is an output when an asynchronous slave device may be responding to a CVAX CPU external reference ($\overline{\text{DMG}}$ deasserted).

When $\overline{\text{DS}}$ is deasserted, $\overline{\text{RDY}}$ is deasserted. When $\overline{\text{DS}}$ and $\overline{\text{SYSRDY}}$ are asserted, $\overline{\text{RDY}}$ is synchronized with MCLKA and ACLKC and then asserted. $\overline{\text{RDY}}$ is deasserted by the deassertion of $\overline{\text{DS}}$ or $\overline{\text{SYSRDY}}$, whichever comes first. An external pullup resistor is required to provide the deasserted state for the system. The $\overline{\text{RDY}}$ synchronizer is cleared on deassertion of $\overline{\text{SYSRDY}}$. It is a high impedance when $\overline{\text{SYSRESET}}$ is asserted.

System Ready ($\overline{\text{SYSRDY}}$)—This is a bidirectional signal. It is an open-drain output when $\overline{\text{DMG}}$ is asserted. It is an input to $\overline{\text{RDY}}$ when $\overline{\text{DMG}}$ is deasserted. $\overline{\text{SYSRDY}}$ connects to the $\overline{\text{RDY}}$ input of asynchronous CVAX support chips. $\overline{\text{SYSRDY}}$ is an output when the memory system is responding to a DMA transaction from an asynchronous device ($\overline{\text{DMG}}$ asserted).

When $\overline{\text{DS}}$ is deasserted, $\overline{\text{SYSRDY}}$ is deasserted and when $\overline{\text{DS}}$ and $\overline{\text{RDY}}$ are asserted, $\overline{\text{SYSRDY}}$ is asynchronously asserted. It is deasserted asynchronously with the deassertion of $\overline{\text{DS}}$ or $\overline{\text{RDY}}$, whichever comes first. An external pullup resistor is required to provide the deasserted state for the system. $\overline{\text{SYSRDY}}$ is a high impedance when $\overline{\text{SYSRESET}}$ is asserted.

Error ($\overline{\text{ERR}}$)—This is a bidirectional signal. It is an open-drain output when $\overline{\text{DMG}}$ is deasserted. It is an input to $\overline{\text{SYSERR}}$ when $\overline{\text{DMG}}$ is asserted. $\overline{\text{ERR}}$ connects to the $\overline{\text{ERR}}$ inputs of the CVAX CPU and CVAX synchronous support chips.

$\overline{\text{ERR}}$ is an output when an asynchronous slave device may be responding to a CVAX CPU external reference ($\overline{\text{DMG}}$ deasserted).

When $\overline{\text{DS}}$ is deasserted, $\overline{\text{ERR}}$ is a high impedance (deasserted). When DS and $\overline{\text{SYSERR}}$ are asserted, $\overline{\text{ERR}}$ is synchronized with MCLKA and ACLKC and asserted. $\overline{\text{ERR}}$ is deasserted by the deassertion of $\overline{\text{DS}}$ or $\overline{\text{SYSERR}}$, whichever comes first. An external pullup resistor is required to provide the

deasserted state for the system. The $\overline{ERR}$ synchronizer is cleared on deassertion of $\overline{SYSERR}$. $\overline{ERR}$ is a high impedance when $\overline{SYSRESET}$ is asserted.

System Error ($\overline{SYSERR}$)—This is a bidirectional signal. It is an open-drain output when $\overline{DMG}$ is asserted. It is an input to $\overline{ERR}$ when $\overline{DMG}$ is deasserted. $\overline{SYSERR}$ connects directly to the $\overline{ERR}$ input on asynchronous CVAX support chips.

$\overline{SYSERR}$ is an output when the memory system is responding to a DMA transaction from an asynchronous device ($\overline{DMG}$ asserted). When $\overline{DS}$ is deasserted, $\overline{SYSERR}$ is a high impedance (deasserted).

When $\overline{DS}$ and $\overline{ERR}$ are asserted, $\overline{SYSERR}$ asynchronously asserts. It deasserts asynchronously with the deassertion of $\overline{DS}$ or $\overline{ERR}$, whichever comes first. An external pullup resistor is required to provide the deasserted state for the system. $\overline{SYSERR}$ is a high impedance when $\overline{SYSRESET}$ is asserted.

Data Strobe ($\overline{DS}$)—This asynchronous input controls the state of $\overline{RDY}$, $\overline{ERR}$, $\overline{SYSRDY}$ and $\overline{SYSERR}$. These signals are deasserted when $\overline{DS}$ is deasserted.

When $\overline{DS}$ is asserted, the signals that are defined outputs by the state of $\overline{DMG}$ may be asserted depending on their respective inputs.

DMA Grant ($\overline{DMG}$)—This asynchronous input provides a bus direction control for driving $\overline{RDY}$, $\overline{ERR}$, $\overline{SYSRDY}$, and $\overline{SYSERR}$.

When $\overline{DMG}$ is deasserted, $\overline{RDY}$ and $\overline{ERR}$ are outputs that respond to the $\overline{SYSRDY}$, $\overline{SYSERR}$, and $\overline{DS}$ inputs. When $\overline{DMG}$ is asserted, $\overline{SYSRDY}$ and $\overline{SYSERR}$ are outputs that respond to the $\overline{RDY}$, $\overline{ERR}$, and $\overline{DS}$ inputs.

Test ($\overline{TEST}$)—This input provides an input enable control. When $\overline{TEST}$ is asserted, the CLKIN input is disabled and all CCLOCK outputs are timed by TCLKIN. This allows the system to be functionally tested at reduced frequencies without requiring the outputs to be set to high impedance. When this signal is not used in a CVAX system, it must be connected to V_{DD} through an external pullup resistor.

Test Clock Input (TCLKIN)—This auxiliary test input provides a time base for all CCLOCK outputs when the $\overline{TEST}$ input is asserted. It can be driven by TTL or MOS levels that are provided by the tester setup. It connects through an external resistor to the V_{DD} supply when not used. Because there are no dynamic nodes in the CCLOCK, the lower frequency limit of TCLKIN is not restricted.

Voltage (V_{DD})—These inputs provide 5 volts, (±5 percent) from the power supply.

Ground (V_{SS})—These inputs provide a ground reference.

Note

All V_{DD} inputs must be connected together and to the power supply using the shortest leads or power plane possible. All ground (V_{SS}) inputs must be connected together using the shortest wires or ground plane possible.

Interfacing Requirements

The CVAX 78135 clock generator interfaces to the CVAX system as shown in Figure 3. The system can be sectioned into a synchronous side and an asynchronous side. The synchronous side consists of the CVAX 78034 CPU, CVAX 78134 FPA, and other synchronous devices. The asynchronous side consists of the asynchronous system interface and associated support devices and the powerup logic. The $\overline{SYSRESET}$, $\overline{SYSRDY}$, and $\overline{SYSERR}$ signals are asynchronous and are controlled by the

MCLKA and MCLKB signals. The $\overline{\text{RESET}}$, $\overline{\text{RDY}}$, and $\overline{\text{ERR}}$ signals are the corresponding synchronous signals generated by the CCLOCK.

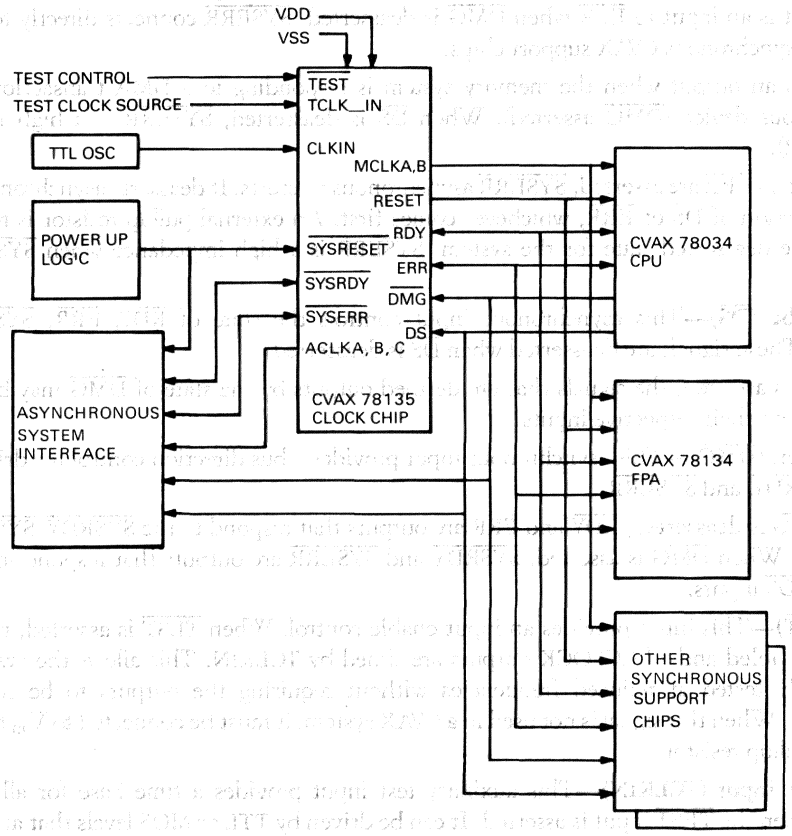


Figure 3 • CVAX 78135 Typical CVAX CPU System with CVAX 78135 Clock Generator

• Specifications

The mechanical, electrical and environmental specifications of the CCLOCK are contained in the following paragraphs. The test conditions for the values specified are listed as follows unless indicated otherwise.

- Temperature (T_A): 70°C
- Power supply voltage (V_{DD}): 4.75 V
- Ground (V_{SS}): 0 V

Mechanical Configuration

The physical dimensions of the CVAX 78135 44-pin surfacemount cerquad package are contained in the Appendix.

Absolute Maximum Ratings

Stresses greater than the absolute maximum ratings may cause permanent damage to the device. Exposure to the absolute maximum ratings for extended periods of time adversely affect the reliability of the device.

- Storage temperature range (T_S): −55°C to 125°C
- Active temperature range (T_A): 0°C to 70°
- Power supply voltage (V_{DD}): −0.5 V to 7.0 V
- Input or output voltage applied: 0.5 V to (V_{DD} + 0.5) V

Recommended Operating Conditions

- Temperature (T_A): 0°C to 70°C
- Power supply voltage (V_{DD}): 4.75 V to 5.25 V

dc Electrical Characteristics

The dc input and output parameters are listed in Table 2.

Table 2 • CVAX 78135 dc Input and Output Parameters

Symbol	Parameter	Requirements Min.	Max.	Units	Test Condition
V_{IH}	High-level input voltage	2.0	—	V	V_{DD} = 5.25 V
V_{IL}	Low-level input voltage	—	0.8	V	V_{DD} = 4.75 V
V_{OH}	High-level output voltage (TTL)	2.4	—	V	I_{OH} = −2.0 mA V_{DD} = 4.75 V
V_{OL}	Low-level output voltage (TTL)	0.4	—	V	I_{OL} = 40 mA V_{DD} = 4.75 V
V_{OHM}	High-level output voltage (MOS)	90% V_{DD}	—	V	$_{OH}$ = −100 µA
V_{OLM}	Low-level output voltage (MOS)	—	10% V_{DD}	V	I_{OL} = 100 µA
I_{IL}	Input leakage current	−10	10	µA	0 < V_{in} < 5.25 V
I_{OL}	Output leakage current	−50	50	µA	0 < V_{in} < 5.25 V
I_{CC}	Active supply current	—	200	mA	I_{out} = 0 T_A = 70°C $\overline{DS}$ deasserted $t_{CYCLEIN}$ = 25 ns* V_{DD} = 5.25 V
C_{in}	Input capacitance	—	10	pF	

Symbol	Parameter	Requirements Min.	Max.	Units	Test Condition
C_{outc}	Output capacitance MCLKA, MCLKB, ACLKA, ACLKB, and ACLKC	—	25	pF	
C_{res}	$\overline{\text{RESET}}$ output capacitance	—	20	pF	
C_{outio}	I/O pin output capacitance	—	15	pF	

*Refer to Table 3.

ac Electrical Characteristics

The following notes apply to Figures 4 through 9 and their associated timing tables.

- All times are in nanoseconds except as noted.
- $C_{load} = 150$ pF
- ac highs for MOS outputs are measured at V_{OHM} and lows are measured at V_{OLM}.
- ac highs for TTL inputs are measured at V_{IH} and lows are measured at V_{IL}.
- ac highs for TTL outputs are measured at V_{OH} and lows are measured at V_{OL}.
- TTL inputs are driven to V_{OL} or V_{OH}.

Clock Inputs and Output Timing

Figure 4 shows the timing for the clock input and Table 3 lists the clock input timing parameters. Figure 5 shows the clock output timing and Table 4 lists the clock output timing parameters.

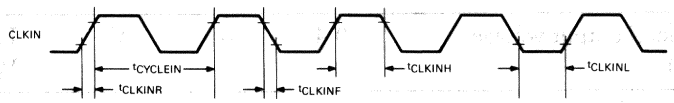


Figure 4 • CVAX 78135 Clock Input Timing

Table 3 • CVAX 78135 Clock Input Timing Parameters*

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{CLKINF}	CLKIN fall	0.5	4.5
t_{CLKINH}	CLKIN high	9.5	—
t_{CLKINL}	CLKIN low	9.5	—
t_{CLKINR}	CLKIN rise	0.5	4.5
$t_{CYCLEIN}$	CLKIN cycle	25	—

*Requirements for TCLKIN are the same as for CLKIN.

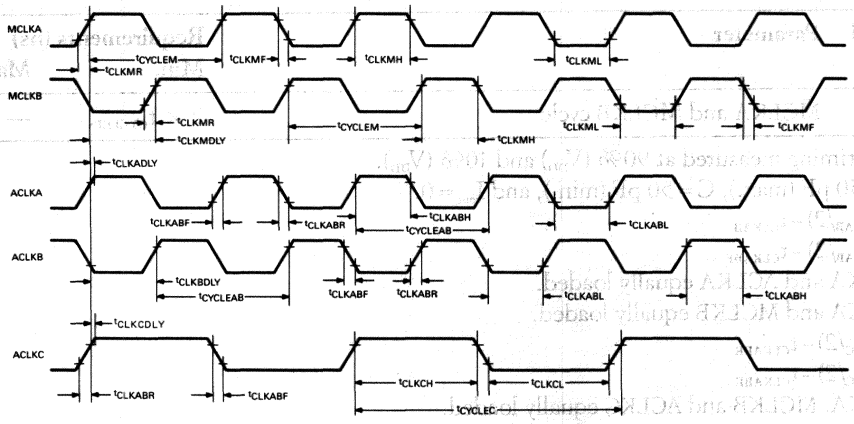


Figure 5 • CVAX 78135 Clock Output Timing

Table 4 • CVAX 78135 Clock Output Timing Parameters

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{CLKABF}	ACLKA, ACLKB and ACLKC fall[1,2]	0.8	5.0
t_{CLKABH}	ACLKA and ACLKB high[1]	[3]	—
t_{CLKABL}	ACLKA and ACLKB low[1]	[4]	—
t_{CLKABR}	ACLKA, ACLKB and ACLKC rise[1,2]	0.8	5.0
$t_{CLKADLY}$	MCLKA to ACLKA delay[5]	–0.5	0.5
$t_{CLKBDLY}$	MCLKA to ACLKB delay[6]	$t_{CYCLEAB}/2$	$t_{CYCLEAB}/2$
t_{CLKCH}	ACLKC high[1]	[7]	—
t_{CLKCL}	ACLKC low[1]	[8]	—
$t_{CLKCDLY}$	MCLKA or MCLKB to ACLKC delay to coincident edge[9]	–0.5	0.5
$t_{CLKMDLY}$	MCLKA to MCLKB phase delay[10]	[11]	[12]
t_{CLKMF}	MCLKA and MCLKB fall[1,13]	0.8	5.0
t_{CLKMH}	MCLKA and MCLKB high[1]	[14]	—
t_{CLKML}	MCLKA and MCLKB low[1]	[15]	—
t_{CLKMR}	MCLKA and MCLKB rise[1,13]	0.8	5.0
$t_{CYCLEAB}$	ACLKA and ACLKB cycle	$2 \times t_{CYCLEIN}$	—
t_{CYCLEC}	ACLKC cycle	$4 \times t_{CYCLEIN}$	—

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{CYCLEM}	MCLKA and MCLKB cycle	$2 \times t_{CYCLEIN}$	—

[1]MOS timing measured at 90% (V_{DD}) and 10% (V_{DD}).
[2]C = 150 pF (max.), C = 50 pF (min.), and $I_{out} = 0$.
[3]$(t_{CYCLEAB}/2) - t_{CLKAB}$
[4]$(t_{CYCLEAB}/2) - t_{CLKABF}$
[5]MCLKA and ACLKA equally loaded.
[6]MCLKA and MCLKB equally loaded.
[7]$(t_{CYCLEC}/2) - t_{CLKABR}$
[8]$(t_{CYCLEC}/2) - t_{CLKABF}$
[9]MCLKA, MCLKB and ACLKC equally loaded.
[10]MCLKA and ACLKB equally loaded, $I_{out} = 0$
[11]$(t_{CYCLEM}/2) - 0.5$
[12]$(t_{CYCLEM}/2) + 0.5$
[13]C = 150 pF (max.), C = 50 pF (min.).
[14]$(t_{CYCLEM}/2) - t_{CLKMR}$ (max.)
[15]$(t_{CYCLEM}/2) - t_{CLKMF}$ (max.)

Clock Initialization

Figure 6 shows the clock initialization timing, and the parameters are listed in Table 5.

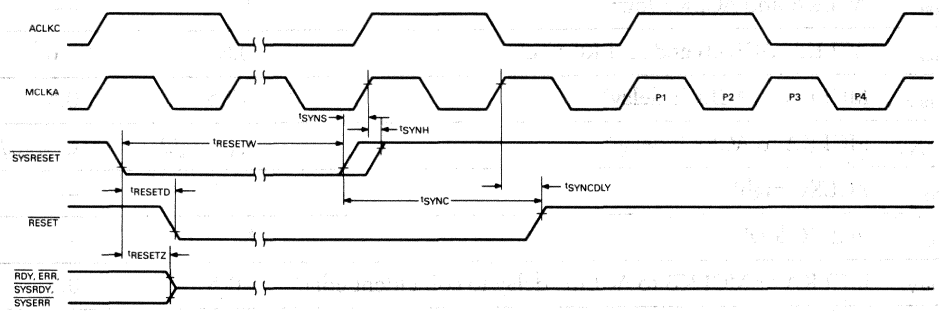


Figure 6 • CVAX 78135 Initialization Timing

Table 5 • CVAX 78135 Initialization Timing Parameters

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{RESETD}	Async assertion time for $\overline{RESET}$	$t_{SYNS} + t_{SYNHC}$	[1]
t_{RESETW}	$\overline{SYSRESET}$ width	[1]	—
t_{RESETZ}	High-impedance output from $\overline{SYSRESET}$	20	—

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{SYNC}	$\overline{RDY}$ and $\overline{ERR}$ assertion from $\overline{SYSRDY}$ and $\overline{SYSERR}$ assertion with $\overline{DMG}$ deasserted	[2]	[1]
$t_{SYNCDLY}$	$_{RDY}$ and $\overline{ERR}$ assertion and $\overline{RESET}$ deassertion from MCLKA (phase 3) edge with $\overline{DMG}$ deasserted	5.0	15
t_{SYNH}	Synchronizer	5.0	—
t_{SYNS}	Synchronizer setup	5.0	—

[1] $15 + (3 \times t_{CYCLEM}) - t_{SYNH}$
[2] $t_{SYNS} + 5 + t_{CYCLEM}$

Ready and Error Timing

Figure 7 shows the timing for the $\overline{RDY}$ and $\overline{ERR}$ signals with $\overline{DMG}$ deasserted, and Table 6 lists the timing parameters.

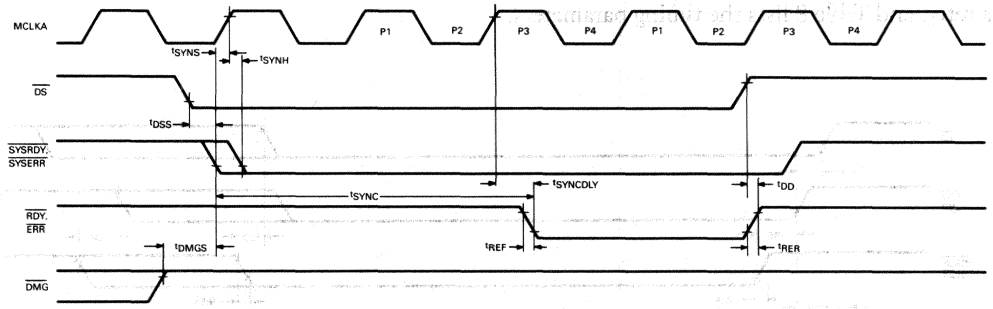


Figure 7 • CVAX 78135 Ready and Error ($\overline{DMG}$ Deasserted) Signal Timing

Table 6 • CVAX 78135 Ready and Error ($\overline{DMG}$ Deasserted) Signal Timing Parameters

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{DD}	$\overline{RDY}$, $\overline{SYSRDY}$, $\overline{ERR}$, and $\overline{SYSERR}$ deassertion from $\overline{DS}$ deassertion	2.0	10
t_{DMGS}	$\overline{DMG}$ setup time before $\overline{DS}$	20	—
t_{DSS}	$\overline{DS}$ setup time before $\overline{RDY}$ or $\overline{ERR}$	10	—
t_{REF}	$\overline{RDY}$/$\overline{ERR}$ fall	1.0	10
t_{RER}	$\overline{RDY}$/$\overline{ERR}$ rise	1.0	40

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{SYNC}	$\overline{RDY}$, $\overline{ERR}$ assertion from $\overline{SYSRDY}$ and $\overline{SYSERR}$ assertion	[1]	[2]
$t_{SYNCDLY}$	$\overline{RDY}$, and $\overline{ERR}$ assertion and $\overline{RESET}$ deassertion from MCLKA (phase 3) edge	5.0	15
t_{SYNH}[3]	Synchronizer hold	5.0	—
t_{SYNS}[3]	Synchronizer setup	5.0	—

[1] $t_{SYNS} + 5 + t_{CYCLEM}$
[2] $15 + (3 \times t_{CYCLEM}) - t_{SYNH}$
[3] t_{SYNS} and t_{SYNH} are the setup and hold times needed at a synchronizer input to ensure that the signal is recognized as expected. Signals with a shorter setup or hold time than specified will be synchronized but the results are unpredictable.

System Ready and System Error Timing

Figure 8 shows the timing for the $\overline{SYSRDY}$ and $\overline{SYSERR}$ signals with $\overline{DMG}$ asserted, and Table 7 lists the timing parameters. Figure 9 shows the timing for the $\overline{SYSRDY}$ and $\overline{SYSERR}$ signals during a retry, and Table 8 lists the timing parameters.

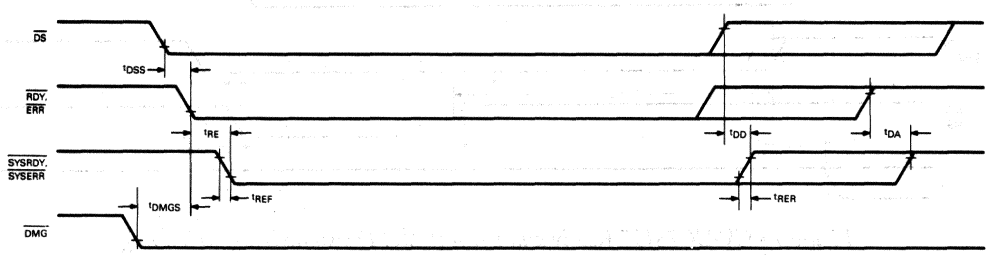


Figure 8 • CVAX 78135 System Ready and System Error ($\overline{DMG}$ Asserted) Signal Timing

Table 7 • CVAX 78135 System Ready and System Error ($\overline{DMG}$ Asserted) Signal Timing Parameters

Symbol	Parameter*	Requirements (ns) Min.	Max.
t_{DA}	$\overline{RDY}$, $\overline{SYSRDY}$, $\overline{ERR}$, and $\overline{SYSERR}$ high impedance from deassertion of asynchronous input prior to $\overline{DS}$ deassertion	4.0	15
t_{DD}	$\overline{RDY}$, $\overline{SYSRDY}$, $\overline{ERR}$, $\overline{SYSERR}$ deassertion from $\overline{DS}$ deassertion	2.0	10
t_{DMGS}	$\overline{DMG}$ setup time before $\overline{DS}$	20	—

Symbol	Parameter*	Requirements (ns) Min.	Max.
t_{DSS}	$\overline{DS}$ setup time before $\overline{RDY}$ or $\overline{ERR}$	10	—
t_{RE}	$\overline{SYSRDY}$ and $\overline{SYSERR}$ assertion from $\overline{RDY}$ and $\overline{ERR}$ assertion	5.0	15
t_{REF}	$\overline{RDY}$ or $\overline{ERR}$ fall	1.0	10
t_{RER}	$\overline{RDY}$ or $\overline{ERR}$ rise	1.0	40

*$\overline{SYSRDY}$ and $\overline{SYSERR}$ are not synchronized to MCLKA or MCLKB in this configuration since the asynchronous portion of the system is receiving these signals.

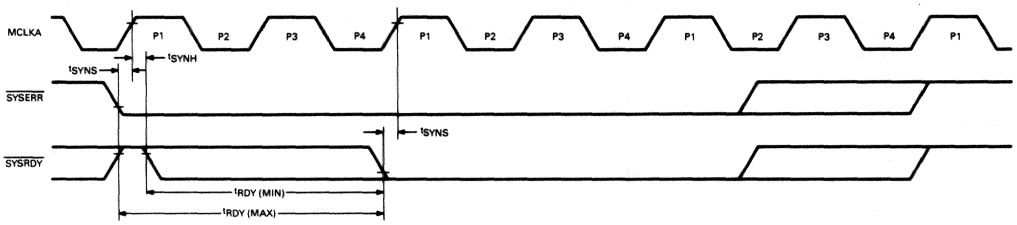


Figure 9 • CVAX 78135 System Ready and System Error (Retry) Signal Timing

Table 8 • CVAX 78135 System Ready and System Error (Retry) Signal Timing Parameters

Symbol	Parameter*	Requirements (ns) Min.	Max.
t_{SYNH}	Synchronizer hold	5.0	—
t_{SYNS}	Synchronizer setup	5.0	—
t_{RDY}	$\overline{SYSERR}$ to $\overline{SYSRDY}$ on retry	$t_{SYNS} + t_{SYNH}$	$2 \times t_{CYCLEM}$

Features

- Support for external ROM
  —8-, 16-, or 32-bit data width
  —Programmable address decoding
  —Cycle completion (RDY assertion)
  —Byte or word unpacking
- 1 KB of battery backed-up RAM (256 by 32 bit)
- VAX SRM compatible console terminal UART similar to DLART with eight baud rates, CTRL/P break detection, and secure console support
- VAX SRM compatible console storage (auxiliary) UART similar to DLART
- 100-Hz interval timer
- Two programmable timers
- 4-bit output port
- Programmable bus timeout
- Battery backed-up VAX SRM time-of-year clock
- I/O bus reset support
- Halt arbitration logic
- Two programmable address decode strobes

Description

The MicroVAX 78332 System Support Chip (SSC) is a multifunction interface that provides the common functions necessary to support the MicroVAX or CVAX system environment. It includes unpacking logic for up to 1 MB of external ROM, a 1-KB RAM, two asynchronous serial line ports, a 4-bit output port, two programmable address decoders, two programmable timers, and a realtime clock. It substantially reduces the number of components necessary to develop a MicroVAX system on a CPU board and is contained in a single 84-pin cerquad package. The functional block diagram of the MicroVAX 78332 SSC is shown in Figure 1.

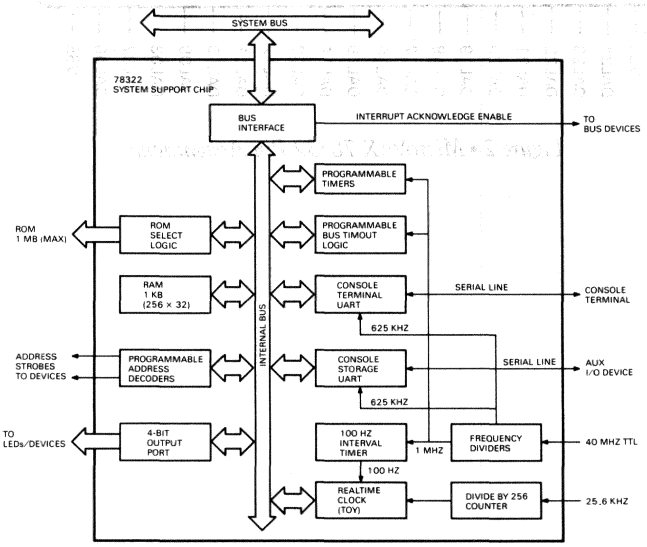


Figure 1• MicroVAX 78332 SSC Functional Block Diagram

Pin and Signal Descriptions

The input and output pins and power and ground connections of the MicroVAX 78332 SSC are shown in Figure 2. Table 1 provides a summary of the signals defined in the following paragraphs.

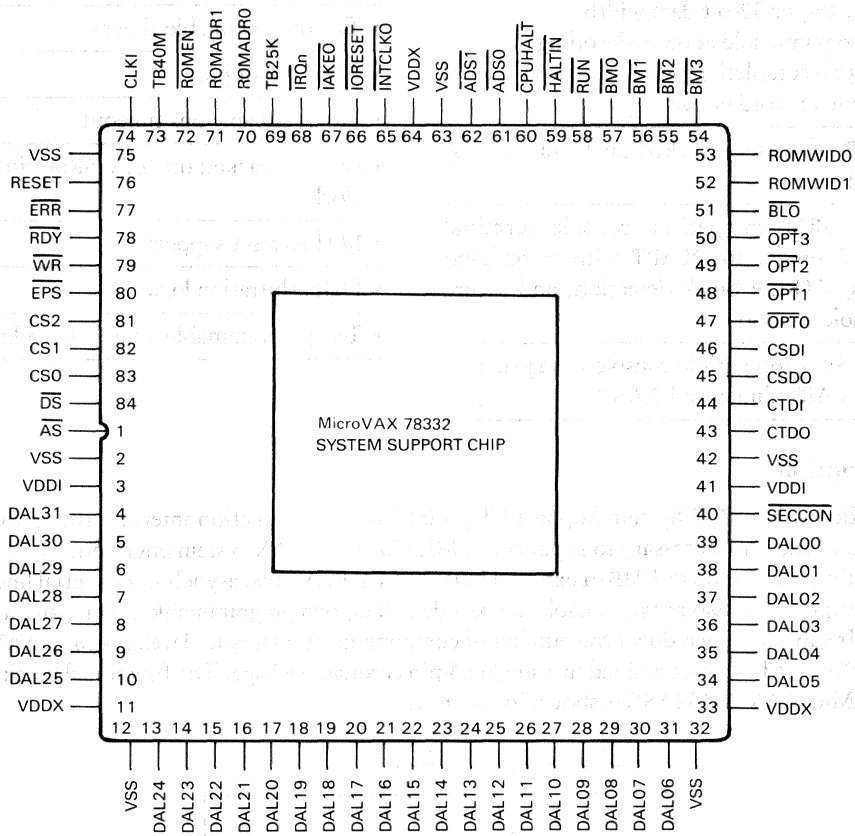


Figure 2 • MicroVAX 78332 Pin Assignments

Table 1 • MicroVAX 78332 Pin and Signal Summary

Pin	Signal	Type	Description/Function
4-10,13-31, 34-39	DAL<31:00>	input/output	Data and Address Lines—Time multiplexed lines used to transfer address and data information between the SSC, the CPU, and the SSC ROM.
81-83	CS<2:0>	input/output	Cycle status—Provides status and control information about the current bus cycle. Connects to CS/DP<2:0> on the CVAX.
1	$\overline{AS}$	input	Address strobe—Provides timing and control information to the SSC.
84	$\overline{DS}$	input	Data strobe—Provides timing and control information for data transfers to and from the SSC.
54-57	$\overline{BM<3:0>}$	input	Byte mask—Indicates which bytes of DAL lines data contain valid information during the second part of an I/O cycle.
79	$\overline{WR}$	input	Write—Specifies the direction of data transfer on the DAL.
78	$\overline{RDY}$	output	Ready—Asserted by the SSC to indicate the end of bus cycle.
77	$\overline{ERR}$	output	Error—Asserted by the SSC to indicate a bus timeout condition.
80	$\overline{EPS}$	input	External processor strobe—Coordinates the MicroVAX external processor transactions.
76	RESET	input	Reset—Asserted during power system transitions and during battery backup mode. The deassertion of the RESET signal initializes the SSC.
60	$\overline{CPUHALT}$	output	CPU Halt—A halt request to the CPU.
59	$\overline{HALTIN}$	input	Halt in—A halt request from the external logic.
40	$\overline{SECCON}$	input	Secure console—When asserted, the halt requests received by the console terminal are not transmitted to the CPU.
58	$\overline{RUN}$	output	Run—Asserted when the halt signals are enabled.
68	$\overline{IRQ<n>}$	output	Interrupt request—An interrupt request to a CPU.

Pin	Signal	Type	Description/Function
67	$\overline{\text{IAKEO}}$	output	Interrupt acknowledge enable out—Asserted by the SSC upon receipt of an interrupt acknowledge from the CPU if no SSC interrupts are pending at the interrupt acknowledge level.
66	$\overline{\text{IORESET}}$	output	I/O System Reset—Asserted by the SSC when the CPU writes to processor register IPR #55. Typically used to reset a Q-bus system.
72	$\overline{\text{ROMEN}}$	output	ROM Enable—Enables the external ROM.
70,71	ROMADR<1:0>	output	ROM Address—Selects the correct bytes during a ROM read operation.
53,52	ROMWID<1:0>	input	ROM Width—Determines the width of the ROM and is also used to cause the SSC input/output lines to become high impedance.
44	CTDI	input	Console Terminal Data In—A serial input to the console terminal receiver.
43	CTDO	output	Console Terminal Data Out—A serial output of the console terminal transmitter.
46	CSDI	input	Console Storage Data In—A serial input to the console storage receiver.
45	CSDO	output	Console Storage Data Out—A serial output of the console storage transmitter.
62,61	$\overline{\text{ADS<1:0>}}$	output	Decoder Strobes—Asserted when the selected addresses are detected by the SSC.
50,47	$\overline{\text{OPT<3:0>}}$	output	Output Port—Can be used to control output devices such as LED indicators.
65	$\overline{\text{INTCLKO}}$	output	Interval Timer Clock Output—A 100-Hz interval timer signal.
74	CLKI	input	Clock In—An SSC clock timing signal nominally 40 MHz.
73	TB40M	input	Time Base 40 MHz—Provides a 40-MHz timebase to the baud rate generator, the bus timeout logic, and the interval and programmable timers. Also provides a timebase to the time-of-year clock if the TB25K input is grounded.
69	TB25K	input	Time Base 25.6 KHz—Provides the timebase for the time-of-year clock.
51	$\overline{\text{BLO}}$	input	Battery Low—Indicates that the battery power is low.

Pin	Signal	Type	Description/Function
3,41	V_{DDI}	input	Voltage—Continuous 5 Vdc power to the SSC internal circuits.
11,33,64	V_{DDX}	input	Voltage—5 Vdc power to the SSC pads.
2,12,32,63	V_{SS}	input	Ground-Ground reference.

Data and Address Lines

Data and Address Lines (DAL<31:00>)—The data and address lines are time-multiplexed and transmit addresses and data between the CPU and other devices. The protocols used for the MicroVAX CPU and the CVAX CPU are as follows.

During the first part of a MicroVAX CPU read or write cycle, DAL<29:02> contain the longword address of the operand. During the second part of a CPU read cycle DAL<31:00> are used to transmit information to the CPU. During the second part of a MicroVAX CPU write cycle, DAL<31:00> are used to receive incoming information.

During the first part of an MicroVAX interrupt acknowledge cycle, DAL<04:00> contain the interrupt priority level of the interrupt being acknowledged, DAL<31:30> = 10 and DAL<29:05> are zeros. During the second part of the cycle, the interrupt vector is transmitted to the CPU on DAL<09:02>.

During a MicroVAX external processor (EP) write command cycle, the CPU transfers the processor register number on DAL<05:00> and the upcoming transaction type on DAL31 (read = 1, write = 0).

During a MicroVAX EP read response cycle, DAL<31:00> transmit information to the CPU. During an EP write data cycle, DAL<31:00> receives incoming information from the CPU.

During the first part of a CVAX CPU read or write cycle, DAL<29:02> contain the longword address of the operand. During the second part of a read cycle, DAL<31:00> transmit information to the CPU.

During the second part of a write cycle, DAL<31:00> receive incoming information from the CPU.

During the first part of a CVAX CPU interrupt acknowledge cycle, DAL<06:02> contain the interrupt priority level of the interrupt being acknowledged and DAL<31:07> and DAL<01:00> equal zero. During the second part of the cycle, DAL<31:00> transmit information to the CPU.

During the first part of a CVAX EP read or write cycle, the CPU transfers the processor register number on DAL<07:02> and zeros on DAL<10:08>. During the second part of an EP read cycle, DAL<31:00> transmit information to the CPU. During the second part of an EP write cycle, DAL<31:00> receive incoming information from the CPU.

Control Lines

Address Strobe ($\overline{AS}$)—During a CPU read or write cycle, a CVAX External Processor register read or write cycle, or an interrupt acknowledge cycle, the CPU asserts the $\overline{AS}$ line when the information on DAL<31:00> is valid and deasserts the line when the bus cycle has been completed.

Data Strobe ($\overline{DS}$)—During a CPU read cycle or interrupt acknowledge cycle, the CPU asserts the $\overline{DS}$ signal to indicate that DAL<31:00> are available to receive incoming data and deasserts the

$\overline{DS}$ signal to indicate that it has received and latched the incoming data. During a CPU write cycle or CVAX external processor register write cycle, it is asserted by the CPU to indicate that DAL<31:00> contain valid data and deasserted when the data is not valid.

During a CPU write cycle, the CPU asserts the $\overline{DS}$ line to indicate that DAL<31:00> contain valid outgoing data and deasserts the $\overline{DS}$ line to indicate that the data will be removed from the bus.

Byte Mask ($\overline{BM}$<3:0>)—The byte mask specifies which bytes of the DAL contain valid information during the second part of a CPU write cycle. The SSC ignores the byte mask except during write operations to the RAM. The byte mask assignments are shown in Table 2.

Table 2 • MicroVAX 78332 Byte Mask Data Selection

Byte Mask Line	Valid Lines
$\overline{BM3}$	DAL<31:24>
$\overline{BM2}$	DAL<23:16>
$\overline{BM1}$	DAL<15:08>
$\overline{BM0}$	DAL<07:00>

Write ($\overline{WR}$)—This input specifies the direction of data transfer on the the DAL. When $\overline{WR}$ is asserted, the CPU transfers data on the DAL. When $\overline{WR}$ is not asserted, the CPU receives data from the DAL. The $\overline{WR}$ signal is latched when the $\overline{AS}$ input is asserted.

Ready ($\overline{RDY}$)—The SSC asserts this output to indicate that the current bus cycle should be successfully terminated. During a read cycle or interrupt acknowledge cycle, the assertion of the $\overline{RDY}$ signal indicates that the SSC has placed the required data on the DAL. During a write cycle, the assertion indicates that the SSC has latched the data. It remains asserted until the $\overline{AS}$ input is deasserted. This is an open-drain output.

Error ($\overline{ERR}$)—The SSC asserts this output to indicate that a timeout of the current bus cycle has occurred. The length of the timeout period is determined by the value loaded into the Bus Timeout Control register of the SSC. The $\overline{ERR}$ output remains asserted until the $\overline{DS}$ input is deasserted. This is an open-drain output.

External Processor Strobe ($\overline{EPS}$)—This signal is used by the MicroVAX CPU to coordinate external processor transactions. It is not used by the CVAX CPU.

Reset (RESET)—The deassertion of the RESET input initializes the SSC to its powerup state. It must be asserted during battery backup mode (i.e., during a power loss) or when there is a transition on a power supply output. When asserted, the DAL<31:00>, $\overline{RDY}$, $\overline{ERR}$, and the CS2 lines are forced to a high-impedance state. All other outputs are deasserted.

Cycle Status (CS<2:0>)—These lines and the $\overline{WR}$ line provide status and control information for the current bus cycle. The CS<2:0> line information is latched when the $\overline{AS}$ line is asserted.

In a CVAX CPU system, the CS<2:0> lines connect to and are time-multiplexed with the Cycle Status/Data Parity lines (CS/DP<2:0>). The SSC ignores the DAL line parity and latches the CS<2:0> lines information at the assertion of the $\overline{AS}$ input.

Table 3 lists the bus cycle selected during a CPU read or write cycle, an interrupt acknowledge cycle, or a CVAX EP read or write cycle.

Table 3 • MicroVAX 78332 Bus Cycle Control Selection (non-MicroVAX EP cycle)*

Write	Control Line			Bus Cycle Type
$\overline{WR}$	CS2	CS1	CS0	
H	L	L	L	READ
H	L	H	L	CVAX EP register read
H	L	H	H	interrupt acknowledge
H	H	L	L	I-stream read
H	H	L	H	read
H	H	H	L	read
H	H	H	H	read
L	L	H	L	CVAX EP register write
L	H	L	H	write
L	H	H	H	write

*$\overline{AS}$ = L and $\overline{EPS}$ = H. The CS<2:0> combinations not listed are reserved.

During a MicroVAX EP transaction involving the SSC, the CS2 line is not used except during an EP Read Response cycle where CS2 is pulled low by the SSC. Table 4 shows the bus cycle selected during a MicroVAX EP read or write cycle.

Table 4 • MicroVAX 78332 Bus Cycle Control Selection (MicroVAX EP cycle)*

Write	Control Line[1]			Bus Cycle Type
$\overline{WR}$	CS2	CS1	CS0	
H	X[3]	H	H	read response
L	X	L	H	write data
L	X	H	L	write command[2]

*$\overline{AS}$ = H and $\overline{EPS}$ = L. The CS<2:0> combinations not listed are reserved.

[1]H = high level, L = low level, X = high or low level

[2]During an EP write command cycle, DAL31 indicates that the transaction that follows is a read (H) or a write (L).

[3]Precharged high (H) and asserted low during read response.

CPU Halt ($\overline{CPUHALT}$)—This signal is asserted for at least eight microcycles (nominally 800 nanoseconds) when a halt request is detected by the SSC. It connects to the HALT input of the CPU.

Halt In ($\overline{HALTIN}$)—This is a level-sensitive input that receives halt requests from external logic. When appropriate, these requests are passed to the CPU through the $\overline{CPUHALT}$ output.

Secure Console (SECCON)—When this input is connected to ground, breaks received by the Console Terminal UART are prevented from asserting $\overline{CPUHALT}$.

Run ($\overline{\text{RUN}}$)—This output is asserted when the halt conditions are enabled and is deasserted when the halt conditions are disabled. During the first microsecond after the RESET input is deasserted, the $\overline{\text{RUN}}$ signal will oscillate at approximately 10 MHz (for test purposes) and then operate normally.

Interrupt Request ($\overline{\text{IRQ}}$<n>)—This open-drain output requests an interrupt from the CPU on one of the four CPU IRQ lines. The interrupt priority level is defined by bits <25:24> of the SSC Configuration register and must correspond to the IRQ level to which the $\overline{\text{IRQ}}$<n> output is connected.

Interrupt Acknowledge Enable Out ($\overline{\text{IAKEO}}$)—The SSC asserts this output when it receives interrupt acknowledge cycle that it has not requested. This output is not asserted if an SSC interrupt is pending and the IPL is the same as defined by bits <25:24> of the SSC Configuration register. The SSC deasserts the $\overline{\text{IAKEO}}$ output when the CPU deasserts the $\overline{\text{DS}}$ signal.

I/O Reset ($\overline{\text{IORESET}}$)—The SSC asserts this output when a write cycle to Internal Processor Register 55 is a request for a bus reset.

ROM Select

ROM Enable ($\overline{\text{ROMEN}}$)—The assertion of this output by the SSC enables the ROM bank to be read. It connects to the Chip Enable inputs of the ROM(s).

ROM Address (ROMADR<1:0>)—When using an external wordwide ROM, the ROMADR1 output connects to the A0 input on the ROM. When using an external byte-wide ROM, the ROMADR1 and ROMADR0 outputs connect to A1 and A0 inputs, respectively, on the ROM.

ROM Width (ROMWID<1:0>)—These outputs select the width of the boot ROM and the DAL that connect to the ROM. It also selects a high-impedance state for all SSC I/O outputs. The selections are listed in Table 5.

Table 5 • MicroVAX 78332 Boot ROM Width Selection

ROMWID 1	ROMWID 0	Width	Lines
H	H	32	DAL<31:00>
H	L	16	DAL<15:00>
L	H	8	DAL<07:00>
L	L		high impedance (all SSC I/O and output lines)

The ROMWID outputs that are to remain a high level connect to V_{DD} through resistors. When both pins are connected to ground, all SSC I/O and output lines are high impedance.

Serial Data

Console Terminal Data In (CTDI)—This input provides serial character data to the console terminal receiver of the SSC.

Console Terminal Data Out (CTDO)—This output provides serial character data from the console terminal transmitter of the SSC.

Console Storage UART Data In (CSDI)—This input provides serial character data to the Console Storage UART receiver of the SSC.

Console Storage UART Data Out (CSDO)—This output provides serial character data from the Console Storage UART transmitter of the SSC.

Miscellaneous Signals

Address Decoder Strobes (ADS<1:0>)—These outputs provide strobe signals to external logic when predefined addresses are detected by the SSC.

Output Port (OPT<3:0>)—These outputs from the Output Port register can be used to control LED indicators or other devices.

Clock Signals

Clock Output ($\overline{\text{INTCLKO}}$)—This output is produced by the 100-Hz Interval Timer of the SSC. It normally connects to the INTTIM input on the CPU in order to generate Interval Clock interrupts. The first assertion of this signal is approximately 8.2 milliseconds after the deassertion of the RESET signal.

Clock In (CLKI)—This high frequency TTL input (nominally 40 MHz) provides the basic clock timing to the SSC. In the transition to normal operation, the RESET input should not be deasserted until CLKI is within specification.

Timebase 40 MHz (TB40M)—This input is driven by an external 40-MHz TTL oscillator and provides the timebase for the baud-rate generators in the UARTs, the Programmable Bus Timeout logic, the 100-Hz Interval Timer, and the programmable timers. If the TB25K input is connected to ground, this oscillator will also supply the timebase for the time-of-year clock when the system power is supplied.

Timebase 25.6 KHz (TB25K)—When driven by an external 25.6-KHz oscillator, this input supplies the timebase for the Time-of-Year (TOY) clock. To maintain the TOY clock when system power is removed, this oscillator should be supplied power from battery backup unit. When this input is connected to ground, the TOY clock uses the TB40M signal as its timebase while system power is supplied. This input requires a CMOS level and must not be switched between the oscillator and ground while the SSC is running.

Power and Ground

Battery Low ($\overline{\text{BLO}}$)—If this input is asserted while the RESET input is asserted, the BLO bit 31 of the SSC Configuration Register is set. It can be cleared only by the user. If the BLO bit is set when the SSC is reset, the time-of-year clock is cleared.

Voltage (V_{DDI})—These inputs provide continuous dc power to the internal circuits of the SSC. When the RESET input is asserted, a lower voltage and current is supplied so that the RAM will hold its state and the time-of-year clock will continue to operate. The RESET input must be asserted when the voltage on these inputs is transitioning.

Voltage (V_{DDX})—These inputs provide dc power to the pad drivers of the SSC. The RESET input must be asserted when the voltage on these inputs is transitioned.

Internal Ground (V_{SS})—These input pins provide the ground reference to the SSC.

Registers

The SSC contains ten VAX Internal Processor Registers (IPR) that may be addressed either by their IPR number (through MTPR or MFPR instructions) or by their I/O space address. All SSC register accesses are 32-bits wide and longword aligned. The SSC registers are contained in a relocatable 2-KB block of I/O space except for the Base Address register (BA) which has a fixed I/O space address

of 20140000. Storage element locations are defined as offsets from the value contained in the BA register. The notation "BA + <offset>" denotes the address of the storage elements.

The SSC registers are listed in Table 6. The offsets are shown in hexadecimal notation and the IPR numbers in decimal notation.

Table 6 • MicroVAX 78332 Internal Register Offset and Number

Offset	IPR Register	
000-003	—	Base Address (BA)
004-00F	—	Reserved
010-013	—	SSC Configuration
014-01F	—	Reserved
020-023	—	Bus Timeout Control
024-02F	—	Reserved
030-033	—	Output Port
034-06B	—	Reserved
06C-06F	27	Time-of-Year (TOY)
070-073	28	Console Storage Receiver Status (CSRS)
074-077	29	Console Storage Receiver Data (CSRD)
078-07B	30	Console Storage Transmitter Status (CSTS)
07C-07F	31	Console Storage Transmitter Data (CSDB)
080-083	32	Console Receiver Control/Status (RXCS)
084-087	33	Console Receiver Data Buffer (RXDB)
088-08B	34	Console Transmitter Control/Status (TXCS)
08C-08F	35	Console Transmitter Data Buffer (TXDB)
090-0DB	—	Reserved
0DC-0DF	55	I/O System Reset (IORESET)
0E0-0EF		Reserved
0F0-0F3	—	Rom Data*
0F4-0F7	—	Bus Timeout Counter*
0F8-0FB	—	Interval Timer*
0FC-0FF	—	Reserved
100-103	—	Timer 0 Control
104-107	—	Timer 0 Interval
108-10B	—	Timer 0 Next Interval
10C-10F	—	Timer 0 Interrupt Vector
110-113	—	Timer 1 Control
114-117	—	Timer 1 Interval
118-11B	—	Timer 1 Next Interval
11C-11F	—	Timer 1 Interrupt Vector
120-1FF	—	Reserved

Offset	IPR Register	
130-133	—	Address Decode Channel 0 Match
134-137	—	Address Decode Channel 0 Mask
138-13F	—	Reserved
140-143	—	Address Decode Channel 1 Match
144-147	—	Address Decode Channel 1 Mask
148-3FF	—	Reserved
400-7FF	—	Internal RAM

*These registers are used for test purposes and should not be accessed by the user.

The hardware updates the register bits in response to events within the SSC. The register information that is accessible by the user is defined as follows.

RW Read/Write—Can be read or written by the user. The hardware can change the value of the bit only when the RESET input is asserted.

RW' Read/Write'—Can be cleared by the hardware at any time. Writing to these bit by the user is ignored if cleared by the hardware during the same cycle.

RO Read Only—Can be read only by the user. Only the hardware can change the value of the bit. Writing to these bits by the user is ignored.

WO Write Only—Can be written only by the user and is read as a zero.

WC Write 1 to Clear—Can be read by the user. The hardware can change the value of these bits. If not being updated by the hardware, the user can clear these bits by writing a 1 to them.

MBZ Must Be Zero—Always read as zero. Writing to these bits is ignored.

Base Address Register

The Base Address (BA) register contains the base address of the relocatable 2-KB block of I/O space in which the SSC internal RAM and 29 control/status registers are located. The software writes the base address of this block. The SSC RAM and registers are then addressed by adding the offset value to the value in the BA register. When the RESET input is asserted, this register is set to the default value of 20140000. With this value, the BA register is located within the 2-KB block of relocatable I/O space assigned to the SSC. The register format is shown in Figure 3 and the function of the bits is described in Table 7.

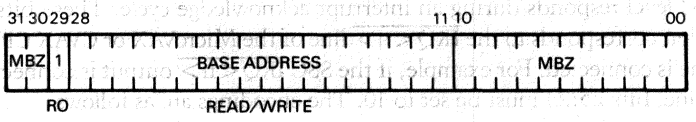


Figure 3 • MicroVAX 78332 Base Address Register Format

Table 7 • MicroVAX 78332 Base Address Register Description

Bit	Function
31:30	MBZ (Must be zero)
29	Set to a one
28:11	Base Address—The base address of the relocatable 2-KB block of I/O space.
10:00	MBZ (Must be zero)

Configuration Register

This register contains the setup information that defines the functions of SSC. The register format is shown in Figure 4 and Table 8 defines the function of the bits.

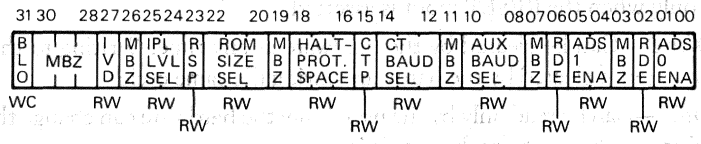


Figure 4 • MicroVAX 78332 Configuration Register Format

Table 8 • MicroVAX 78332 Configuration Register Description

Bit	Description
31	BLO (Battery Low)—Set if the $\overline{BLO}$ input is asserted when the RESET input is asserted and when the V_{DDI} inputs are grounded. The BLO bit can be cleared only by the user. If the BLO bit is set when the SSC is reset, the time-of-year clock is cleared.
30:28	MBZ (Must be zero)
27	IVD (Interrupt Vector Disable)—When set, the SSC does not produce interrupt vectors or assert the $\overline{RDY}$ output in response to an interrupt acknowledge cycle. The RESET input clears this bit.
26	MBZ (Must be zero)
25:24	IPL LVL SEL (Interrupt Priority Level Select)—These bits specify the IPL level to which the SSC level responds during an interrupt acknowledge cycle. These bits are set to the value that corresponds to the $\overline{IRQ<n>}$ line of the MicroVAX or CVAX CPU to which the SSC line is connected. For example, if the SSC $\overline{IRQ<n>}$ output is connected to the CPU IRQ2 line, bits 25:24 must be set to 10. The encodings are as follows

Bits 25	Bits 24	IPL Level	IRQ Line	Priority
0	0	14	IRQ0	lowest (default)
0	1	15	IRQ1	
1	0	16	IRQ2	
1	1	17	IRQ3	highest

Bit	Description
23	RSP (ROM Speed)—This bit selects the ROM access time. The RESET input clears this bit. The ROM access times are Bit 23 = 0: 350 ns (default) Bit 23 = 1: 250 ns
22:20	ROM SIZE SEL (ROM Size Select)—These bits specify the extent of the ROM address space. The RESET input clears these bits. The default ROM size is 8 KB. The encodings are defined below.

Bits 22	21	20	ROM Address Space	ROM Size (KB)
0	0	0	20040000-20041FFF	8 (default)
0	0	1	20040000-20043FFF	16
0	1	0	20040000-20047FFF	32
0	1	1	20040000-2004FFFF	64
1	0	0	20040000-2005FFFF	128
1	0	1	20040000-2007FFFF	256
1	1	0	20040000-200BFFFF	512
1	1	1	20040000-2013FFFF	1024

19	MBZ (Must be zero)
18:16	HALT PROT SPACE (Halt protect space)—Selects the halt-protected address space. The RESET input clears these bits. The lowest 8 KB of ROM space is halt-protected by default. The halt-protected address space may be larger than the ROM address space. The encodings are

Bits 18	17	16	Halt-protected ROM Address Space	Halt-protected Extent (
0	0	0	20040000-20041FFF	8 (default)
0	0	1	20040000-20043FFF	16
0	1	0	20040000-20047FFF	32
0	1	1	20040000-2004FFFF	64
0	0	2	0040000-2005FFFF	128
1	0	1	20040000-2007FFFF	256
1	1	0	20040000-200BFFFF	512
1	1	1	None	None

15	CTP (Control P Enable)—CTRL/P is recognized as a break in the console terminal UART if this bit is set. When this bit is cleared, 20 consecutive space bits are recognized as a break. This bit is cleared by the RESET input.

Bit	Description

14:12 CT BAUD SEL (Console terminal baud select)—Selects the baud rate of the console terminal UART. These bits are cleared by the RESET input. The default baud rate is 300. The boot code should write the proper value into the register before the first console access. The SSC baud clock runs about 1.75 percent faster. The bit baud rates available are

Bits 14	13	12	Baud Rate (default)
0	0	1	600
0	1	0	1200
0	1	1	2400
1	0	0	4800
1	0	1	9600
1	1	0	19.2 K
1	1	1	38.4 K

11 MBZ (Must be zero)

10:08 AUX BAUD SEL (Auxiliary baud select)—Selects the baud rate of the console storage UART. These bits are cleared by the RESET input. The default baud rate is 300. The boot code should write the proper value into the register before the first console access. The SSC baud clock runs about 1.75 percent faster. The baud rates available are

Bits 10	09	08	Baud Rate (default)
0	0	1	600
0	1	0	1200
0	1	1	2400
1	0	0	4800
1	0	1	9600
1	1	0	19.2 K
1	1	1	38.4 K

07 MBZ (Must be zero)

06 RDE (Ready Enable)—When set, the $\overline{\text{RDY}}$ output of Programmable Address Strobe Channel 1 is asserted eight SSC microcycles (nominally 800 nanoseconds) after the corresponding address strobe. When RDE is cleared, the SSC takes no action after asserting the address strobe. This bit is cleared by the RESET input.

05:04 ENA (Enable)—These bits enable the read and write channels of Address Strobe Channel 1 as follows:

ENA 05	04	Read	Write
0	0	disabled	disabled (default)
0	1	disabled	enabled
1	0	enabled	disabled
1	1	enabled	enabled

03 MBZ (Must be zero)

Bit	Description
02	RDE (Ready Enable)—When set, the $\overline{RDY}$ output of the Programmable Address Strobe Channel 0 is asserted eight microcycles (nominally 800 nanoseconds) after the corresponding address strobe. When RDE is cleared, the SSC takes no action after asserting the address strobe. This bit is cleared by the RESET input.
01:00	ENA (Enable)—These bits enable the read and write channels of Address Strobe Channel 0 as follows:

Bits		Read	Write
01	**00**		
0	0	disabled	disabled (default)
0	1	disabled	enabled
1	0	enabled	disabled
1	1	enabled	enabled

Bus Timeout Control Register

The SSC monitors the assertion and deassertion of the $\overline{AS}$ input to prevent disabling the system operation resulting from unanswered CPU read or write accesses, CVAX EP read or write accesses, or interrupt acknowledge cycles. The bus timeout is controlled by the Bus Timeout Control register that stores the required timeout interval. Each time the $\overline{AS}$ input is asserted, the SSC clears and starts an internal counter. When the $\overline{AS}$ input is deasserted, the counter is stopped. If the counter value becomes the same as the value in the Bus Timeout Control register, the counter is stopped, the $\overline{ERR}$ output is asserted, and the BTO bit 31 in this register is set. This indicates that the bus cycle should be aborted. The $\overline{ERR}$ output is deasserted when the $\overline{DS}$ input is deasserted. If the timed-out transaction is a CPU Read or CPU Write transaction, the RWT bit 30 is also set. This register is cleared by the RESET input. The register format is shown in Figure 5 and Table 9 defines the function of the bits.

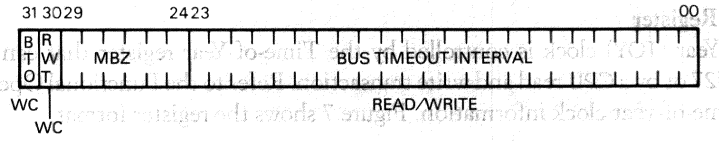


Figure 5 • MicroVAX 78332 Bus Timeout Control Register Format

Table 9 • MicroVAX 78332 Bus Timeout Register Description

Bits	Description
31	BTO (Bus Timeout)—When set, this bit indicates that a bus timeout has occurred during a transaction.
30	RWT (Read or Write Transaction)—When set, this bit indicates that a bus timeout has occurred during a CPU read or CPU write transaction.
29:24	MBZ (Must be zero)

Bits	Description
23:00	Bus Timeout Interval—These bits specify the timeout period. The available range of 1 to FFFFFF (hexadecimal) corresponds to a selectable timeout interval in the range of 1 microsecond to 16.77 seconds in 1 microsecond increments. Writing a zero to this field disables the bus timeout function.

4-Bit Output Port Register

The 4-bit Output Port register provides four data outputs OPT<3:0> that can be used to control LED indicators or similar devices. The data value in this register continually drives the outputs. This register is cleared during the powerup sequence. Figure 6 shows the register format and Table 10 lists the bit functions.

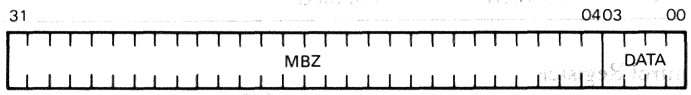


Figure 6 • MicroVAX 78332 Output Port Register Format

Table 10 • MicroVAX 78332 Output Port Register Description

Bits	Description
31:04	MBZ (Must be zero)
03:00	Data—The register data value that is continually driven on the $\overline{OPT<3:0>}$ output. Bit 03 corresponds to the $\overline{OPT3}$ output and bit 1 to the $\overline{OPT1}$ output.

Time-of-Year Register

The Time-of-Year (TOY) clock is controlled by the Time-of-Year register that can be addressed either as IPR #27 or by a CPU read and write transaction. Refer to the Functional Operation section for detailed time-of-year clock information. Figure 7 shows the register format.

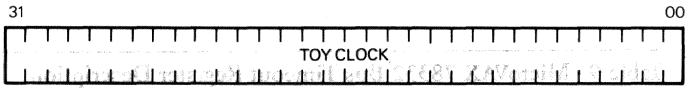


Figure 7 • Time-of-Year Register Format

Console Terminal and Console Storage UARTS

The SSC contains a console storage UART and a console terminal UART that operate similarly to Digital's DC319 DLART. Each UART contains four internal processor registers. The IPR numbers assigned to each register are listed in Tables 11 and 12.

Table 11 • MicroVAX 78332 Console Storage UART Registers

IPR Number	Register
28	Console Storage Receiver Status (CSRS)
29	Console Storage Receiver Data (CSRD)
30	Console Storage Transmitter Status (CSTS)
31	Console Storage Transmitter Data (CSTD)

Table 12 • MicroVAX 78332 Console Terminal UART Registers

IPR Number	Register
32	Console Receiver Control Status (RXCS)
33	Console Receiver Data Buffer (RXDB)
34	Console Transmitter Control/Status (TXCS)
35	Console Transmitter Data Buffer (TXDB)

These registers are typically accessed by external processor register protocols, but may also be accessed by CPU read and write transactions. Only 8-bit data and single stop bit operations without parity are supported.

The selectable baud rates are 300, 600, 1200, 2400, 4800, 9600, 19.2 K, and 38.4 K. Baud rate selection is achieved by writing to the SSC Configuration register. Framing and overrun errors are indicated by the setting of error bits in the receiver data registers CSRD and RXDB.

The Console Storage UART and Console Terminal UART are similar. The differences are noted in the register descriptions that follow.

Receiver Registers—Each UART has a Receiver Control and Status register (CSRS and RXCS) and a Receiver Data Buffer (CSRD and RXDB) register. These registers are cleared by the RESET signal. Figure 8 shows the format of the CSRD and RXCS registers and Table 13 describes the function of the bits. Figure 9 shows the format of the CSRS and RXCS registers and Table 14 lists the function of the bits.

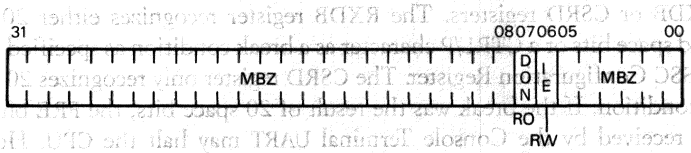


Figure 8 • MicroVAX 78332 Receiver Control and Status (CSRS/RXCS) Registers Format

Table 13 • MicroVAX 78332 Receiver Control and Status (CSRS/RXCS) Register Description

Bits	Description
31:08	MBZ (Must be zero)
07	DON (Done)—Set when a character is received. Cleared when the RXDB is read.

Bits	Description
06	IE (Interrupt enable)—Can be set or cleared by writing to the RXCS or CSRS registers. An interrupt is generated whenever IE and DON transitions to a one state. Interrupt requests are cleared when the corresponding interrupt request is acknowledged or by clearing the IE or DON bits.
05:00	MBZ (Must be zero)

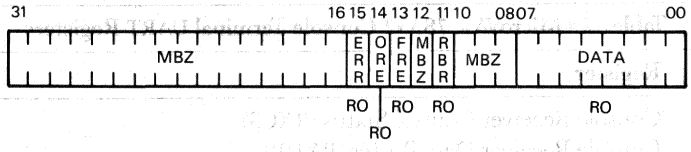


Figure 9 • MicroVAX 78332 Receiver Data Buffer (CSRD/RXDB) Register Format

Table 14 • MicroVAX 78332 Receiver Data Buffer (CSRD/RXDB) Register Description

Bits	Description
31:16	MBZ (Must be zero)
15	ERR (Error)—Set if ORE bit 14 or FRE bit 13 are set. These error bits are updated when data is loaded into the register and cleared when the RXDB or CSRD registers are read.
14	ORE (Overrun error)—The receiver is double buffered. If both buffers are full when data is received, the assembly register is overwritten and this bit is set when the overwriting character is loaded into RXDB or CSRD register. When set, the ERR bit 15 as also set.
13	FRE (Framing error)—Set if a framing error occurs. When set, the ERR bit 15 is also set.
12	MBZ (Must be zero)
11	RBR Received Break—Set if the receiver detects a break condition. Cleared by reading the RXDB or CSRD registers. The RXDB register recognizes either 20 consecutively received space bits or a CTRL/P character as a break condition as specified the CTP bit 15 of the SSC Configuration Register. The CSRD register only recognizes 20 space bits as a break condition. If the break was the result of 20 space bits, the FRE bit 13 is also set. Breaks received by the Console Terminal UART may halt the CPU. However, breaks received by the Console Storage UART will not halt the CPU.
10:08	MBZ (Must be zero)
'07:00	Data—Contains the received data.

Transmitter Registers—Each UART has a Transmitter Control and Status register (CSTS and TXCS) and a Transmitter Data Buffer register (CSTD and TXDB). The data to be sent out is written into bits 07:00 of the CSTD or TXDB registers which are cleared by the RESET input. The format of the of the CSTS and TXCS registers is shown in Figure 10 and Table 15 defines the function of the bits. Figure 11 shows the format of the CSTD and TXDB registers and Table 16 defines the function of the bits.

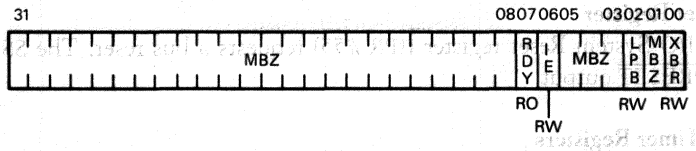


Figure 10 • MicroVAX 78332 Transmitter Control and Status (CSTS/TXCS) Registers Format

Table 15 • MicroVAX 78332 Transmitter Control and Status (CSTS/TXCS) Registers Description

Bits	Description
31:08	MBZ (Must be zero)
07	(RDY) Ready—Set by the hardware when the transmitter data buffer (CSTD and TXDB) registers are available to accept data or when the RESET signal is deasserted. Writing a character to the CSTD or TXDB register causes the UART to send the character and to clear this bit until the character is transferred to the serialization buffer. This bit is set by the RESET input.
06	IE (Interrupt enable)—Set by the software. Cleared by the RESET input, when the corresponding interrupt request is acknowledged, or when this bit or the RDY bit 07 is cleared. An interrupt is generated when IE and RDY transition to a 1.
05:03	MBZ (Must be zero)
02	LPB (Loopback)—Setting this bit connects the transmitter serial output to the receiver serial input. Also sets the external serial-output pin to MARK. This bit is cleared by the RESET input.
01	MBZ (Must be zero)
00	XBR (Transmit break)—Set when the UART sets serial output line to the space condition when it has finished transmitting the current character. Clearing this bit terminates the break. This bit is cleared by the RESET input.

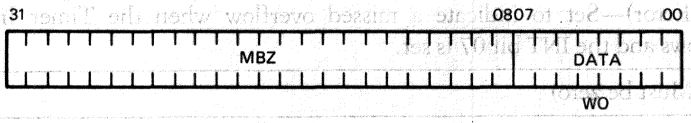


Figure 11 • MicroVAX 78332 Transmitter Data Buffer (CSTD/TXDB) Registers Format

Table 16 • MicroVAX 78332 Transmitter Data Buffer (CSTD/TXDB) Registers Description

Bits	Description
31:08	MBZ (Must be zero)
07:00	Data—Data to be transmitted.

I/O System Reset Register

Writing to the I/O System Reset register (IPR #55) requests a bus reset. The SSC responds by asserting the $\overline{\text{IORESET}}$ output.

Programmable Timer Registers

The SSC includes general purpose programmable timers 0 and 1 that are similar to the VAX Interval Clock. Each programmable timer consists of four I/O space registers: the Timer Control register, the Timer Interval register, the Timer Next Interval register, and the Timer Interrupt Vector register. A timer is programmed by loading the negative (two's complement) of the desired interval value into the Timer Next Interval Count register. The timer is started by writing a 51 (hexadecimal) into the Timer Control register. An interrupt will then occur every interval count or microsecond.

Timer Control Registers 0 and 1—The configuration of the Timer Control register is shown in Figure 12 and Table 17 defines the function of the bits in the register. Control bit 02 (STP) has been added to the configuration of the standard VAX register to stop the timer when an overflow occurs. The overflow condition causes an interrupt request on an $\overline{\text{IRQ<n>}}$ line at the user-selected IPL level. The interrupt vector is also user programmable. These registers are cleared by the RESET input.

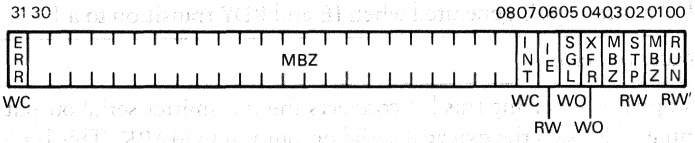


Figure 12 • MicroVAX 78332 Timer Control Registers (0 and 1) Format

Table 17 • MicroVAX 78332 Timer Control Registers (0 and 1) Description

Bit	Description
31	ERR (Error)—Set to indicate a missed overflow when the Timer Interval register overflows and the INT bit 07 is set.
30:08	MBZ (Must be zero)
07	INT (Interrupt)—Set when the Timer Interval register overflows. If the IE bit 06 is set when the INT bit is set, an interrupt request is posted.
06	IE (Interrupt enable)—Set or cleared by the software to indicate that an interrupt request should be posted when the INT bit 07 is set.
05	SGL (Single)—When the RUN bit 00 is cleared, writing a 1 to this bit causes the Timer Interval register to be incremented by a value of 1. When the Timer Interval Count (ICR) overflows because of the assertion of SGL, STP is ignored and the counter is reloaded. When the RUN bit 00 or XFR bit 04 are set, write operations to the SGL bit are ignored. This bit is always read as a zero.

Bit	Description
04	XFR (Transfer)—Writing a one to this bit causes the Timer Next Interval register to be copied to the Timer Interval register. This bit is always read as a zero.
03	MBZ (Must be zero)
02	STP (Stop)—This bit determines whether the timer stops after it overflows. When this bit is set, the RUN bit 00 is set, and the Timer Interval register overflows, the RUN bit is cleared and the counting stops.
00	RUN—When this bit is set, the Timer Interval register is incremented once per microsecond. The INT bit 07 is set when the timer overflows. If the STP bit is set when the timer overflows, the RUN bit is cleared by the hardware.

Timer Interval Count Registers (0 and 1)—These registers contain the interval count value. Figure 13 shows the format of the register information.

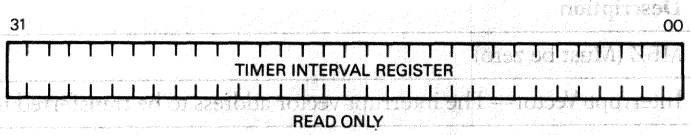


Figure 13 • MicroVAX 78332 Timer Interval Count Registers (0 and 1) Format

Timer Next Interval Count Registers (0 and 1)—These registers contain the value that is loaded into the Timer Interval Count registers after an overflow has occurred or in response to writing a 1 to set XFR (bit 04) of the Timer Control register. This register is cleared by the RESET input. The format for the register information is shown in Figure 14.

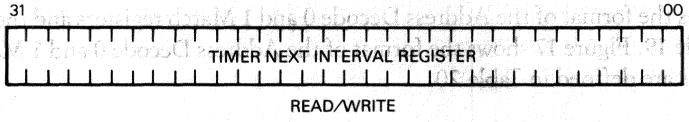


Figure 14 • MicroVAX 78332 Timer Next Interval Count Registers (0 and 1) Format

Timer Interrupt Vector Registers (0 and 1)—These registers store the interrupt vector value to be transferred to the CPU. An interrupt request is posted when the IE bit 06 and INT bit 07 are transitioned to a 1. When the SSC detects an interrupt acknowledge cycle and one of the timers is set to the highest internal priority requesting an interrupt, the interrupt vector for that timer is

transferred to the DAL. The corresponding interrupt request is then cleared. Interrupt requests can also be cleared by clearing the IE or the INT bits. Timer 0 has the higher priority. The Interrupt Vector registers are cleared by the RESET input. The format of the Timer Interrupt Vector register is shown in Figure 15. Table 18 defines the register bits.

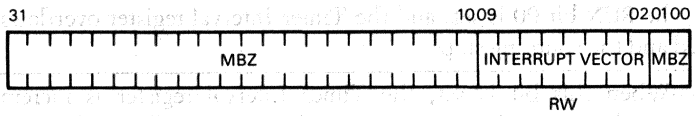


Figure 15 • MicroVAX 78332 Timer Interrupt Vector Registers (0 and 1) Format

Table 18 • MicroVAX 78332 Timer Interrupt Vector Registers (0 and 1) Description

Bit	Description
31:10	MBZ (Must be zero)
09:02	Interrupt Vector—The interrupt vector address to be transferred to the CPU.
01:00	MBZ (Must be zero)

Decode Channels

The Programmable address decoders are used to decode the address on the DAL to select channel 0 or 1. Each channel consist of an address decode channel Mask and Match register. When the $\overline{AS}$ input is asserted and the bus cycle is a CPU read or write transaction, the address on the DAL is compared to all the bits of the Match register for which the corresponding Mask register bit is zero. If the comparison is successful, the corresponding output strobe is asserted. The $\overline{ADS0}$ output to the external logic is asserted to select Channel 0 and the $\overline{ADS1}$ is asserted to select Channel 1. Bits 06:04 of the SSC Configuration register control the operation of the programmable address decoder for Channel 1 and bits 02:00 control the operation of Channel 0. When the RESET input is asserted, both output strobes are disabled and the Match and Mask registers are cleared.

Figure 16 shows the format of the Address Decode 0 and 1 Match registers and the register bits are defined in Table 19. Figure 17 shows the format of the Address Decode 0 and 1 Mask registers and the register bits are defined in Table 20.

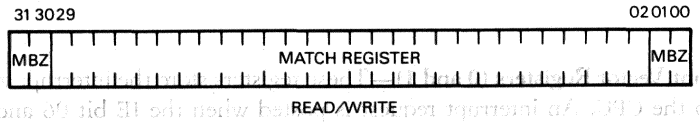


Figure 16 • MicroVAX 78332 Address Decode Channel (0 and 1) Match Registers Format

Table 19 • MicroVAX 78332 Address Decode Channel (0 and 1) Match Registers Description

Bit	Description
31:30	MBZ (Must be zero)
29:02	MATCH—Contains the address to be compared with the DAL address.
01:00	MBZ (Must be zero)

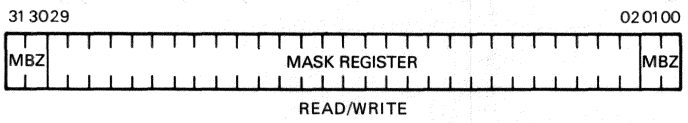


Figure 17 • MicroVAX 78332 Address Decode Channel (0 and 1) Mask Register Format

Table 20 • MicroVAX 78332 Address Decode Channel (0 and 1) Mask Registers Description

Bit	Description
31:30	MBZ (Must be zero)
29:02	MASK—Each of the bits that is to correspond to an address bit of the Match register is cleared. The remaining bits are set.
01:00	MBZ (Must be zero)

Test Registers

The ROM Data register (BA + 0F0), Bus Timeout Counter (BA + 0F4) and the Interval Timer (BA + 0F8) registers are used for test purposes during manufacturing and should not be accessed by the user. The results of such accesses are unpredictable.

• Functional Description

The SSC interfaces directly to the MicroVAX bus as shown in Figure 18. It contains two programmable address decoders that can be used to control external device operation and two serial line UARTs for a console and auxiliary device. It includes an internal RAM and provides support for an external ROM. The $\overline{\text{IAKEO}}$ output provides interrupt acknowledge support for other interfaces on the bus.

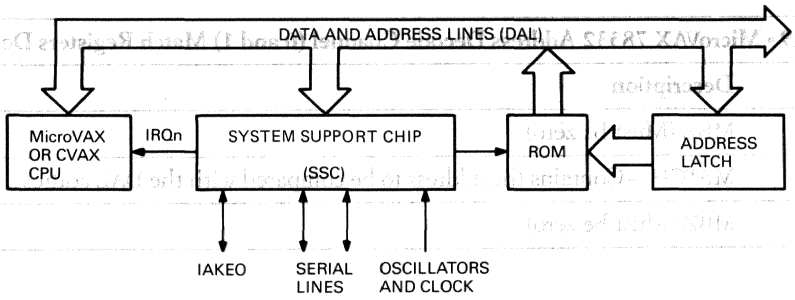
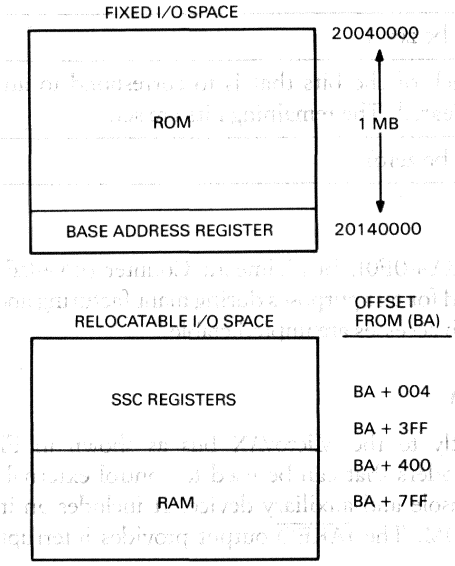


Figure 18 • MicroVAX 78332 CPU to SSC Interface System Configuration

I/O Space Assignments

The SSC operates with the fixed I/O space and relocatable I/O space shown in Figure 19. The address assignments for the external ROM and Base Address register are within the fixed I/O space. The internal SSC registers and RAM are assigned to the relocatable I/O space.



NOTE:
BA = CONTENTS OF THE BASE ADDRESS REGISTER.

Figure 19 • MicroVAX 78332 I/O Space Allocations

The ROM is controlled by an external address latch that stores the appropriate bits of DAL<29:02>. The SSC drives the ROM chip select inputs and the data lines of the ROM connect directly to the DAL as shown. The ROM address space begins at address 20040000. The ROM must contain at least 8 KB (default). A larger ROM size can be selected by writing to SSC Configuration register bits 22:20 before making a ROM access at an address greater than 8 KB. The ROMs can be 16, 32, 64, 128, 256, 512 KB and 1 MB. The SSC responds to all CPU reads operations within the ROM space. Write operations to a space other than those specified are ignored. The SSC can be used with fast or slow ROMs. When bit 23 of the Configuration register is set, the ROM access time is 250 nanoseconds. When it is cleared, the access time is 350 nanoseconds which is the default condition. Except for write operations to the RAM, all accesses are 32-bits wide and longword aligned.

The RAM and the SSC registers are located at fixed offsets from the BA address of 20140000 loaded into the SSC Base Address register. The DAL<29:02> contain the longword address of the storage element being accessed. Byte writes to the RAM are specified by DAL<29:02> and the byte mask ($\overline{BM}$<3:0>).

Interrupt Logic

Because the SSC does not contain an $\overline{IAKEI}$ input, it must be assigned the highest external device priority that responds to interrupt acknowledge cycles at its designated interrupt level. The $\overline{IRQ}$<n> output from the SSC connects to the appropriate CPU $\overline{IRQ}$<n> line and the Configuration register bits <25:24> must be set to specify this level.

Interrupt Requests—The SSC requests an interrupt by asserting the $\overline{IRQ}$<n> output when any of the conditions listed in Table 21 occur if their respective Interrupt Enable bits are set.

Table 21 • MicroVAX 78332 Interrupt Requests Priority and Vector Address

Priority Assignment	Condition	Vector Address
1	Console Terminal UART Receiver Ready	000000F8
2	Console Terminal UART Transmitter Ready	000000FC
3	Console Storage UART Receiver Ready	000000F0
4	Console Storage UART Transmitter Ready	000000F4
5	Timer 0 ICR Overflow	*
6	Timer 1 ICR Overflow	*

*User programmable and stored in Interrupt Vector registers.

- The following conditions will cause the SSC to generate an interrupt request.
- Console Terminal UART Receiver Ready—When the IE bit 06 and the DON bit 07 of the Console Terminal Receiver Control and Status register transition to a 1.
- Console Terminal UART Transmitter Ready—When the IE bit 06 and the RDY bit 07 of the Transmit Control and Status register transition to a 1.
- Console Storage UART Receiver Ready—When IE bit 06 and the DON bit 07 of Console Storage Receiver Status register transition to a 1.

- Console Storage UART Transmitter Ready—When the IE bit 06 and the RDY bit 07 of the Console Storage Transmitter register transition to a 1.
- Timer <0:1> ICR overflow—When the IE bit 06 and INT bit 07 of the Timer Control register transition to a 1.

Interrupt Acknowledge—The SSC responds to interrupt acknowledge cycles as follows:

- If the interrupt acknowledge is not at the IPL level specified by the SSC or if no internal SSC interrupts are pending, the SSC asserts the $\overline{\text{IAKEO}}$ output to indicate that it has no interrupts pending at the given IPL level.
- If the interrupt acknowledge is at its IPL level, and if there is at least one internal SSC interrupt pending, the response of the SSC depends on the state of the IVD bit 27 of the Configuration register. If the IVD is cleared, the SSC places the interrupt vector of the highest priority internal interrupt pending onto the DAL, clears its corresponding internal interrupt request, and if no other internal interrupts are pending, deasserts the IRQ<n> output.
- If IVD is set, the SSC clears its internal interrupt request for the highest priority pending internal interrupt. If there are no other internal interrupts are pending, it deasserts the $\overline{\text{IRQ<n>}}$ output.
- The deassertion of the $\overline{\text{DS}}$ input causes the deassertion of the $\overline{\text{IAKEO}}$ output.

Break Detect/Transmit Logic

The Console Terminal and Console Storage UARTs include break detection and transmit logic. The UART registers are described in Register section.

Break Detect—The Console Terminal UART recognizes either 20 consecutively received space bits (default) or a CTRL/P command received as a break condition, as determined by the CTP bit 15 of the Configuration register. The break generates a CPU halt operation if the halts are enabled and the console is not secured. The Console Storage UART recognizes only 20 consecutively received space bits as a break condition. A break received by the Console Storage UART cannot generate a CPU halt. If either UART detects a valid break condition, The RBR bit 11 of the RXDB or CSRD registers are set. If the break was the result of 20 consecutively received space bits, the FRE bit 13 is also set. The RBR is cleared by reading the RXDB or CSRD registers and can be set only by a break condition received by the UART.

Break Transmit—Setting the XBR bit 00 of the TXCS or CSTS registers causes the UART to set the serial output line to the space condition. Clearing the bit terminates the break. The UART does not react to a change in the state of XBR bit until it has finished transmitting the current character. When the XBR bit is set, the transmitter operates normally but the output line remains low. Therefore, the user can send dummy characters in order to time the break. After clearing the XBR bit, the user can provide an extended MARK character by allowing the transmitter to idle for the desired period.

Halt Arbitration Logic

The console terminal UART can request a CPU halt when a break condition is detected if the $\overline{\text{SECCON}}$ input is not asserted. A CPU halt request can also be generated if the $\overline{\text{HALTIN}}$ input from the external logic is asserted. Either of these conditions normally results in the assertion of the

$\overline{\text{CPUHALT}}$ output of the SSC that connects to the $\overline{\text{HALT}}$ input of the CPU. The halt arbitration logic of the SSC may conditionally prevent the CPU halt request as described.

The halt-protected address space of the ROM is defined by bits 18:16 of the SSC Configuration register. A CPU halt request is disabled when halt-protected space is accessed by an I-stream read transaction. On each I-stream read transaction, the SSC determines whether the target address is in the halt-protected space. If the address is in this space, then the $\overline{\text{CPUHALT}}$ output is disabled until the next I-stream read transaction. If the address is not in the halt-protected space, then the $\overline{\text{CPUHALT}}$ output is asserted until the next I-stream read transaction occurs.

Assertion of the $\overline{\text{HALTIN}}$ input will assert the $\overline{\text{CPUHALT}}$ output if the halt condition is enabled. If the $\overline{\text{HALTIN}}$ output is asserted when the halt functions are disabled and the halt is then enabled, the SSC asserts $\overline{\text{CPUHALT}}$ until halts are again disabled or until the $\overline{\text{HALTIN}}$ output is deasserted.

If a break is received by the Console Terminal UART when the halt requests are enabled and the console is not secured ($\overline{\text{SECCON}}$ deasserted), the SSC asserts the $\overline{\text{CPUHALT}}$ output until the halt conditions are disabled, the console is secured, or the break condition is cleared by software.

The software can execute a kernel mode HALT command to cause a CPU halt when the halts are disabled. Asserting the RESET input will also enable the halt requests. The $\overline{\text{RUN}}$ output is asserted when the halt conditions are enabled.

When used with the CVAX CPU, the SSC does not detect I-stream references that are directed to the CVAX CPU internal cache. When used with the MicroVAX CPU, a copy (MOVC) instruction to the halt-protected locations disables the halt requests until the copy transaction is complete.

Bus Support Logic

Each time the $\overline{\text{AS}}$ input is asserted, the SSC clears and starts an internal counter. When $\overline{\text{AS}}$ is deasserted, the counter is stopped. When the counter reaches a value equal to the value loaded into the Bus Timeout Control register, the counter is stopped, the BTO bit 31 in this register is set, and the $\overline{\text{ERR}}$ output is asserted to indicate that the bus cycle should be aborted. If the timed-out transaction was a CPU read or CPU write, the RWT bit 30 is also set. The $\overline{\text{ERR}}$ output remains asserted until the $\overline{\text{AS}}$ input is deasserted.

The SSC includes logic to recognize an external processor (EP) write cycle to the I/O System RESET register (IPR #55). In Q-bus systems, this typically indicates a request for an I/O system reset.

If the write operation is performed by a MicroVAX EP cycle, the SSC responds by asserting $\overline{\text{IORESET}}$ output for two SSC microcycles (nominally 200 nanoseconds) following the completion of the external processor write command cycle.

If the write operation is performed by a CPU write or CVAX EP write cycle, the SSC responds by asserting $\overline{\text{IORESET}}$ output for eight SSC microcycles (nominally 800 nanoseconds) after which the $\overline{\text{RDY}}$ output is asserted.

The falling edge of the $\overline{\text{IORESET}}$ signal can be used to reset the I/O system. The $\overline{\text{IORESET}}$ signal is deasserted when the $\overline{\text{DS}}$ input is deasserted.

Realtime Clock and Interval Timer

The Time-of-Year (TOY) clock is an unsigned 32-bit binary counter whose least-significant bit represents a resolution of 10 milliseconds. It consists of one longword register that is typically addressed as external processor register IPR #27 but may also be accessed by CPU read and write transactions. The register counts only when it contains a nonzero value. The TOY clock is driven from the TB25K input by an external 25.6-KHz oscillator. If TB25K is connected to ground, the

timebase for the clock is supplied by the 40-MHz oscillator at the TB40M input which also provides the timebase for the baud rate generator and the Interval and Programmable timers.

The counter function is maintained during powerfail conditions by the battery-backup supply to the SSC and to the 25.6-KHz external oscillator. If the BLO bit 31 (Battery low) of the Configuration register is set, the SSC is reset and the register is cleared and remains cleared until a nonzero value is written by software.

The Interval timer provides a 100-Hz input to the TOY clock and to the $\overline{\text{INTCLKO}}$ output. It can be used as the $\overline{\text{INTTIM}}$ input to the CPU which drives the CPU ICCS register (IPR #24).

Programmable Address Decoder

The programmable address decoders (channel 0 and channel 1) selectively decode bus addresses during CPU read and write transactions to generate address strobe signals for external devices. Each address decoder consists of a Match register and Mask register which are within the relocatable I/O address space.

When the $\overline{\text{AS}}$ input is asserted, the address on DAL <29:02> is compared with all the corresponding bits of the Match register that have been selected. The Match register bits that are to be compared with bus address bits are selected by the Mask register. If a Mask register bit is zero, then the corresponding Match register bit will be used in the comparison. The remaining bits of the Match register that are not selected by a Mask register bit are not used in the comparison.

When a match exists, an output strobe $\overline{\text{ADS1}}$ (channel 1) or $\overline{\text{ADS0}}$ (channel 0) is asserted between one and two SSC microcycles (nominally 100 to 200 nanoseconds) after the assertion of the $\overline{\text{AS}}$ input provided that the bus cycle is a CPU read or write transaction and the assertion of the strobe is enabled by ENA bits 05:04 for channel 1 or bits 03:02 for channel 2 of the Configuration register.

After the $\overline{\text{ADS1}}$ or $\overline{\text{ADS2}}$ output strobe is asserted, the SSC can assert the $\overline{\text{RDY}}$ output eight microcycles (nominally 800 nanoseconds) later to permit the external device time to respond. The $\overline{\text{RDY}}$ output is controlled by the RDE bits in the Configuration register. The deassertion of the $\overline{\text{DS}}$ input causes the deassertion of the address strobe.

The address decoders for channel 0 or 1 should be not be programmed with the $\overline{\text{RDY}}$ signal asserted if another device in the system can respond to the read or write transaction programmed into that channel or if the programmed address is located within the SSC ROM, RAM, or I/O register address space.

When RESET is asserted, the $\overline{\text{ADS}}$<0:1> output strobes are disabled and the Match and Mask registers are cleared.

Some examples of implementing the address strobes are

- A channel can be programmed to respond to a single longword read address. The $\overline{\text{ADS0}}$ or $\overline{\text{ADS1}}$ strobe is used to gate the value selected by external switches to the DAL. The SSC can then assert the $\overline{\text{RDY}}$ output to complete the cycle.
- A channel can be programmed to decode only some of the high-order DAL. The strobe is then used with an external decoder to select other devices.
- An address strobe can drive the chip select (CS) input of another peripheral chip such as a direct memory access (MicroDMA) or vectored interrupt controller (MicroVAX VIC). The peripheral chip must then assert the $\overline{\text{RDY}}$ output.

Modes of Operation

The SSC operates in normal mode and battery-backup mode. In normal mode the system is powered up and running. The input dc power is supplied by the system power supply and the RESET input is deasserted. In the battery-backup mode, the system is powered down, but the SSC receives power from an external battery and the RESET input is asserted. RESET is asserted before the initial powerup sequence and is deasserted after the system is in normal operation. It is also asserted before the transition to battery-backup mode and is deasserted when normal operation is resumed.

Bus Transactions

The SSC supports CPU read, CPU write, MicroVAX external processor (EP) write command, MicroVAX EP write data, MicroVAX EP read response, CVAX EP read, CVAX EP write, idle, and interrupt acknowledge. The SSC interfaces asynchronously to the MicroVAX CPU or CVAX CPU. Refer to the *ac Specifications* section for the transaction timing diagrams described in the following paragraphs.

CPU Read, CVAX EP Read, or Interrupt Acknowledge

During CPU read transactions or interrupt acknowledge cycles, the CPU addresses the SSC or the external ROM to receive data. A CPU read transaction requires a minimum of six SSC microcycles (nominally 600 nanoseconds). The $\overline{WR}$ input is unasserted and the byte mask $\overline{BM}<3:0>$ information is ignored. The SSC latches the information on DAL<31:00>, $\overline{WR}$, and CS<2:0> when the $\overline{AS}$ input is asserted. The type of read access is determined by the CS<2:0> input.

During the first part of a read cycle, the CPU transfers the address on the DAL. If the access is a longword read, the CPU transfers the physical longword address on DAL<29:02>. If the access is a CVAX EP Read, the CPU transfers the processor register number on DAL<07:02> and zeros on DAL<10:08>. For a CVAX system if the access is an interrupt acknowledge cycle, the CPU transfers the priority of the interrupt being acknowledged (IPL) on DAL<06:02>. The DAL<31:07> and DAL<01:00> = 0. With a MicroVAX system, the IPL is on DAL<04:00>, DAL<31:30> = 10, and DAL<29:05> = 0. The CPU then asserts the $\overline{AS}$ input to indicate that the address is valid. When no device responds to the address, the SSC may assert the $\overline{ERR}$ output to indicate that a bus timeout has occurred.

During the second part of a read cycle, the CPU accepts the addressed data from the DAL. If the access is to internal storage, the SSC transfers the required data on DAL<31:00> and asserts the $\overline{RDY}$ output. The CPU reads the data and deasserts the $\overline{AS}$ and $\overline{DS}$ inputs to end the read transaction. If the access is directed at external bytewide ROM, the SSC asserts the $\overline{ROMEN}$ signal when $\overline{DS}$ is asserted and then performs four ROM read sequences. It latches the ROM data and increments the ROMADR<1:0> output after each read operation. The SSC then deasserts the $\overline{ROMEN}$ output, transfers the unpacked longword onto DAL<31:00>, and asserts the $\overline{RDY}$ output. The CPU reads this data and deasserts the $\overline{AS}$ and $\overline{DS}$ inputs to end the read transaction.

If the access is directed to an external wordwide ROM, the SSC asserts the $\overline{ROMEN}$ output when the $\overline{DS}$ input is asserted and then performs two ROM read sequences. It latches the ROM data after each read operation and inverts the ROMADR1 after the first read. The SSC then deasserts the $\overline{ROMEN}$ output, transfers the unpacked longword to DAL<31:00>, and asserts the RDY output. The CPU reads the data from the DAL and deasserts the $\overline{AS}$ and $\overline{DS}$ inputs to end the read transaction.

If the access is directed at external longwordwide ROM, the SSC asserts the $\overline{ROMEN}$ output when the $\overline{DS}$ input is asserted and latches the ROM data when it is valid. The SSC then deasserts

$\overline{\text{ROMEN}}$, transfers the latched longword on DAL<31:00>, and asserts the $\overline{\text{RDY}}$ output. The CPU reads the ROM data and deasserts the $\overline{\text{AS}}$ and $\overline{\text{DS}}$ outputs to end the read transaction.

During an interrupt acknowledge cycle, the SSC transfers the interrupt vector data on DAL<09:02> and asserts the $\overline{\text{RDY}}$ output. The CPU reads this data and then deasserts the $\overline{\text{AS}}$ and $\overline{\text{DS}}$ outputs to end the read transaction.

CPU Write and CVAX EP Write

During a write cycle, the CPU writes information to storage elements in the SSC. A write cycle requires six SSC microcycles (nominally 600 nanoseconds). The first half of a write cycle is similar to a CPU read transaction except that the $\overline{\text{WR}}$ input is asserted. The CPU transfers the address and the operand length onto DAL<31:02> and asserts the $\overline{\text{AS}}$ input. If the access is directed to internal storage in the SSC, the SSC latches the data from DAL<31:00> after the $\overline{\text{DS}}$ input is asserted. The $\overline{\text{BM}}$<3:0> lines specify which bytes of the target longword should be written. The SSC stores the data and asserts $\overline{\text{RDY}}$ output. The CPU then deasserts $\overline{\text{AS}}$ and $\overline{\text{DS}}$ signals to end the write transaction. Write transactions to the ROM address space are ignored.

If a device does not responds to the address, a bus timeout may occur and the SSC will assert the $\overline{\text{ERR}}$ output.

MicroVAX External Processor Register Transactions

The SSC responds to two sequences of MicroVAX External Processor Register transactions: an EP Write command followed by an EP Read Response and an EP Write command followed by EP Write Data command.

During an EP Write command/EP Read Response, the CPU reads data from the SSC. In the first part of the transaction, the CPU performs an EP Write Command transaction. The DAL<05:00> contain the address of the required register and DAL31 is a 1 to indicate that the read transaction will follow.

The SSC latches the DAL<31:00> information on the rising edge of $\overline{\text{EPS}}$ input. During the next two SSC microcycles (one MicroVAX microcycle is nominally 200 nanoseconds), the SSC accesses and stores the requested data. After this delay, the CPU executes an EP Read Response cycle during which the SSC uses the $\overline{\text{EPS}}$ signal as a strobe to transfer the data to DAL<31:00> and to pulldown the $\overline{\text{CS2}}$ output level.

EP Write Command/Write Data

The CPU writes data to the SSC during this transaction. In the first part of the transaction, the CPU performs an EP Write Command transaction. The DAL<05:00> specify the location of the required register. When DAL31 bit is a 0, a write transaction will follow. The SSC transfers the DAL<31:00> information on the rising edge of $\overline{\text{EPS}}$ input. In the next MicroVAX microcycle, the SSC latches the data from DAL<31:00> on the rising edge of EPS. The SSC stores the data internally during the following two SSC microcycles. No accesses may therefore be directed at the SSC for two SSC microcycles (one MicroVAX microcycle is nominally 200 nanoseconds) after an EP Write Data transaction.

Transaction Time Estimates

Table 22 shows the estimated maximum transaction time for a longword, word, and byte transfer when the CPU and the SSC are operating at 40 MHz.

Table 22 • MicroVAX 78332 SSC Estimated Transaction Times

Access Type	MicroVAX			CVAX		
	Longword	Word	Byte	Longword	Word	Byte
EP Read	600	—	—	600	—	—
EP Write	800	—	—	600	—	—
Internal RAM Read	800	—	—	700	—	—
Internal RAM Write	800	—	—	600	—	—
250 ns External ROM	1000	1400	2000	1000	1300	1900
350 ns External ROM	1000	1600	2400	1100	1500	2300

• Interfacing Requirements

Figure 20 shows a typical system interconnection of the SSC and MicroVAX CPU or CVAX CPU. The input and output signals between the SSC, terminals, and external devices are also shown.

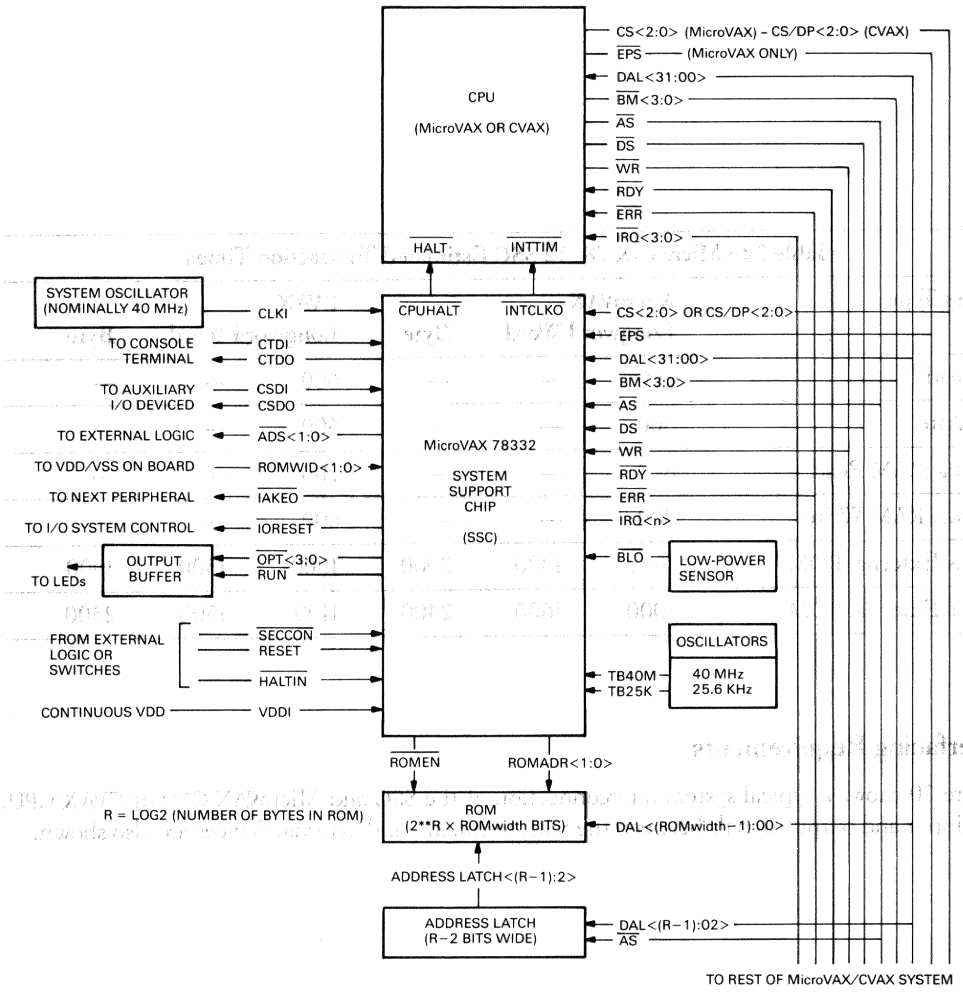


Figure 20 • MicroVAX 78332 SSC to CPU Typical Interconnections

Power Supply Interfacing

A typical block diagram of the power supply and external power control circuit is shown in Figure 21. The power circuit provides continuous power to the Time-of-Year (TOY) clock and to the RAM circuits to maintain memory data during a power interruption or failure. During normal operation, the power supply provides both V_{DDI} and V_{DDX} voltages to the SSC. During battery backup mode, the power source is switched by the power control logic to the battery and the V_{DDI} input provides the power to maintain the TOY clock and RAM. The V_{DDI} also provides continuous power to the external 25.6-Hz oscillator to maintain the TOY clock operation.

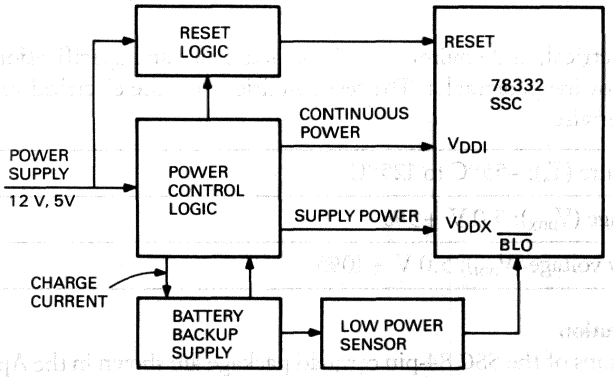


Figure 21 • MicroVAX 78332 Power Supply Interconnection

The power connections to the SSC are shown in Figure 22. Each V_{DDI} and V_{DDX} pin should be bypassed to V_{SS} with a 0.01-F capacitor located as close to the package pin as possible. All V_{DDX} pins should connect to the same supply. Both V_{DDI} pins connect together and are bypassed with a single 0.33-F capacitor.

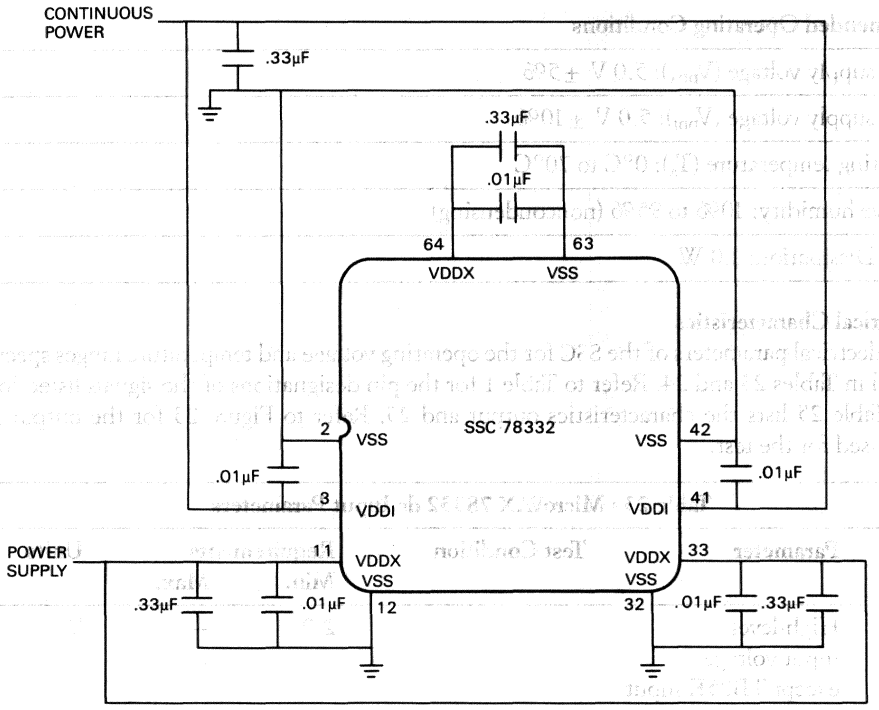


Figure 22 • MicroVAX 78332 Power Supply Connections

• Specifications

The mechanical, electrical, and environmental characteristics and specifications for the SSC are described in the following paragraphs. The test conditions for the electrical values are as follows unless specified otherwise.

- Ambient temperature (T_A): −55°C to 125°C
- Power supply voltage (V_{DDX}): 5.0 V ±5%
- Continuous supply voltage (V_{DDI}): 5.0 V ±10%

Mechanical Configuration

The physical dimensions of the SSC 84-pin cerquad package are shown in the Appendix.

Absolute Maximum Ratings

Stresses greater than the absolute maximum ratings may cause permanent damage to the device. Exposure to the absolute maximum ratings for extended periods may adversely affect the reliability of the device.

- Supply voltage (V_{DD}): 0.5 V to 7.0 V
- Input (V_{in}) and output voltage (V_{out}): 0.5 V to 7.0 V
- Ambient temperature (T_A): 0°C to 70°C
- Storage temperature (T_S): 55°C to 125°C

Recommended Operating Conditions

- Power supply voltage (V_{DDX}): 5.0 V ±5%
- Power supply voltage (V_{DDI}): 5.0 V ±10%
- Operating temperature (T_A): 0°C to 70°C
- Relative humidity: 10% to 95% (noncondensing)
- Power Dissipation: 1.0 W

dc Electrical Characteristics

The dc electrical parameters of the SSC for the operating voltage and temperature ranges specified are listed in Tables 23 and 24. Refer to Table 1 for the pin designations of the signals listed in the tables. Table 25 lists the characteristics output and 25. Refer to Figure 23 for the output load circuits used for the test.

Table 23 • MicroVAX 78332 dc Input Parameters

Symbol	Parameter	Test Condition	Requirements Min.	Requirements Max.	Units
V_{IH}	High-level input voltage except TB25K input		2.0	—	V
	TB25K input		3.2	—	V

Symbol	Parameter	Test Condition	Requirements Min.	Max.	Units
V_{IL}	Low-level input voltage		—	0.8	V
I_{IL}	Input leakage current	$0 < V_{in} < 5.25$ V	-10	10	MA
C_{in}	Input capacitance except TB25K input		—	12	pF
	TB25K input		—	15	pF

Table 24 • MicroVAX 78332 dc Output Parameters

Symbol	Parameter	Test Condition	Requirements Min.	Max.	Units	Load Circuit
V_{OH}[1]	High-level output voltage	$I_{OL} = 3.2$ mA	—	0.4	V	Fig 23A
V_{OL}[1]	Low-level output voltage	$I_{OH} = -2.0$ mA	4.0	—	V	Fig 23A
V_{OL}[2]	Low-level output voltage	$I_{OL} = 23$ mA	—	0.4	V	Fig 23B
I_{OH}[2]	Output leakage current	$0 < V_{OH} < V_{DD}$	—	10	A	Fig 23B
V_{OH}[3]	High-level output voltage	$I_{OH} = -8.0$	$V_{DD}-1.0$	—	V	Fig 23C
V_{OL}[3]	Low-level output voltage	$I_{OL} = 10$ mA		0.4	V	Fig 23C

[1]Outputs DAL<31:00>, CTDO, CSDO, $\overline{\text{CPUHALT}}$, $\overline{\text{ADS}}$<1:0>, $\overline{\text{INTCLKO}}$, $\overline{\text{IORESET}}$, $\overline{\text{IAKEO}}$, ROMADR<1:0>, and $\overline{\text{ROMEN}}$. Output signals capable of driving a fan-out load of eight LSTTL loads or two standard TTL loads.

[2]Outputs $\overline{\text{ERR}}$, $\overline{\text{RDY}}$, $\overline{\text{IRQ}}$<n>, and CS2. Open-drain pulldown output capable of operating with a 250- pullup resistor.

[3]Outputs $\overline{\text{OPT}}$<3:0> and $\overline{\text{RUN}}$. $\overline{\text{OPT}}$<3:0> capable of driving TTL or low current LED indicators.

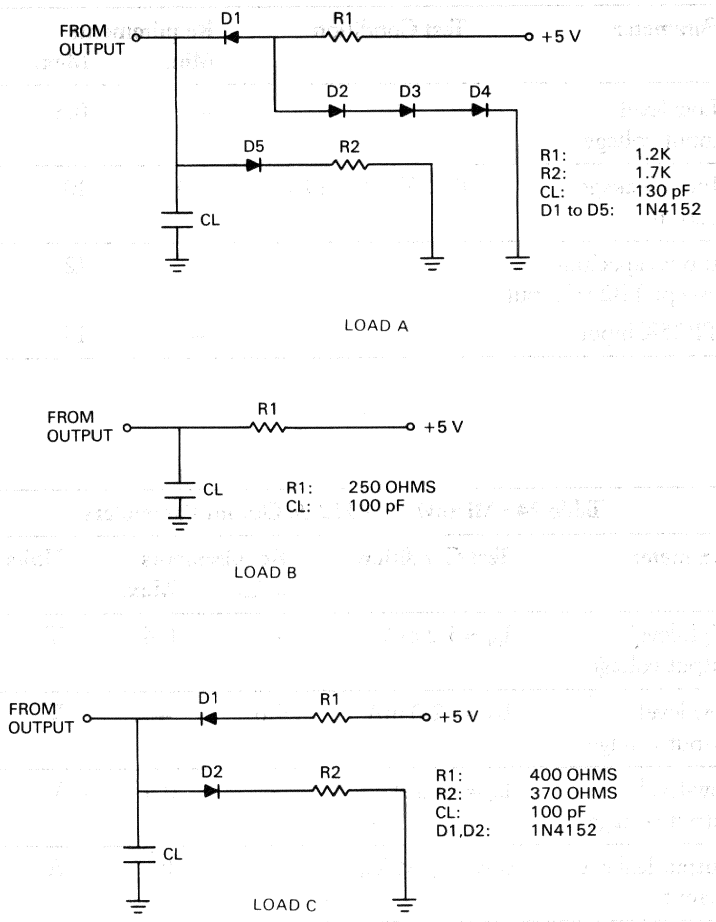


Figure 23 • MicroVAX 78332 dc Output Load Circuits

ac Characteristics

The clock input waveform and timing symbols are shown in Figure 24. Table 25 lists the clock input timing parameters.

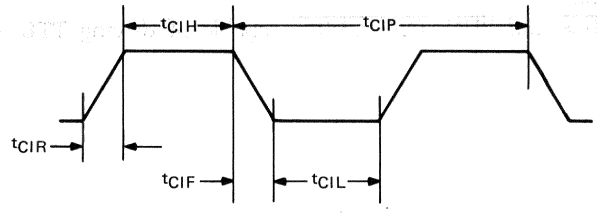


Figure 24 • MicroVAX 78332 Clock Input Waveform

Table 25 • MicroVAX 78332 Clock Input Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{CIR}	Clock In rise time	—	4.5
t_{CIF}	Clock In fall time	—	4.5
t_{CIH}	Clock In high	8.0	—
t_{CIL}	Clock In low	8.0	—
t_{CIP}	Clock In period	25	250

Figure 25 shows the timing and symbols for the SSC initialization and Table 26 lists the initialization parameters. The following specifications apply to the signals.

1. During cold-start powerup, the order in which V_{DDI} and V_{DDX} are powered up is unimportant, and the $\overline{BLO}$ input is ignored.
2. For total power down of V_{DDI} and V_{DDX} with no battery backup, the sequencing or transition times of V_{DDI} and V_{DDX} are not specified and the RESET input need not be asserted.
3. In any transition to normal operation, the RESET input should not be deasserted until the CLKI, TB40M, and all other input signals are within specification.
4. The deassertion of the RESET input initializes the SSC to its powerup state. The SSC should not be accessed until at least 1.0 microsecond after RESET is deasserted.
5. In battery backup mode, the high level of input signals TB25K, $\overline{BLO}$, and RESET must reach the value of V_{DDI}. All other input must have a low impedance to ground during battery backup and should be powered from V_{DDX} during normal operation.

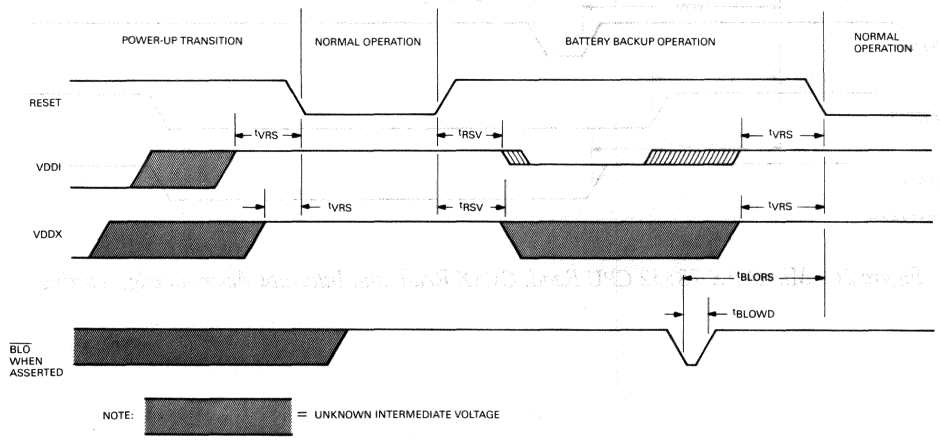


Figure 25 • MicroVAX 78332 Initialization Timing

Table 26 • MicroVAX 78332 Initialization Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{VRS}	V_{DDI} or V_{DDX} in normal operation range to RESET deasserted	0	—
t_{RSV}	RESET asserted to V_{DDI} or V_{DDX} out of normal operation range	50	—
t_{BLORS}	$\overline{BLO}$ asserted to RESET deasserted	50	—
t_{BLOWD}	$\overline{BLO}$ pulse width	25	—

The timing sequence for the CPU Read, CVAX EP Read, and Interrupt Acknowledge transactions are shown in Figure 26. Table 27 lists the timing parameters for the transactions shown. The specifications listed are relevant for transactions directed only at the SSC except for t_{ASH} and t_{ASL} parameters.

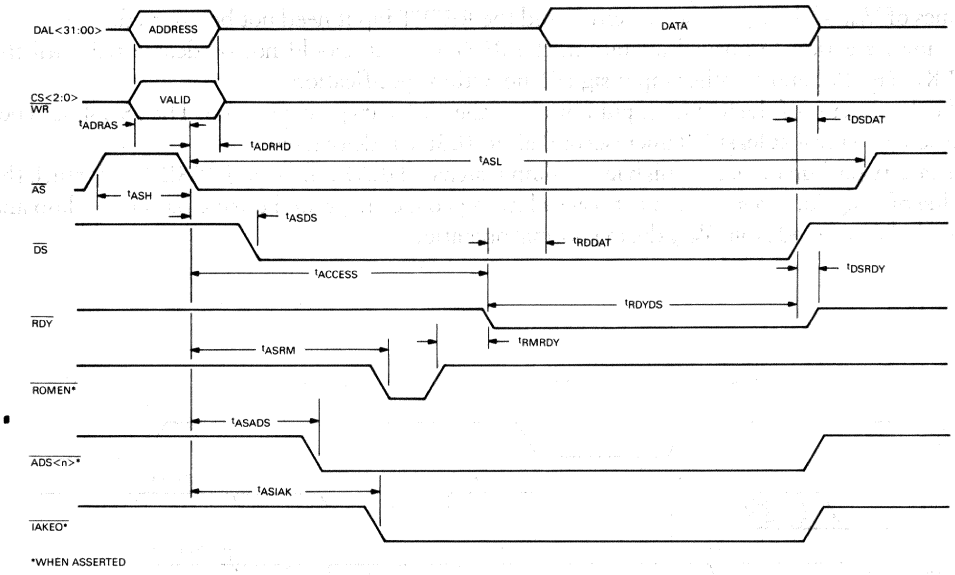


Figure 26 • MicroVAX 78332 CPU Read, CVAX Read, and Interrupt Acknowledge Timing

Table 27 • MicroVAX 78332 CPU Read, CVAX EP Read, and Interrupt Acknowledge Timing Parameters

Symbol[1]	Definition	Requirements (ns) Min.	Max.
t_{ACCESS}	$\overline{AS}$ asserted to $\overline{RDY}$ asserted		
	CSR access	150	250
	RAM access	250	350
	250-ns longwordwide ROM access	550	650
	250-ns wordwide ROM access	850	950
	250-ns bytewide ROM access	1450	1550
	350-ns longwordwide ROM access	650	750
	350-ns wordwide ROM access	1050	1150
	350-ns bytewide ROM access	1850	1950
t_{ADRAS}	Address setup before $\overline{AS}$ asserted	15	—
t_{ADRHD}	Address hold after $\overline{AS}$ asserted	10	—
t_{ASADS}	$\overline{AS}$ asserted to $\overline{ADS<n>}$ asserted	100	200
t_{ASDS}	$\overline{AS}$ asserted to $\overline{DS}$ asserted	25	—
t_{ASH}	Address strobe high time	45	—
t_{ASIAK}	$\overline{AS}$ asserted to $\overline{IAKEO}$ asserted	150	250
t_{ASL}	Address strobe low time	75	—
t_{ASRM}	$\overline{AS}$ asserted to $\overline{ROMEN}$ asserted	175	275
t_{DSDAT}	$\overline{DS}$ deasserted to DAL line high impedance	—	50
t_{DSRDY}	$\overline{DS}$ deasserted to $\overline{RDY}$ deasserted	—	50
t_{RDDAT}	$\overline{RDY}$ asserted to data valid	—	75
t_{RDYDS}	$\overline{RDY}$ asserted to $\overline{DS}$ deasserted	75	—
t_{RMRDY}	$\overline{ROMEN}$ deasserted to $\overline{RDY}$ asserted	75	—

[1]Except for t_{ASH} and t_{ASL} values, the above specifications are relevant only for transactions directed at the SSC.

Figure 27 shows the signal timing for the CPU Write and CVAX EP Write transactions. Table 28 lists the timing parameters.

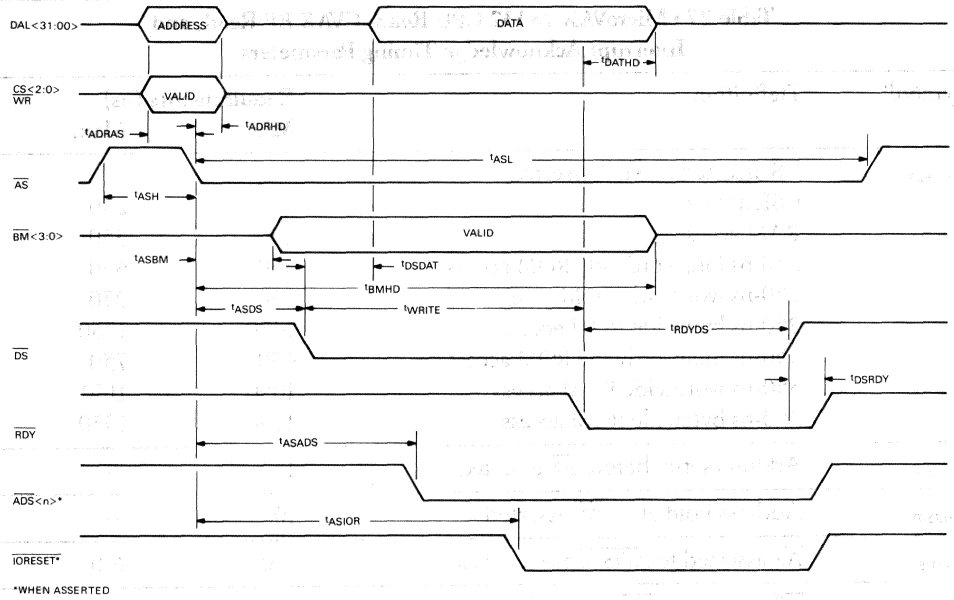


Figure 27 • MicroVAX 78332 CPU Write and CVAX EP Write Transaction Timing

Table 28 • MicroVAX 78332 CPU Write and CVAX EP Write Transaction Timing Parameters

Symbol[1]	Definition	Requirements (ns) Min.	Max.
t_{ADRAS}	Address setup before $\overline{AS}$ asserted	15	—
t_{ADRHD}	Address hold after $\overline{AS}$ asserted	10	—
t_{ASADS}	$\overline{AS}$ asserted to $\overline{ADS}<n>$ asserted	100	200
t_{ASBM}	$\overline{AS}$ asserted to byte mask valid	—	75
t_{ASDS}	$\overline{AS}$ asserted to $\overline{DS}$ asserted	0	—
t_{ASH}	Address strobe high time	45	—
t_{ASIOR}	$\overline{AS}$ asserted to $\overline{IORESET}$ asserted	175	275
t_{ASL}	Address strobe low time	75	—
t_{ASRDY}	$\overline{AS}$ deasserted to $\overline{RDY}$ high impedance	—	25
t_{BMHD}	Byte mask hold time after $\overline{AS}$	275	—
t_{DATHD}	Data hold time after $\overline{RDY}$ asserted	75	—
t_{DSDAT}	$\overline{DS}$ asserted to data valid	—	50
t_{DSRDY}	DS deasserted to $\overline{RDY}$ deasserted	—	50

Symbol[1]	Definition	Requirements (ns) Min.	Max.
t_{IORDY}	$\overline{IORESET}$ asserted to $\overline{RDY}$ asserted	775	800[2]
t_{RDYDS}	$\overline{RDY}$ asserted to $\overline{DS}$ deasserted	75	—
t_{WRITE}	$\overline{DS}$ asserted to $\overline{RDY}$ asserted	100	200[2]

[1]Except for t_{ASH} and t_{ASL} values, the above specifications are relevant only for transactions directed at the SSC.

[2]When $\overline{IORESET}$ is asserted, t_{IORDY} determines when $\overline{RDY}$ is asserted and t_{WRITE} is not used.

Figure 28 shows the the minimum transaction timing for the MicroVAX EP Write Command and Read Response transaction and Table 29 lists the timing parameters.

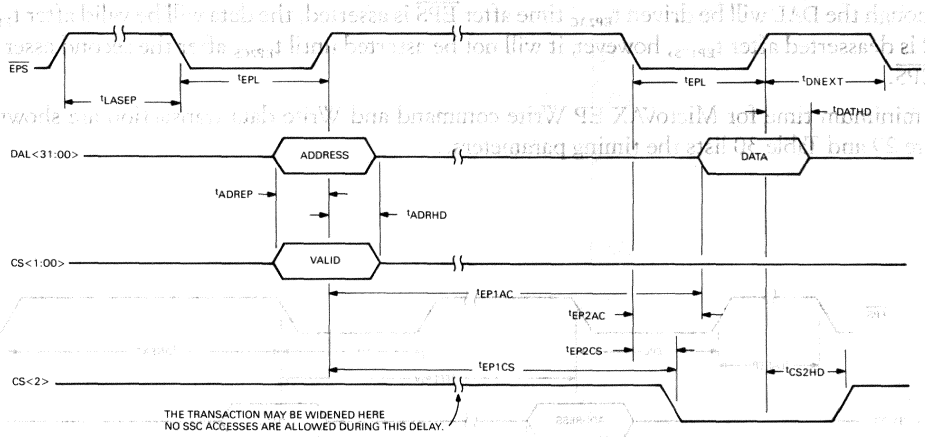


Figure 28 • MicroVAX 78332 MicroVAX EP Write and Read Response Transaction Timing

Table 29 • MicroVAX 78332 MicroVAX EP Write and Read Response Transaction Parameters

Symbol[1]	Definition	Requirements (ns) Min.	Max.
t_{ADREP}	Address setup time before $\overline{EPS}$ deassertion	15	—
t_{ADRHD}	Address hold time after $\overline{EPS}$ deassertion	10	—
t_{CS2HD}	CS2 hold time after $\overline{EPS}$ deassertion	—	25
t_{DATHD}	Data hold time after $\overline{EPS}$ deassertion	—	25

Symbol[1]	Definition	Requirements (ns) Min.	Max.
t_{DNEXT}	$\overline{EPS}$ deasserted to next assertion of $\overline{AS}$	75	—
	$\overline{EPS}$ deasserted to next assertion of $\overline{EPS}$	225	—
t_{EP1AC}	$\overline{EPS}$ deasserted to data valid	225	325
t_{EP1CS}	$\overline{EPS}$ deasserted to CS2 asserted	150	250
t_{EP2AC}	$\overline{EPS}$ asserted to data valid	—	50[2]
t_{EP2CS}	$\overline{EPS}$ asserted to CS2 asserted	—	25[3]
t_{EPL}	$\overline{EPS}$ assertion time	75	—
t_{LASEP}	Last deassertion of $\overline{EPS}$ to assertion of $\overline{EPS}$	225	—

[1]Except for t_{EPL}, t_{LASEP}, and t_{DNEXT}, these specifications are relevant only for transactions directed at the SSC.
[2]Although the DAL will be driven t_{EP2AC} time after $\overline{EPS}$ is asserted, the data will be valid after t_{EP1AC}.
[3]CS2 is deasserted after t_{EP1CS}, however, it will not be asserted until t_{EP2CS} after the second assertion of $\overline{EPS}$.

The minimum time for MicroVAX EP Write command and Write data transaction are shown in Figure 29 and Table 30 lists the timing parameters.

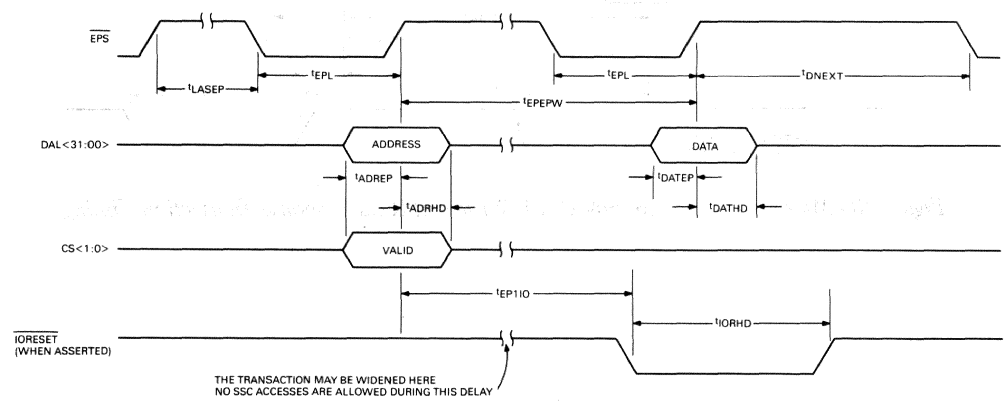


Figure 29 • MicroVAX 78332 MicroVAX EP Write Command and Write Data Transaction Timing

Table 30 • MicroVAX 78332 MicroVAX EP Write Command and Write Data Transaction Parameters

Symbol[1]	Definition	Requirements (ns) Min.	Max.
t_{ADREP}	Address setup time before $\overline{EPS}$ deassertion	15	—
t_{ADRHD}	Address hold time after $\overline{EPS}$ deassertion	10	—
t_{DATEP}	Data setup time before $\overline{EPS}$ deassertion	15	—
t_{DATHD}	Data hold time after $\overline{EPS}$ deassertion	15	—
t_{DNEXT}	$\overline{EPS}$ deassertion to next assertion of $\overline{AS}$	225	—
	$\overline{EPS}$ deassertion to next assertion of $\overline{EPS}$	225	—
t_{EP1IO}	$\overline{EPS}$ deassertion to $\overline{IORESET}$ assertion	175	275
t_{EPEPW}	Address $\overline{EPS}$ deassertion to data $\overline{EPS}$ deassertion	150	—
t_{EPL}	$\overline{EPS}$ assertion time	75	—
t_{IORHD}	$\overline{IORESET}$ assertion time	200	200
t_{LASEP}	Last deassertion of $\overline{EPS}$ to assertion of $\overline{EPS}$	225	—

[1]Except for t_{EPL}, t_{LASEP}, and t_{DNEXT}, these specifications are relevant only for transactions directed to the SSC.

Table 36 - MicroVAX 78332 MicroVAX I/P Wire Control and Wire Data Transaction Timing

Symbol	Parameter	Description (ns) Min.	Max.
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]

Preliminary [illegible] these specifications are subject to change without notice.

Section 2—Bus Support Devices

The bus support devices provide the interfaces for CVAX memory bus, VAXBI bus, and Q22-bus.

CVAX 78588 Memory Controller—The CVAX 78588 CMCTL is a high-performance dynamic memory controller for CVAX systems. It provides an interface between devices on the CVAX bus and the MOS private memory interconnect bus for memory arrays.

CVAX 78711 Q22-bus Interface Chip—The CVAX 78711 CQBIC provides an asynchronous interface between the CVAX CPU bus and the Q22-bus. It supports byte, word, and longword transfers and block mode DMA transfers.

DC514 CMOS VAXBI Bus Interface Chip—The DC514 CBIC is a high-performance interface used between the VAXBI bus and a user-developed interface of a node. It combines the functions of the the VAXBI 78742 BCAI and the VAXBI 78732 BIIC.

Features

- High-performance CMOS dynamic-memory controller for CVAX systems
- Two error checking modes: 7-bit ECC or single-bit parity
- Address multiplexing for 256 Kbit by 1 and 1 Mbit by 1
- RAM access time of 120 and 150 nanoseconds
- Synchronous or asynchronous interface to DMA devices
- Optimized write-through cache control
- Supports memory array diagnostics
- CPU interface compatible with CVAX Bus
- Integral refresh logic
- Single 5-volt power supply

Description

The CVAX 78588 Memory Controller (CMCTL) is contained in a 132-pin package and provides an interface between devices on the CVAX bus and an MOS private memory interconnect (PMI) bus to memory arrays. The CMCTL performs read or write operations initiated by the CVAX CPU in synchronous mode or initiated by external DMA devices connected to the CVAX bus in synchronous or asynchronous mode. The CMCTL controls from one to four memory arrays and one, two, or four banks of dynamic random access memory per array. The CMCTL allows mixed RAM sizes and provides error checking between arrays. Figure 1 is a functional block diagram of the CMCTL.

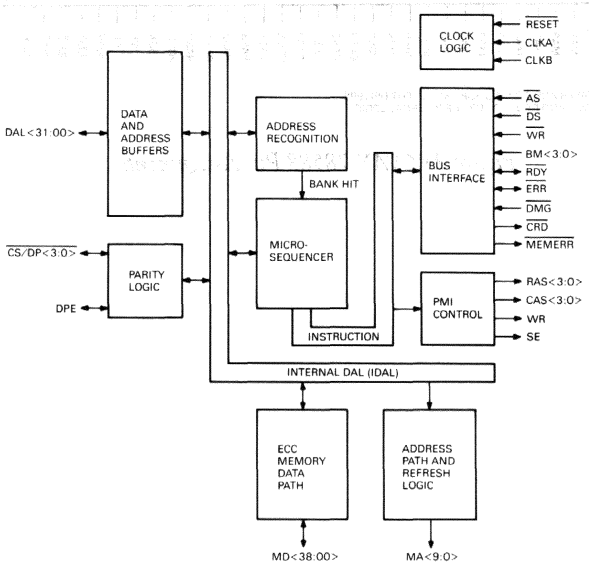


Figure 1 • CVAX 78588 Memory Controller Functional Block Diagram

Pin and Signal Description

The CMCTL operates with the I/O signals and power and ground connections shown in Figure 2. The signals are summarized in Table 1 and the signal functions are described in the following paragraphs.

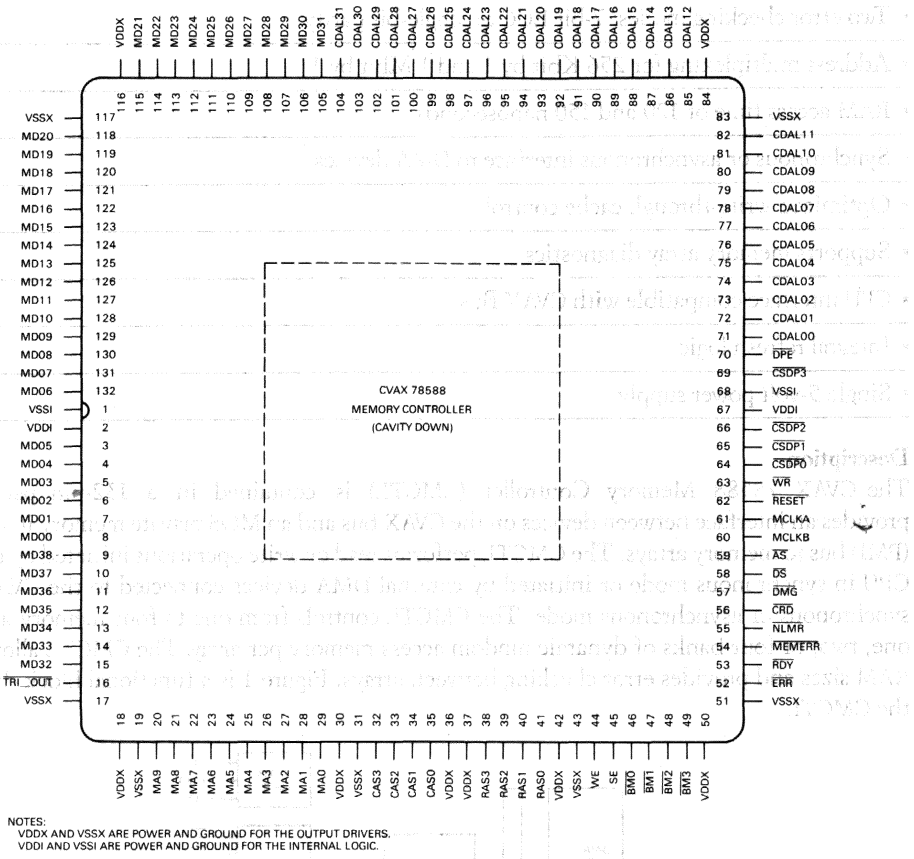


Figure 2 • CVAX 78588 Pin Assignments

Table 1 • CVAX 78588 Pin and Signal Summary

Pin	Signal	Input/Output	Definition/Function
59	$\overline{AS}$	input	Address strobe—Indicates that the CVAX bus contains valid control and address information.
50-47	$\overline{BM<3:0>}$	input	Byte masks—Specify which bytes on CDAL<31:00> contain valid information during memory write operations.
32-35	$\overline{CAS<3:0>}$	output	Column address strobe—Asserted during memory operations to indicate that MA<9:0> lines contain valid column address information. In fast diagnostic mode, the CAS<3:0> lines are asserted simultaneously. In normal diagnostic mode or signature read, only one of CAS<3:0> is asserted. The CAS<3:0> lines are deasserted during refresh mode.
56	$\overline{CRD}$	output	Corrected read data—During memory read operations, this line indicates that the data on CDAL<31:00> is correct. During masked memory write operation, it indicates that the CDAL<31:00> contains correctable memory data in the read part of the operation and no parity error.
69,66-64	$\overline{CS/DP<3:0>}$	input/output	Control status/data parity—Transfers cycle status and data parity information.
104-85,	CDAL<31:00>	input/output	CVAX data and address—Transfers 82-71 address and data information between the CMCTL, CVAX CPU, and external DMA devices.
57	$\overline{DMG}$	input	DMA grant—Asserted to indicate that a DMA operation is in process. When deasserted, it indicates that the operation is initiated by the CVAX CPU.
70	$\overline{DPE}$	input/output	Data parity enable—Enables parity checking and indicates that the $\overline{CS/DP<3:0>}$ contain valid parity information.
58	$\overline{DS}$	input	Data strobe—Asserted during read operations to indicate that the CMCTL can transfer information on CDAL<31:00> and during write operations to indicate that CDAL<31:00> contains valid data.
52	$\overline{ERR}$	input/output	Error—Indicates abnormal termination of the current bus cycle. The $\overline{ERR}$ and $\overline{RDY}$ inputs may be simultaneosly asserted to request a retry of the bus cycle.
20-29	MA<9:0>	output	Memory address—Time-multiplexed output specifying a row, column, or memory refresh address.

Pin	Signal	Input/Output	Definition/Function
61	MCLKA	input	Clock A—Provides the time base to the CMCTL and is 180 degrees out of phase with the MCLKB input.
60	MCLKB	input	Clock B—Provides the time base to the CMCTL and is 180 degrees out of phase with the MCLKA input.
9-15, 105-115, 118-132, 3-8	MD<38:00>	input/output	Memory data—Provides memory data between the CMCTL and the memory arrays.
54	$\overline{\text{MEMERR}}$	output	Memory error—Asserted as an interrupt to the CPU when a parity error is detected on CDAL<31:00> during a CVAX CPU single-transfer unmasked memory write operation.
55	$\overline{\text{NLMR}}$	output	Nonlocal memory reference—Asserted to indicate that the memory, loaded from CDAL<31:00>, is not within the range of the CMCTL.
38-41	RAS<3:0>	output	Row address strobe—Indicates which MA<9:0> lines have a valid row address and which MD lines have a valid command for a memory read or write operation.
53	$\overline{\text{RDY}}$	input/output	Ready—Asserted to indicate normal termination of a current bus cycle. The $\overline{\text{RDY}}$ and $\overline{\text{ERR}}$ inputs may be simultaneously asserted to request a retry of the bus cycle.
62	$\overline{\text{RESET}}$	input	Reset—Asserted to initialize the CMCTL.
45	SE	output	Signature enable—Asserted with the RAS<3:0> information to indicate a refresh operation and with the CAS<3:0> information to indicate that lines MD<4:0> contain signature read information from the CMCTL.
16	$\overline{\text{TRI OUT}}$	input	Three-state outputs—Asserted to indicate that all outputs are high impedance.
44	$\overline{\text{WE}}$	output	Write enable—Asserted with the CAS<3:0> information to indicate that the MD<38:00> information is valid. If deasserted with the CAS<3:0> information, the MD<38:00> information is CMCTL inputs.
63	$\overline{\text{WR}}$	input	Write—Asserted to specify a read operation and deasserted for a write operation.
2,18,30, 36,37,42, 50,67,84, 116,1,17,	V_{DD}	input	Voltage—Power supply voltage
19,31,43, 68,83,117	V_{SS}	input	Ground—Common ground reference.

Data and Address Lines

CVAX Data and address (CDAL<31:00>—Bidirectional time-multiplexed lines used to transfer addresses and data between the CMCTL, CVAX CPU, and external DMA devices.

During the first part of a read or write cycle, these lines provide address and control information to the CMCTL. The information transferred during a memory read or write transaction is listed in Table 2.

Table 2 • CVAX 78588 Read or Write CDAL Information

CDAL 31	30	Length	CDAL 29	Type	CDAL <28:02>	CDAL <01:00>
L	L	hexword	L	memory space	longword address	ignored by
L	H	longword	H	I/O space	for transfer	CMCTL
H	L	quadword				
H	H	octaword				

During the second part of a memory or control status register (CSR) write operation, the CDAL<31:00> provide information to the CMCTL. During the second part of a memory or CSR read operation, the CDAL<31:00> transfer information from the CMCTL. During memory read operations, the data on the MD<31:00> lines are transferred to the CDAL<31:00>. During a memory write operation, the data on CDAL<31:00> is transferred to the MD<31:00> lines.

Address strobe ($\overline{AS}$)—This input is asserted by the external logic to indicate that the CVAX bus contains valid control and address information. When asserted, the CDAL<31:00> and control signals $\overline{BM}$<3:0>, $\overline{CS/DP}$<3:0>, and $\overline{WR}$ are evaluated. At the conclusion of the bus cycle, the external logic deasserts $\overline{AS}$.

Byte masks ($\overline{BM}$<3:0>)—During memory write operations, these inputs specify which bytes on CDAL<31:00> contain valid information as shown in Table 3. The byte masks are not used by the CMCTL during configuration register (CSR) read or write operations or during memory read operations.

Table 3 • CVAX 78588 Byte Mask Assignments

$\overline{BM}$<3:0>*				Valid Bytes
L	L	L	L	CDAL<31:00>
H	H	H	L	CDAL<07:00>
H	H	L	H	CDAL<15:08>
H	L	H	H	CDAL<23:16>
L	H	H	H	CDAL<31:24>
H	H	H	H	read but no write

*All other binary combinations that specify the validity of two or three bytes on CDAL<31:00> are allowed.

When $\overline{AS}$ is asserted, the CMCTL evaluates the $\overline{BM<3:0>}$ information for a memory write operation. If the operation is a multiple transfer and the first transfer completes successfully, then following each assertion of $\overline{DS}$, the $\overline{BM<3:0>}$ information is evaluated on each data transfer to determine unmasked and masked memory write operations. If the $\overline{BM<3:0>}$ lines are all asserted, an unmasked memory write is performed by the CMCTL. Otherwise, a masked memory write is performed. The CMCTL ignores the $\overline{BM<3:0>}$ information during memory read and CSR read or write operations. A masked memory write occurs on a byte or word operation.

Corrected read data ($\overline{CRD}$)—This output is asserted during memory read operations and masked memory write operations to indicate that the CMCTL data has been corrected. During memory read operations, if the CDAL<31:00> contains corrected data from the CMCTL, both the $\overline{CRD}$ and $\overline{RDY}$ inputs are asserted. During masked memory write operations, a memory read is performed to detect and correct single-bit errors before the masked write to memory occurs. If a correctable error occurs during a memory read portion and no parity errors were detected on CDAL<31:00>, the CMCTL asserts both the $\overline{CRD}$ and $\overline{RDY}$ outputs.

Control status/data parity ($\overline{CS/DP<3:0>}$)—These bidirectional, time-multiplexed lines transfer control, status, and parity information. In the first part of an I/O cycle the $\overline{CS/DP3}$ input is asserted by a DMA device to request a synchronous operation. This input has an internal pullup resistor to accommodate asynchronous DMA devices that do not drive the $\overline{CS/DP3}$ line.

The $\overline{CS/DP<2:0>}$ inputs and the WR signal provide control information about the current bus cycle when the $\overline{AS}$ input is asserted as defined in Table 4.

Table 4 • CVAX 78588 Bus Cycle Selection

$\overline{WR}$	$\overline{CS/DP}$ line 2	1	0	Bus cycle	CMCTL Function*
H	L	L	L	request D-stream read	read
H	L	L	H	reserved	NOP (no operation)
H	L	H	L	external IPR read	NOP (no operation)
H	L	H	H	interrupt acknowledge	NOP (no operation)
H	H	L	L	request I-stream read	read
H	L	H	H	demand D-stream read (lock)	read memory (lock) or read CSR (no lock)
H	H	H	L	demand D-stream read (modify intent)	read
H	H	H	H	demand D-stream read (no lock or modify intent)	read
L	L	L	L	reserved	NOP (no operation)
L	L	L	H	reserved	NOP (no operation)
L	L	H	L	external IPR write	NOP (no operation)
L	L	H	H	reserved for DMA device use	NOP (no operation)

$\overline{WR}$	$\overline{CS/DP}$ line 2	1	0	Bus cycle	CMCTL Function*
L	H	L	L	reserved	NOP (no operation)
L	L	H	H	write unlock	write memory (unlock) or write CSR (no unlock)
L	H	H	L	reserved	NOP (no operation)
L	H	H	H	write no unlock	write

*The read and write operations are executed only if the address on CDAL<31:00> is within the programmed range of the CMCTL. If the address is not in the range, a no operation (NOP) occurs.

During the second part of an I/O cycle, the $\overline{CS/DP}$<3:0> outputs provide byte parity for data on CDAL<31:00> during a memory or CSR read/write. Even parity is checked or generated for even bytes and odd parity is checked or generated for odd bytes.

During a write operation, the $\overline{CS/DP}$<3:0> provides input information. If the $\overline{DPE}$ input is asserted, the CMCTL tests the CDAL<31:00> for parity errors. The $\overline{CS/DP}$<3:0> information must have valid parity for all bytes on CDAL<31:00> regardless of the $\overline{BM}$<3:0> inputs. If parity errors are detected during a memory write operation, the data and incorrect check bits are written to memory. A CSR write operation with parity errors is aborted.

During a read operation, the $\overline{CS/DP}$<3:0> outputs contain parity information generated by the CMCTL for all bytes regardless of the the $\overline{BM}$<3:0> inputs. The parity assignments are listed in Table 5.

Table 5 • CVAX 78588 Read Operation Parity Assignments

Parity bit	Byte
$\overline{CS/DP3}$	CDAL<31:24>
$\overline{CS/DP2}$	CDAL<23:16>
$\overline{CS/DP1}$	CDAL<15:08>
$\overline{CS/DP0}$	CDAL<07:00>

DMA Grant ($\overline{DMG}$)—This input is asserted by external logic to signify a DMA operation. It is deasserted to specify an operation that was initiated by the CVAX CPU.

Data Parity Enable ($\overline{DPE}$)—This bidirectional signal is used to control the checking or generation of data parity when the $\overline{DS}$ input is asserted. During a memory read operation, the CMCTL asserts $\overline{DPE}$ if the $\overline{CS/DP}$<3:0> lines contain valid parity information. During a write cycle, the $\overline{DPE}$ input enables parity checking on the incoming data. If not asserted during a write cycle, the CMCTL ignores the $\overline{CS/DP}$<3:0> information. This is an open-drain output and must be connected to V_{DD} through an external resistor to maintain high level when the outputs are a high impedance.

Data Strobe ($\overline{DS}$)—This input provides timing information for data transfers. During a memory or CSR read operation, it is asserted by the CVAX CPU or external logic to allow the CMCTL to transfer data to the CVAX bus. When an asynchronous operation is specified by the $\overline{CS/DP}$<3:0> information, the CMCTL stalls a read operation until the $\overline{DS}$ input is asserted. When external logic

receives and latches the data, it deasserts $\overline{DS}$. When deasserted at the end of a CMCTL operation, it causes the $\overline{CS/DP<3:0>}$, CDAL<31:00>, and $\overline{DPE}$ lines to become a high impedance and deasserts the $\overline{ERR}$ and $\overline{RDY}$ signals.

Error ($\overline{ERR}$)—This signal is used by the external logic and the CMCTL to indicate an error condition and is asserted during memory read operations until the CMCTL transfers data to the CVAX bus. If it is asserted by external logic during memory read operations, the CMCTL terminates the operation and does not transfer data. The CMCTL asserts $\overline{ERR}$ to indicate that a CDAL<31:00> parity error has occurred during a DMA write operation or that an uncorrectable error has been detected during a memory read or on the read portion of a masked memory write operation. Memory parity errors are considered uncorrectable. During a retry on a read lock request, the $\overline{RDY}$ signal is asserted.

The $\overline{ERR}$ signal is asserted by the CMCTL when a read-lock request occurs and the lock-bit test results in a hit. Both the $\overline{ERR}$ and $\overline{RDY}$ signals are asserted for one sampling window to indicate a retry. For asynchronous DMA read operations, the $\overline{ERR}$ input is asserted first and the $\overline{RDY}$ input is asserted after phase.

When a single-transfer unmasked write operation (dump and run) is initiated by the CVAX CPU in which a parity error is detected, the $\overline{ERR}$ signal is not asserted by the CMCTL. Refer to the $\overline{MEMERR}$ signal description for additional information.

When the $\overline{DS}$ input is deasserted, the $\overline{ERR}$ signal is also deasserted. When the $\overline{AS}$ input is deasserted, $\overline{ERR}$ is a high impedance. This output requires an external pullup resistor connected V_{DD} to maintain high level when the outputs are a high impedance.

Memory Address (MA<9:0>)—These time-multiplexed lines provide row address, column address, or memory refresh address to the memory array.

Master Clock A (MCLKA)—A clock input that provides the timebase for the CMCTL. It is phase-shifted by 180 degrees from the MCLKB input.

Master Clock B (MCLKB)—A clock input that provides the timebase for the CMCTL. It is phase-shifted by 180 degrees from the MCLKA input.

Memory Data (MD<38:00>)—Bidirectional lines that provide memory data between the CMCTL and one of four external memory arrays. When the CMCTL is in ECC mode, the MD<38:32> lines contain the seven ECC check bits. In parity mode, the MD32 line contains odd parity, the MD<38:33> lines are ignored during memory read operations, and the MD<38:33> lines contain zeros during memory write operations. If an error detection mode is not enabled (controlled by the CSR in the memory), then the MD<38:32> information is ignored during memory read operations. The MD<38:32> lines contain zeros during memory write operations.

When the RAS<3:0> outputs are valid, each MD30, MD20, MD10, and MD00 line contains a valid command for the external logic on the memory array. The MD lines indicate the logic value of the fast diagnostic test bit 9 of control and status register (CSR17). If each of these MD lines is a zero, subsequent memory read and write operations occur normally. If each line is a 1, subsequent memory read and write operations occur in fast diagnostic test mode, all RAS<3:0> strobes are asserted and deasserted simultaneously, and all CAS<3:0> strobes are simultaneously asserted and deasserted.

During a signature read operation, the MD<04:00> lines are asserted when not controlled by a memory array. When the SE output is asserted with the assertion of one of the CAS<3:0> lines, signature information is transferred on the MD<04:00> lines by each memory array as defined in Table 6.

Table 6 • CVAX 78588 Memory Data Signature Information

MD 04	MD 03	Memory check bits
H	H	zero
H	L	one
L	H	seven
L	L	reserved

MD 01	MD 00	Memory banks	MD02	RAM size
H	H	none	H	256 Kb
H	L	one	L	1 Kb
L	H	two		
L	L	four		

Memory Error ($\overline{\text{MEMERR}}$)—This output interrupts single-transfer unmasked memory write operations initiated by the CVAX CPU in which a parity error has been detected on the incoming data. In this operating mode, the $\overline{\text{RDY}}$ input is asserted before the input data parity is checked. This output provides a means to report late errors. It is an open-drain output and must connected to V_{DD} through an external resistor.

Nonlocal Memory Reference ($\overline{\text{NLMR}}$)—This output is asserted by the CMCTL to indicate to the external logic that the memory or CSR address from the CDAL<31:00> is not a CMCTL address. It is deasserted when the $\overline{\text{AS}}$ input is deasserted.

Row Address Strobe (RAS<3:0>)—The CMCTL asserts one of the these lines during a memory read or write operation if the address on the CDAL<31:00> is within the programmed range of the CMCTL and if a refresh request is not pending. At a low-to-high transition, the RAS<3:0> information indicates that the MA<9:0> lines contain a valid row address and the MD30, MD20, MD10, and MD00 lines have a valid command for logic on the memory array. The RAS<3:0> lines are asserted simultaneously during a memory refresh or fast diagnostic test mode operation.

Column Address Strobe (CAS<3:0>)—These outputs are asserted by the CMCTL during memory read or write operations to indicate that a valid column address is on the MA<9:0> lines if the CDAL<31:00> lines contain a valid CMCTL address and if a refresh request is not pending. During a signature read operation, one of the CAS<3:0> lines is asserted, depending on which signature read request bit is set in the CSR register that is accessed. During a refresh operation, the CAS<3:0> lines are deasserted.

Ready ($\overline{\text{RDY}}$)—This signal indicates when bus operations can occur. As an input, it prevents the CMCTL from completing a read operation. As an output, it indicates that a read operation with valid parity has been completed or that parity has been checked.

During memory read operations, $\overline{\text{RDY}}$ is an input until the CMCTL transfers data to the CVAX bus. When asserted by the external logic, the CMCTL terminates the read operation and does not transfer data to the CVAX bus. The $\overline{\text{RDY}}$ signal is asserted by the CMCTL to indicate that the $\overline{\text{CS}}$/$\overline{\text{DP3:0}}$> lines, CDAL<31:00>, and $\overline{\text{DPE}}$ line contain valid data. During single-transfer unmasked memory write operations initiated by the CVAX CPU, the $\overline{\text{RDY}}$ signal is asserted when the CMCTL has latched the data from CDAL<31:00>. For all other write operations, the CMCTL asserts the $\overline{\text{RDY}}$ signal after the CDAL<31:00> data is latched and valid parity has been detected.

The $\overline{RDY}$ signal may be asserted during the sampling window, except for a retry of asynchronous transfers when it is asserted on phase 1 (P1) of a clock cycle. It is deasserted when the $\overline{DS}$ input is deasserted and becomes a high impedance when the $\overline{AS}$ input is deasserted. The $\overline{RDY}$ input must be connected to V_{DD} through a resistor.

Reset ($\overline{RESET}$)—When this input is asserted, the CAS<3:0>, $\overline{CRD}$, $\overline{NLMR}$, RAS<3:0>, SE, and WE lines are deasserted. The $\overline{CS/DP}$<3:0>, CDAL<31:00>, $\overline{DPE}$, $\overline{ERR}$, $\overline{MEMERR}$, and $\overline{RDY}$ lines are set to a high impedance. The MA<9:0> and MD<38:00> lines, the control and status registers, the refresh counter, and the refresh request counter are cleared. The lock bit is set to the unlocked condition. When $\overline{RESET}$ is deasserted, the CMCTL is synchronized with the first low-to-high transition of the MCLKA input at the start of phase (P1) and the refresh timeout counter begins counting.

Signature Enable (SE)—This output is asserted with the CAS<3:0> outputs to request signature information and to indicate that the MD<04:00> lines contain input information. When the SE output is asserted, the RAS<3:0> and WE outputs are deasserted. The SE output and RAS<3:0> outputs are asserted when the CMCTL is performing a memory-refresh operation.

Three-state Outputs ($\overline{TRI\ OUT}$)—When asserted, this input causes all outputs to become high impedance.

Write (WR)—This input is asserted to specify a write bus cycle and is deasserted to specify a read bus cycle.

Write Enable (WE)—This output is asserted to enable a memory read or write operations. If WE and CAS<3:0> are asserted during memory write cycles, the MD<38:00> information is valid. If deasserted when the CAS<3:0> lines are asserted, the MD<38:00> contains input information.

Voltage (V_{DD})—This is the 5-volt input from the power supply.

Ground (V_{SS})—This is the signal and power ground reference.

Registers

The CMCTL contains 18 control and status registers (CSR0 through CSR17). CSR0 through CSR15 are configuration registers, CSR16 is a system error status register, and CSR17 is a Mode Control and Diagnostic Status Register. Each registers must be longword accessed.

Configuration Registers (CSR0–CSR15)—A configuration register is assigned to each of the 16 banks of memory that can be connected to the CMCTL. CSR0 through CSR3 correspond to the four banks on array 0, CSR4 through CSR7 to the four banks on array 1, CSR8 through CSR11 correspond to the four banks on array 2, and CSR12 through CSR15 to the four possible banks on array 3. Figure 3 shows the read format and Figure 4 shows the write format of these registers. Table 7 describes the function of the register information.

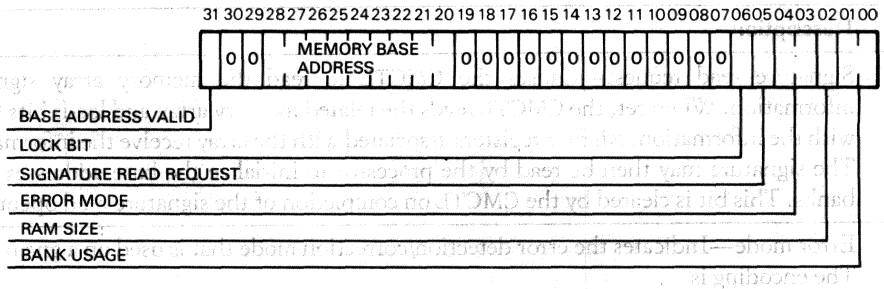


Figure 3 • CVAX 78588 Configuration Registers (CSR0–CSR15) Read Format

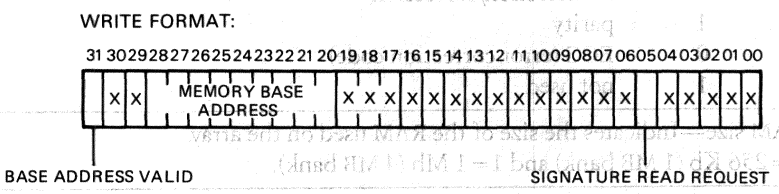


Figure 4 • CVAX 78588 Configuration Registers (CSR0–CSR15) Write Format

Some of these registers contain information used at the array level. The fields of the four registers related to an array will contain the same information. For example, a signature read request issued by the processor can be performed through any one of the four registers related to an array. All of the related registers receive the signature information. The CSR0 through CSR15 are cleared when the $\overline{\text{RESET}}$ signal is asserted.

Table 7 • CVAX 78588 Configuration Registers (CSR0–CSR15) Description

Bit	Description
31	Base address valid—Set when the base address is written to enable the addressing of the bank indicating that the base address CSR<28:20> is valid. This bit is cleared during powerup and may be used by diagnostics to disable a memory bank.
30:29	Not used—Read as zeros and a write has no effect.
28:20	Memory Base Address—Specifies the base address of the related memory bank. If the bank contains 256 KB RAMs, all nine bits are used in the address compare. If the RAM size is 1 MB, only bits 28:22 are used. All nine bits are read and written. Refer to the *Addressing* section for the use of the base address.
19:07	Not used—Read as zeros
06	Lock bit—Indicates the status of the lock bit for all 16 banks of RAM. This bit is cleared during powerup and unlocks the CMCTL. When set, the CMCTL is prevented from performing a read-lock request. Memory read operations without the lock qualifier are not affected by the status of this bit and the installation of this function is optional.

Bit	Description
05	Signature read request—Causes the CMCTL to read the memory array signature information. When set, the CMCTL reads the related memory array and loads bits 04:00 with the information. All four registers associated with the array receive the information. The signature may then be read by the processor to initialize the base addresses of all banks. This bit is cleared by the CMCTL on completion of the signature read operation.
04:03	Error mode—Indicates the error detection/correction mode that is used on a given array. The encoding is

Bit 04	Bit 03	Mode
0	0	no detection/correction
0	1	parity
1	0	ECC (error correction code)
1	1	not used

02	RAM size—Indicates the size of the RAM used on the array. 0 = 256 Kb (1 MB bank) and 1 = 1 Mb (4 MB bank).
01:00	Bank used—Indicates the number of banks used on an array as follows

Bit 01	Bit 00	Banks
0	0	array not present
0	1	one
1	0	two
1	1	four

System Error Status Register

The CMCTL stores error data in the system error status register (CSR16). The error status flags (bits 30:29) are cleared by writing a 1 to the bit. Once these bits are cleared, the state of the error status flags will not change. This register is cleared when the $\overline{\text{RESET}}$ input is asserted. The read format of this register is shown in Figure 5, and Figure 6 shows the write format. Table 8 describes the function of the register information.

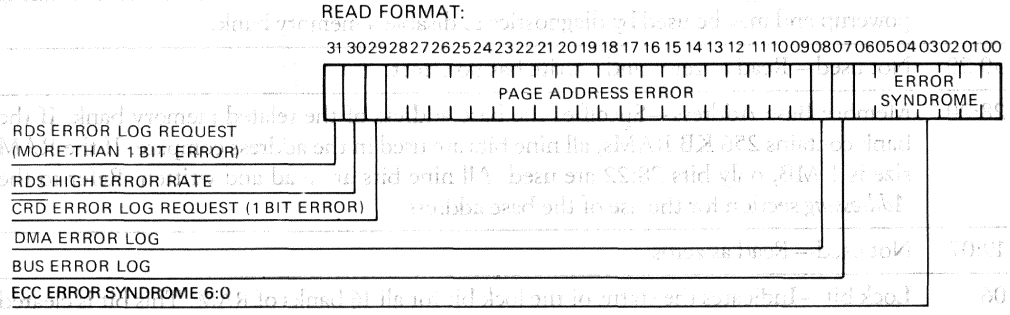


Figure 5 • CVAX 78588 System Error Status Register Read Format

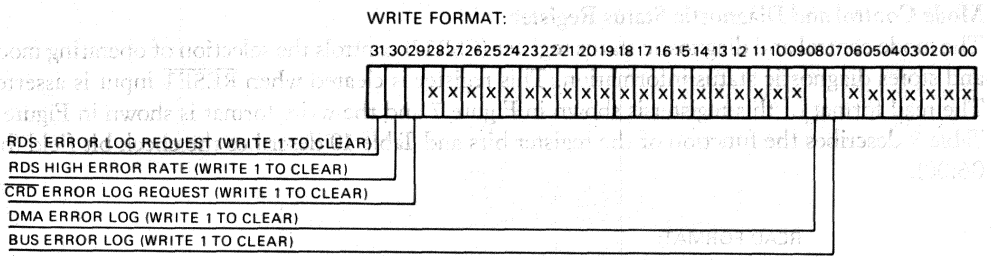


Figure 6 • CVAX 78588 System Error Status Register Write Format

Table 8 • CVAX 78588 System Error Status Register Description

Bit	Description
31	RDS error log—This bit is set when an uncorrectable ECC or parity error occurs during a memory read or masked write operation. It is cleared by writing a 1 to this bit.
30	RDS high error log—This bit is set when an uncorrectable ECC or parity error occurs while the RDS error log bit (bit 31) is set. It is cleared by writing a 1 to this bit.
29	CRD error log—This bit is set when a correctable (single bit) error occurs during a memory read or masked write operation. It is cleared by writing a 1 to this bit.
28:09	Page address of error—Identifies the page (512-byte block) where an error occurred during a memory operation. The logging of the error address is prioritized. If an address has already been logged by an error of equal or higher priority, the address is not overwritten. The error conditions that may cause the logging of the error address may occur during either a CVAX CPU-initiated transfer or a DMA operation. The error condition priority is 1. A bus parity error occurs during a write operation and is logged by the Bus Error Log bit 07 of the system error status register. 2. An uncorrectable error occurs during a memory read or masked write operation and is logged by the RDS Error Log Request bit 31. In parity mode, parity errors are considered uncorrectable. 3. A correctable error occurs during a memory read or masked write operation and is logged by the CRD error log bit 29.
08	DMA error log—Set when an error has occurred during a DMA operation and cleared by writing a 1 to this bit.
07	Bus error log—Set when a bus parity error has been detected during a write operation and cleared by writing a 1 to this bit.
06:00	Error syndrome—If a memory error is detected in ECC mode, this field stores the error syndrome which is loaded with the error address field when a memory error is detected. When parity mode is enabled and a memory error is detected, this read-only field will be read as zeros.

Mode Control and Diagnostic Status Register

The mode control and diagnostic status register (CSR17) controls the selection of operating modes and stores diagnostic status information. This register is cleared when $\overline{\text{RESET}}$ input is asserted. The read format of this register is shown in Figure 7 and the write format is shown in Figure 8. Table 9 describes the function of the register bits and Table 10 describes the check bit field (bits 06:00).

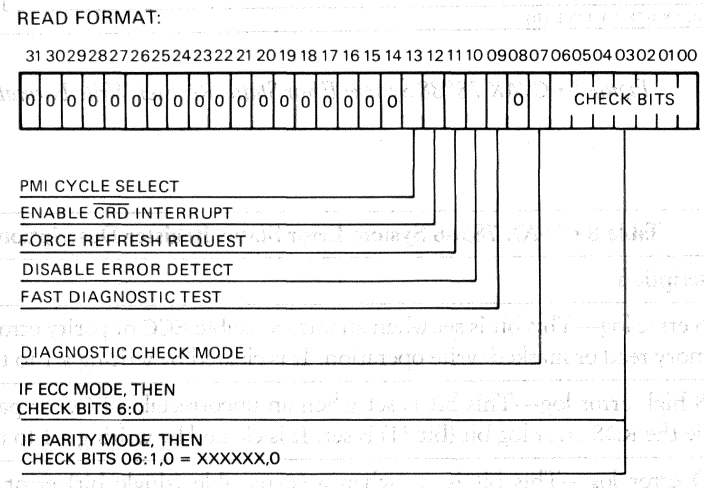


Figure 7 • CVAX 78588 Mode Control and Diagnostic Status Register Read Format

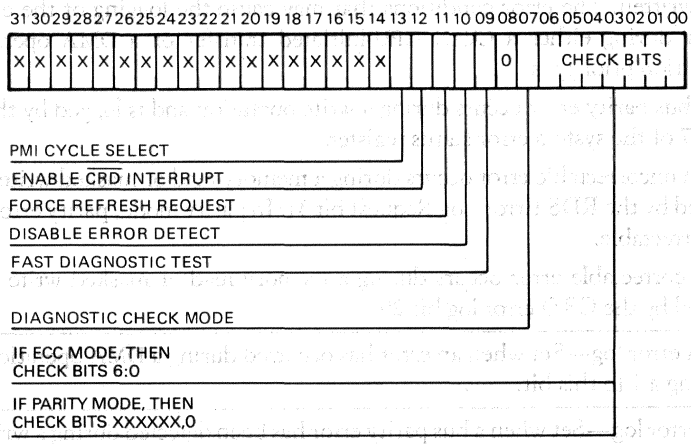


Figure 8 • CVAX 78588 Mode Control and Diagnostic Status Register Write Format

Table 9 • CVAX 78588 Mode Control and Diagnostic Status Register Description

Bit	Description
31:14	Not used and read as zeros
13	PMI cycle select—Set to select the private memory interconnect (PMI) cycle as an integral multiple of CVAX bus cycles. This feature maintains a fixed PMI cycle during the time that the speed of CVAX bus cycle increases from 100 to 80 nanoseconds. It allows the use of RAMs with faster access time. When this bit is cleared, a cycle is added to each memory read transfer, and the $\overline{RDY}$ output is slipped one cycle. When set, a cycle is removed from each memory read transfer. Refer to the *Operations* section for additional information. The relationship of the bus cycles and access time is **CVAX bus cycle** / **PMI bus cycle** / **RAM access time** 100 ns / 4/2 / 120 ns 100 ns / 5/3 / 150 ns 80 ns / 5/3 / 120 ns
12	Enable CRD—When set, the correctable (single-bit) CRD errors are corrected by the ECC logic. The $\overline{CRD}$ interrupt signal is not asserted to report the error. This bit does not affect the page address bits 28:09, the CRD error log bit 29 of CSR16, or the logging and reporting of uncorrectable errors. This bit is cleared when no error occurs or an uncorrectable error occurs.
11	Force refresh—This bit causes a memory refresh operation to be performed immediately following the CSR write transaction that sets this bit. When cleared, the refresh control logic operates in normal mode. Setting this bit allows the speed of the refresh logic to be increased during CMCTL manufacturing test. This bit is write only and is read as zero. It is cleared by the CMCTL after the forced refresh is completed.
10	Disable memory error detect—When set, the error detection (ECC and parity mode) and correction (ECC mode only) is disabled. The error logging in CSR16 is disabled and memory error reporting is inhibited on the $\overline{ERR}$ or $\overline{CRD}$ outputs. When cleared, the error detection and correction are enabled.
09	Fast diagnostic test—When set and bits 29:26 of the address are cleared, all the RAS<3:0> and CAS<3:0> strobes are asserted for a memory read or write operation. This bit and the CDAL<29:26> information are used to decrease the time of the initial diagnostic memory test.
08	Not used and read as zero.
07	Diagnostic check mode—Set to select the diagnostic check mode. When set, the content of the check bits 06:00 is transferred to lines MD<38:32> or check bit 00 is transferred to MD32 for a memory write operation in ECC or parity mode. When cleared, check bits 06:00 or 00 are loaded from the MD<38:32> or MD32 lines during a memory read operation in ECC or parity mode. Only unmasked memory write operations can be used in this mode.
06:00	Check bits—This field contains the check bit(s) resulting from or used with the error detection and correction operations. The relationship of this information depends on the modes selected and the type of operations as listed.

Bit	Description		
	Diagnostic check mode		
	ECC mode	**Parity mode**	**Write operation**
	enabled	disabled	bits 06:00 transfer to MD<28:32>
	disabled	enabled	bit 00 transfers to MD32
	disabled	disabled	000000 transfers to MD<38:33>
	disabled	disabled	0000000 transfers to MD<38:32>
	Nondiagnostic check mode		
	ECC mode	**Parity mode**	**Memory read operation**
	enabled	disabled	MD<28:32> transfer to bits 06:00
	disabled	enabled	MD32 transfers to bit 00
	disabled	disabled	000000 transfers to bits 06:01
	disabled	disabled	0000000 transfers to bits 06:00

Error Checking

The CMCTL can operate without error checking, in error checking and correction (ECC) mode, or in parity mode. In ECC mode, the CMCTL flags and corrects single-bit errors and flags double-bit errors. Single-bit errors are corrected on CDAL<31:00> but not corrected in memory. Error correction in memory is an optional software function. In parity mode, the CMCTL performs as in ECC mode except that memory parity errors are considered uncorrectable. If an uncorrectable error is detected during a transfer, the assertion of $\overline{\text{ERR}}$ signal terminates the memory read operation. As an example, if an uncorrectable error is detected in the first longword read from memory during an octaword transfer, the transfer is terminated by asserting $\overline{\text{ERR}}$. The remaining three longwords are not read from memory. In either ECC or parity mode, a correctable error is reported as an interrupt by the assertion of the $\overline{\text{CRD}}$ and $\overline{\text{RDY}}$ signals.

ECC mode data—During memory write operations, the MD31:00> contains the same information as CDAL<31:00> together with seven check bits generated from an ECC generator. During memory read operations, error checking and correction is generated from the MD31:00> inputs and compared to the MD<38:32> check bits. The ECC mode uses a 32-bit modified Hamming code to encode a 32-bit data longword with seven check bits. When an error is detected, the syndrome is loaded into bits 06:00 of the system error status register. The ECC logic detects and corrects single-bit errors in the MD<31:00> data field. Single-bit errors in the check bit field are detected and reported. Double-bit errors are detected but not corrected and are reported by asserting the $\overline{\text{ERR}}$ signal.

The modified Hamming code, shown in Table 10, generates seven check bits that are stored in memory. For a memory write operation, bits MD<31:00> that are exclusively OR (XOR) gated are indicated by an X in each row. From this value, parity that is even is generated on C1, C2, and CT and odd parity is generated on C4, C8, C16, and C32.

During a memory read operation, the data bits indicated with an X in each row are XOR gated again and check bits are again generated. These check bits are XOR gated with the check bits stored in memory. If the 7-bit result is zero, no errors were generated. If the result is not zero, one or more errors in the data has been detected or an error in a check bit has been detected. The ECC error syndrome is the result of an error and is stored in CSR16 bits 06:00. If the syndrome matches any column of bits that contain an X in MD<32:38>, the error is correctable and the column number corresponds to the bit that is corrected. For example, if the memory check bits are 0111100 and an error is indicated in MD00, the check bits generated on a memory write would be 1100100.

Therefore, the 0111100 value is XOR gated with 1100100 to equal 1011000. This result corresponds to syndrome bits ST, S32, S16, S08, S04, S02, and S01 and can be read under column MD00 by reading the bits that contain an X as a one and the bits without an X as zero. Any syndrome value that does not match the value in Table 10 indicates an uncorrectable error. Table 11 lists sample syndromes that can be read from bits 06:00.

Table 10 • CVAX 78588 32-Bit Modified Hamming Code for ECC

	MD<31:00>																								MD<38:32> Generated Check Bits						
Syn. Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	CTC 38	32	16	8	4	2	1
S1	X	X	X	X					X	X	X	X					X	X	X	X					X	X	X	X											X
S2			X	X	X	X					X	X	X	X					X	X	X	X					X	X	X	X						X	
S4		X		X	X		X			X		X	X		X			X		X	X		X				X		X		X				X		
S8	X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X				X			
S16	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X			X				
S32	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X										X					
S34	X			X		X	X			X	X		X			X		X	X		X			X		X	X		X			X	X					

Table 11 • CVAX 78588 CSR16 Syndrome Readable Bits

CSR bit 06:00	Single-bit errors
1011000	MD00
1111001	MD31
0000001	MD32
1000000	MD38
0000011	uncorrectable

During a memory write operation when a data parity error is detected on CDAL<31:00>, incorrect check bits are generated and transferred to the MD<38:32> outputs for detection on a subsequent memory read operation. The algorithm that generates the incorrect check bits complements the generated check bits for the MD<34:32> outputs and transfers the generated check bits unchanged to MD<38:35> outputs.

Parity mode data—During a memory read operation, the CMCTL generates odd parity from the MD<31:00> inputs and compares the generated value to MD32. If the parity bit comparison does not match, an error is detected. During a memory write operation, odd parity is written to memory by the MD32 check bit. When a data parity error is detected on CDAL<31:00>, the force incorrect check bit logic complements the generated odd parity before it is transferred to the MD32 output.

Addressing Scheme

The CMCTL can control up to 16 banks of 256 Kb or 1 Mb RAM. Each location can consist of 32 bits of data and 0, 1, or 7 bits of error detection or correction information. Each bank has a 32-bit base address that resides in the CMCTL. Seven or 9 bits of this address are significant. If the RAM size is 256 Kb (bit 02 of CSR0 through CSR15 is zero), the base address is mapped to CDAL<28:20>. If the RAM size is 1 Mb (bit 2 is 1) the base address is mapped to CDAL<28:22>. Refer to the *Register* section for reading and writing the base addresses. When a

base address matches the address on CDAL<31:00>, the bank related to that address is activated for either a read or write operation. The CDAL29 is always zero indicating a memory address space.

RAM access is provided by the MA<9:0>, RAS<3:0>, and CAS<3:0> outputs. The RAS<3:0> and CAS<3:0> enable one of the banks. The MA<9:0> outputs (MA9 is not used for 256 Kb RAMs) provide the memory address within a bank as described.

MA<9:0> addresses—The source of the MA<9:0> output information depend on the type of memory operation. The MA<9:0> output contains a row address during the low-to-high transition of the RAS<3:0> output and a column address during a low-to-high transition of the CAS<3:0> outputs.

During a single transfer memory operations

Row	CDAL20 transfers to MA9 CDAL<19:11> transfers MA<8:0>
Column	CDAL20 transfers to MA9 CDAL<10:02> transfers to MA<8:0>

During multiple transfer memory operations, the MA<9:0>, RAS<3:0>, and CAS<3:0> output information is in page mode format. The CMCTL compares the address on CDAL<31:00> with the CSR address. One of the RAS<3:0> lines is then asserted followed by the assertion of one of the CAS<3:0> lines. After completing the first operation, the RAS<3:0> line is held asserted while the CAS<3:0> line is continually negated and reasserted for the remainder of the operation. Because the RAS<3:0> line is not negated and reasserted, the time required to perform the operation is minimized. MA9 is not used when 256 Kb RAMs are present. The address information on MA<9:0> is

Row	CDAL21 transfers to MA9 CDA<19:11> transfers to MA<8:0>
Column(0)	CDAL20 transfers to MA9 CDAL<10:02> transfers to MA<8:0>
Column(n)*	CDAL20 transfers to MA9 CDAL<10:04> transfers to MA<8:2> LW CTR 1:0 transfers to MA<1:0>

*n=1 (quadword)
n=2 (hexword)
n=3 (octaword)

The longword counter (LW CTR) is initialized with the address value on CDAL<01:00> for the first transfer. For a quadword transfer, LW CTR bit 0 is then complemented. If the transfer is hexword or octaword, the counter is incremented two or three times, respectively.

If an internal memory refresh request is pending while the CMCTL is in the idle state, the CMCTL performs only a RAS memory refresh. If the $\overline{AS}$ input is asserted, the address on CDAL<31:00> is latched and the requested operation is stalled until the memory refresh completes. All the RAS<3:0> outputs on the memory bus are asserted and all CAS<3:0> outputs are deasserted. The source of the MA<9:0> refresh address is a 9-bit (0,8:0) refresh counter. The counter is cleared by the $\overline{RESET}$ input and incremented after the refresh operation has been completed. The $\overline{RESET}$ input also clears the refresh timer that provide the refresh request. Table 12 specifies the number of clock cycles (MCLKA or MCLKB) between each refresh request.

Table 12 • CVAX 78588 Refresh Request Timing

PMI cycle select[1]	Clock periods(ns)	Clock cycles	Overhead (percent)	Refresh rate (μs)[2]
1	55	226	2.7	12.43
1	50	226	2.7	11.3
0	40	228	3.5	9.12
0	50	228	3.5	11.4

[1]Bit 13 of the mode control and status register (CSR17)
[2]Includes 4.0 microseconds for the completion of an asynchronous DMA octaword write operation and for a slower clock cycle.

RAS<3:0> and CAS<3:0> addresses—If a base address of a memory bank matches the address on CDAL<28:02>, one of the RAS<3:0> outputs and one of the CAS<3:0> outputs is asserted. Table 13 summarizes the bank addressing on each array as a function of the asserted RAS<3:0> and CAS<3:0> signals.

Table 13 • CVAX 78588 Bank Row and Column Address Selection

Bank	Strobe	Bank	Strobe
0	RAS0 and CAS0	8	RAS0 and CAS2
1	RAS1 and CAS0	9	RAS1 and CAS2
2	RAS2 and CAS0	10	RAS2 and CAS2
3	RAS3 and CAS0	11	RAS3 and CAS2
4	RAS0 and CAS1	12	RAS0 and CAS3
5	RAS1 and CAS1	13	RAS1 and CAS3
6	RAS2 and CAS1	14	RAS2 and CAS3
7	RAS3 and CAS1	15	RAS3 and CAS3

Register addressing—Each register (CSR0 through CSR17) has a fixed address in the I/O space of the CVAX CPU. If a register's address matches the address on CDAL<31:00>, a read or write operation to or from one of the registers occurs. The register addresses are listed in Table 14.

Table 14 • CVAX 78588 Control and Status Register Addresses

Register (hexadecimal)	Address	Register (hexadecimal)	Address
0	20080100	9	20080124
1	20080104	10	20080128
2	20080108	11	2008012C
3	2008010C	12	20080130

Register (hexadecimal)	Address	Register (hexadecimal)	Address
4	20080110	13	20080134
5	20080114	14	20080138
6	20080118	15	2008013C
7	2008011C	16	20080140
8	20080120	17	20080144

Operations

The CMCTL performs memory read, memory write, CSR read, and CSR write operations under the control of the CVAX bus. Before an operation can be initiated, a signature read operation is performed during system boot time to load a bank address into CSR0 through CSR15. The signature information for each memory array connected to the PMI bus is stored in CSR0 through CSR15. A signature read operation is initiated for each register by writing a 1 to the signature read request bit in CSR0, CSR4, CSR8, or CSR12. The $\overline{RDY}$ output is not asserted until the signature information is loaded into the CSR. During this operation, the MD<04:00> outputs are first asserted. One of the CAS<3:0> outputs is then asserted together with the SE output to enable the signature read logic on the selected memory array. The RAS<3:0> and $\overline{WE}$ signals are deasserted. The memory array transfers the array signature information to the MD<04:00> outputs. Refer to the *ac Specifications* for the signature read operation. The relationship between the CSRs and CAS<3:0> is provided in the *Addressing* section.

The signature information in the CSRs can be read to determine the number of banks and the RAM size on each memory array. The system boot program determines the address for each banks of RAM and can selectively write to a valid bank address in CSR0 through CSR15.

Memory read—The CMCTL supports masked or unmasked, single- and multiple-longword read operations (quadword, hexword, and octaword). Before a memory read operation, the $\overline{RESET}$ signal must be asserted to initialize CSR0 through CSR15.

The number of cycles necessary to complete a read transfer is a function of the transfer type, the clock (MCLK) input period, the type of memory error detected, if a parity error was detected in an external cache, or if a DMA retry occurred.

The number of cycles required to complete the different types of synchronous memory read operations if no error is detected is shown in Table 15.

Table 15 • CVAX 78588 Synchronous Memory Read Performance

Transfer type	PMI cycle select[1]	CVAX cycle[2]	Total cycles[3]	Megabytes/second
longword	1	4/0	4	10
quadword	1	4/26	13.3	
hexword	1	4/2	8	15
octaword	1	4/2	10	16

Transfer type	PMI cycle select[1]	CVAX cycle[2]	Total cycles[3]	Megabytes/second
longword	0	5/0	5	10
quadword	0	5/3	8	12.5
hexword	0	5/3	11	13.6
octaword	0	5/3	14	14.3

[1]Bit 13 of the mode control and status register (CSR17)
[2]The PMI bus cycle period is an integral number of CVAX bus cycles. Example: 4/2 specifies that the first longword is read in four CVAX bus cycle and the remaining longwords in two CVAX bus cycles. When the PMI cycle select is 1, the PMI bus cycle is 4/2 and one cycle is 100 ns. When the PMI cycle select bit is 0, the PMI bus cycle is 5/3 and one cycle is 80 ns.
[3]One CVAX bus cycle or two MCLKA or MCLKB clock cycles.

The following are exceptions to the table information:

- If a transfer is collides with a refresh operation, add four cycles if the PMI cycle select bit is 1 and five cycles if the PMI cycle select bit is 0.
- Add one cycle to the first transfer of a read lock.
- Add one cycle for each transfer with a correctable error if it is the first transfer or if the PMI cycle select bit is a 1.
- Add two cycles for each transfer with an uncorrectable error. The memory read operation is terminated during the transfer that the error was detected.
- Add one cycle for each asynchronous transfer.
- A read lock with the lock bit already set requires the same number of cycles as a single transfer.
- A read operation that is aborted by external logic requires the same number of cycles as a single transfer.

Aborting read—Before the first longword is transferred, the CMCTL evaluates the $\overline{ERR}$ or $\overline{RDY}$ signal to determine if a memory read operation should be allowed to continue or to be aborted. To abort a local memory read access provides the ability to access an external cache together with a local memory access and to support bus interlocked read operations.

If an external cache asserts the $\overline{RDY}$ signal, the CMCTL will abort its memory read operation. If external logic lock mechanism is implemented, asserting the $\overline{RDY}$ and $\overline{ERR}$ signal simultaneously during a read lock will result in a CVAX CPU retry and will abort the CMCTL memory read operation.

Table 16 lists the number of cycles between the assertion of the $\overline{AS}$ signal and the $\overline{RDY}$ signal or $\overline{ERR}$ inputs to CMCTL. If $\overline{RDY}$ or $\overline{ERR}$ is asserted within the specified number of cycles, the CMCTL aborts the read operation.

Table 16 • CVAX 78588 Read Abort Cycle Timing

Read type	PMI cycle select[1]	CVAX cycle[2]	Input cycles[3]
normal	1	4/2	2
normal	0	5/33	
lock	1	4/2	3
lock	0	5/3	4

[1]Bit 13 of the mode control and status register (CSR17)
[2]The PMI bus cycle period is an integral number of CVAX bus cycles. Example: 4/2 specifies that the first longword is read in four CVAX bus cycle and the remaining longwords in two CVAX bus cycles. When the PMI cycle select is 1, the PMI bus cycle is 4/2 and one cycle is 100 ns. When the PMI cycle select bit is 0, the PMI bus cycle is 5/3 and one cycle is 80 ns.
[3]One CVAX bus cycle or two MCLKA or MCLKB clock cycles.

Single transfers—During a single-transfer read from memory operation, the CMCTL loads the address from CDAL<31:00>, compares the address to the values stored in CSR0 through CSR15, evaluates CDAL<31:30> for transfer length, and evaluates the $\overline{CS}$ and $\overline{DMG}$ inputs.

During a synchronous operation, the CMCTL evaluates the $\overline{RDY}$ or $\overline{ERR}$ inputs to determine if the operation should be aborted. This function is inhibited during assynchronous operation. If $\overline{RDY}$ or $\overline{ERR}$ is asserted, the memory read cycle on the PMI bus is completed, but the CMCTL does not drive the CVAX bus. If $\overline{RDY}$ or $\overline{ERR}$ is not asserted, the CMCTL reads data from the PMI bus and checks the data for errors. The CMCTL asserts $\overline{RDY}$ or $\overline{ERR}$ to indicate the transfer is complete.

Multiple transfers—During a multiple transfer read operation, the CMCTL loads an address from CDAL<31:00> and returns two longwords (quadword), three longwords (hexword), or four longwords (hexword). The CMCTL then loads the address from CDAL<31:00>, compares the address to the values stored in CSR0 through CSR15, evaluates CDAL<31:30> for transfer length, and evaluates the $\overline{CS}$ and $\overline{DMG}$ inputs.

During a synchronous operation, the CMCTL evaluates the $\overline{RDY}$ or $\overline{ERR}$ inputs to determine if the operation should be aborted. The abort function is inhibited during asynchronous operation. If $\overline{RDY}$ or $\overline{ERR}$ is asserted, the PMI bus read cycle is completed. The CMCTL does not drive the CVAX bus and the operation is aborted. If $\overline{RDY}$ or $\overline{ERR}$ is not asserted, the CMCTL reads data from the PMI bus, checks the data for errors, passes the data to CDAL<31:00>, and asserts the $\overline{RDY}$ or $\overline{ERR}$ to flag the termination of the transfer. The operation is terminated when the last longword is read from the PMI bus.

Read lock—The CMCTL performs the read lock operations similar to a memory read operation and monitors the lock bit 6 of configuration registers (CSR0-CSR15). If a read lock is detected, the CMCTL requests a retry and completes the PMI bus read operation on the first transfer. It then terminates the operation on the CVAX bus by asserting both the $\overline{ERR}$ and $\overline{RDY}$ signals. The CMCTL will not retry a read lock. If a read lock is not detected, the CMCTL completes the requested read operation.

The CMCTL sets the lock bit if an uncorrectable error does not occur on any transfer. An extra cycle is added to the first transfer to allow sufficient time for external asynchronous logic to terminate the operation with the $\overline{RDY}$ or $\overline{ERR}$ signals.

Memory write—The CMCTL performs single and multiple word memory write operations (masked or unmasked) on each transfer. The $\overline{\text{RESET}}$ signal must be asserted prior to the operation. The number of cycles necessary to complete a write operation is a function of the transfer type, the status of PMI cycle select bit in CSR17, the $\overline{\text{BM<3:0>}}$ inputs, and whether the transfer is synchronous or asynchronous.

Table 17 lists the number of cycles required to complete each type of synchronous write. When the PMI cycle select bit is 1, one CVAX bus cycle is 100 nanoseconds. When the PMI cycle select bit is 0, one CVAX bus cycle is 80 nanoseconds.

Table 17 : CVAX 78588 Synchronuos Write Performance

Transfer type	PMI cycle Select[1]	Byte mask	CVAX cycle[2]	Total cycles[3]	MB/second
longword	1	unmasked	2/4	4	10
longword	1	unmasked	3/0	4	10
quadword	1	unmasked	3/2	6	13.3
hexword	1	unmasked	3/2	8	15
octaword	1	unmasked	3/2	10	16
longword	1	masked	5/5	6	6.6
quadword	1	masked	5/5	11	6.6
hexword	1	masked	5/5	16	6.6
octaword	1	masked	5/5	21	6.6
longword	0	unmasked	2/4	4	12.5
longword	0	unmasked	3/0	4	12.5
quadword	0	unmasked	3/2	6	16.6
hexword	0	unmasked	3/2	8	18.8
octaword	0	unmasked	3/2	10	20
longword	0	masked	6/6	7	7.1
quadword	0	masked	6/6	13	7.1
hexword	0	masked	6/6	19	7.1
octaword	0	masked	6/6	25	7.1

[1]Bit 13 of the mode control and status register (CSR17)

[2]The PMI bus cycle period is an integral number of CVAX bus cycles. Example: 4/2 specifies that the first longword is read in four CVAX bus cycle and the remaining longwords in two CVAX bus cycles. When the PMI cycle select is 1, the PMI bus cycle is 4/2 and one cycle is 100 ns. When the PMI cycle select bit is 0, the PMI bus cycle is 5/3 and one cycle is 80 ns.

[3]One CVAX bus cycle or two MCLKA or MCLKB clock cycles.

The following are exceptions to the synchronous write operations:

- When an operation collides with a refresh operation, add four cycles if the PMI cycle select bit is a 1 and five cycles if it is a 0.
- If a data parity error or an uncorrectable memory error occurs during a write operation, add one cycle.
- When the $\overline{BM<3:0>}$ information changes between masked and unmasked during a write transfer, add one cycle.
- Add one cycle for each asynchronous transfer.

Error handling—When $\overline{DPE}$ is asserted, the CMCTL tests the data parity on CDAL<31:00>. If a data parity error is detected during an unmasked or masked transfer with no uncorrectable memory errors, the CMCTL writes the data and the incorrect check bits onto the PMI bus. The algorithm for generating incorrect check bits is specified in the *Error Checking* section.

The CMCTL performs masked write transfers by doing a memory read and checking the data for memory errors. If a correctable memory error is detected, it is corrected before the memory write is completed. If an uncorrectable error is detected including the data on CDAL<31:00>, the CMCTL does not execute the write portion of the masked write. A memory read operation instead of the memory write operation does not change the data in memory.

The $\overline{ERR}$ output is asserted to indicate uncorrectable memory errors and DMA-related data parity errors. A correctable error is reported as an interrupt by the assertion of the $\overline{CRD}$ output. The $\overline{MEMERR}$ output is asserted to indicate data parity errors resulting from operations initiated by the CVAX CPU.

Single transfers—During a single transfer write operation, the CMCTL loads the address from CDAL<31:00>, compares the address to the values stored in CSR0 through CSR15, evaluates CDAL<31:30> for transfer length and monitors the $\overline{CS}$ and $\overline{DMG}$ inputs.

If the $\overline{DMG}$ signal is not asserted, the write cycle is CVAX-initiated and the CMCTL evaluates the BM<3:0> inputs to determine if the write transfer is masked or unmasked. An unmasked write is a dump and run operation to allow the CMCTL to keep up with an external write-through cache and not degrade the single transfer write performance. During a masked write operation, the CMCTL receives data from CDAL<31:00> and asserts the $\overline{RDY}$ signal. If the $\overline{DPE}$ signal is asserted, the data is checked for parity errors. If no error is detected, a PMI bus write is initiated with correct check bits. If an error is detected, the CMCTL asserts the $\overline{MEMERR}$ output, a PMI bus write operation is initiated, and the data is transferred to the PMI bus with incorrect check bits.

During DMA transfers, the $\overline{DMG}$ signal is asserted. If $\overline{DPE}$ is asserted, the data is checked for parity errors. If an error is not detected, the $\overline{RDY}$ signal is asserted, the CMCTL transfers the data to the PMI bus with correct check bits, and the CVAX bus is released. If an error is detected, the data is transferred to the PMI bus with incorrect check bits and the ERR output is asserted for two CVAX bus cycles.

Multiple transfers—Multiple-transfer write operations are performed as a page mode write operation to the PMI bus. The CMCTL loads a single address and performs multiple page mode memory data transfers (including check bits) from the CVAX bus to the PMI bus. Refer to the *Addressing* section for a description of the page mode.

During the write operation, the CMCTL receives the address from CDAL<31:00> and compares the address to the values stored in CSR0 through CSR15. It evaluates the CDAL<31:30> for transfer length, the $\overline{CS}$ and $\overline{DMG}$ inputs, and the $\overline{BM<3:0>}$ inputs to determine if the transfer is

masked or unmasked. The $\overline{RDY}$ or $\overline{ERR}$ is asserted (depending on the error conditions detected) to enable the external logic to start the next data transfer. When the last longword has been written to the PMI bus, the CMCTL terminates the operation.

Write unlock—The CMCTL performs a write unlock operation similar to memory write operations except that it clears the lock bit.

CSR read—A CSR read is performed similar to a single-transfer memory read operation except that operation cannot be aborted by the $\overline{RDY}$ and $\overline{ERR}$ inputs. A synchronous CSR read operation is completed within a minimum of four cycles. Asynchronous CSR read operations require additional cycles.

CSR Write—A CSR write operation is performed as a longword (unmasked) transfer. A synchronous CSR write operation is completed within a minimum of four CVAX bus cycles. An asynchronous transfer reqires additional cycle. Because CSR operations are unmasked, the $\overline{BM}<3:0>$ outputs are not used to validate the CDAL<31:00> data during the actual CSR write operations. The CMCTL begins the write operation by loading and checking the CDAL<31:00> address and by evaluating the $\overline{CS}$ and $\overline{DMG}$ inputs. If the $\overline{DPE}$ input is asserted, the data on CDAL<31:00> is checked for parity errors. If an error is detected, the write operation is aborted and the $\overline{ERR}$ output is asserted for two CVAX bus cycles. If an error is not detected, the selected CSR is loaded with CDAL<31:00> information and the $\overline{RDY}$ output is asserted.

Bus Operating Modes

When a DMA device is bus master of the CVAX bus, the CMCTL can transfer data either synchronously or asynchronously. The DMA bus operating mode is controlled by the $\overline{CS/DP3}$ line. When the CVAX CPU is bus master of the CVAX bus, the CMCTL response is synchronous with the CVAX CPU.

Asynchronous DMA Mode—If $\overline{CS/DP3}$ is not asserted during the first part of an I/O cycle initiated by a DMA device, the CMCTL responds asynchronously to the CVAX bus by monitoring the $\overline{AS}$ and $\overline{DS}$ inputs. Before the CMCTL starts the next operation, it waits for the assertion of the $\overline{AS}$ before evaluating the CVAX bus to determine the next operation. The $\overline{DS}$ input must be asserted before the CMCTL initiates a transfer. When the CMCTL acknowledges a transfer, it asserts the $\overline{RDY}$ or $\overline{ERR}$ output and waits for the deassertion and subsequent assertion of $\overline{DS}$ before continuing to the next transfer.

During read operations, the $\overline{DS}$ input must be asserted before the CVAX bus outputs are enabled. During write operations, $\overline{DS}$ must be deasserted to determine if CDAL<31:00> has valid data before performing the write operation.

Synchronous Mode—All operations initiated by the CVAX CPU are synchronous. The CMCTL responds synchronously during DMA transfers if the $\overline{CS/DP3}$ signal is asserted during the first part of an I/O cycle. During synchronous operations, the $\overline{DS}$ signal is ignored and the CMCTL does not stall. Before starting a synchronous operation, the CMCTL checks for the assertion of $\overline{AS}$.

• Interfacing Requirements

A typical CVAX system interface, using the CVAX 78588 CMCTL, is shown in Figure 9. The CMCTL can control up to four DRAM arrays on the PMI bus. It provides a nonlocal memory reference for a DMA interface to initiate transfer from external bus devices.

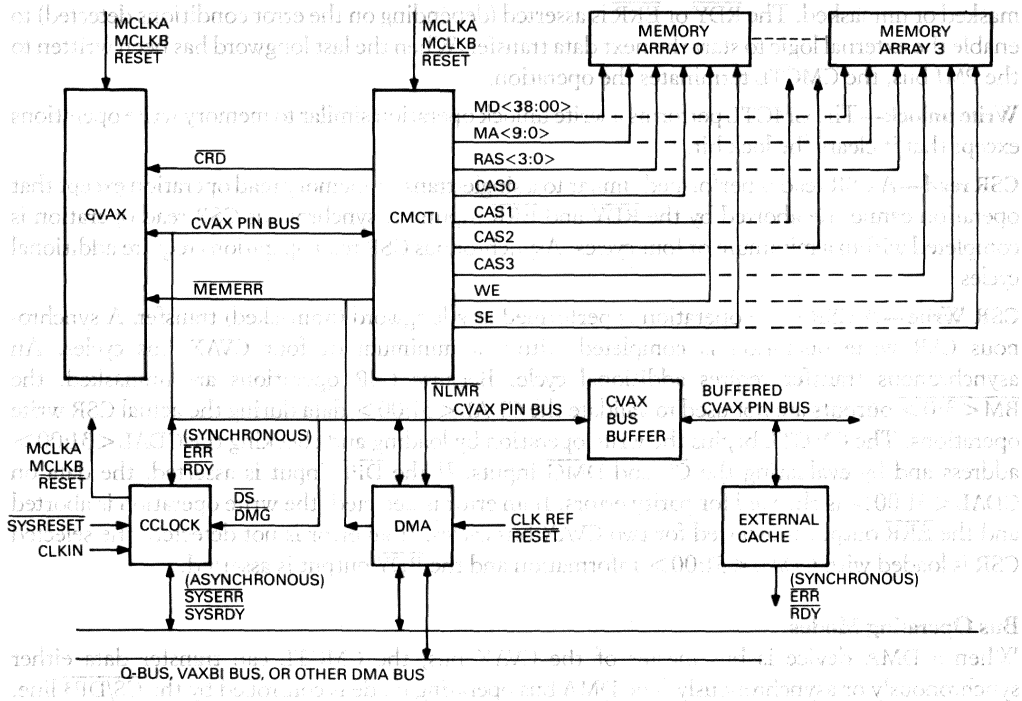


Figure 9 • CVAX 78588 CPU System Interface Diagram

• Specifications

The mechanical, electrical, and environmental characteristics and specifications for the CVAX 78588 are described in the following paragraphs. The test conditions for the electrical values are as follows unless specified otherwise.

- Ambient temperature (T_A): 70°C
- Supply voltage (V_{DD}): 4.75 V to 5.25 V
- Ground reference (V_{SS}): 0 V

Absolute maximum ratings

Stresses greater than absolute maximum ratings may cause permanent damage to a device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

- Power supply voltage (V_{DD}): –0.5 V to 7.0 V
- Storage temperature (T_S): –55°C to 125°C
- Ambient temperature (T_A): 0°C to 70°C
- Package dissipation: 1.8 W
- Input or output voltage applied: (V_{SS}–0.5 V) to (V_{DD} + 1.5 V = 7.0 V)

Recommended Operating Conditions

- Ambient temperature (T_A): 0°C to 70°C
- Power supply voltage (V_{CC}): 5.0 V ±5%
- Supply current (I_{CC}): 336 mA (maximum)
- Relative humidity: 10% to 95% (noncondensing)
- Air flow: 100 linear feet/minute

dc Electrical Characteristics

Table 18 contains the dc electrical parameters for the input and output signals of the CMCTL. Refer to Table 1 for the pin numbers of the signals referenced in Table 18. Table 19 lists the tests that are applicable for the inputs and outputs.

Table 18 • CVAX 78588 dc Input and Output Parameters

Symbol	Parameter	Requirements Min.	Max.	Unit	Test Conditions
V_{IH}	High-level input voltage				
	(TTL)	2.0	—	V	V_{DD} = 5.25 V
	(MOS)	70% V_{DD}	—	V	
V_{IL}	Low-level input voltage				
	(TTL)	—	0.8	V	
	(MOS)	—	30% V_{DD}	V	
V_{OL}	Low-level output voltage				
	V_{OL1} (TTL)	—	0.4	V	I_{OL} = 2.0 mA
	V_{OL2} (MOS)	—	0.4	V	I_{OL} = 24.0 mA
V_{OH}	High-level output voltage				
	(MOS)	90% V_{DD}	—	V	I_{OH} = −0.5 mA
V_{OL}	Low-level output voltage				
	(MOS)	—	10% V_{DD}	V	I_{OL} = 0.5 mA
I_L	Input leakage current	−10	10	µA	$0 < V_{in} < 5.25$ V
I_O	Output leakage current	−10	50	µA	$0 < V_{out} < 5.25$ V V_{DD} = 5.25 V $\overline{TRI\ OUT}$ = 0 V
I_{DD}	Active supply current	—	336	mA	I_O = 0, T_A = 70°C V_{DD} = 5.25 V

Table 19 • CVAX 78588 dc Test Summary

Signal Name	Applicable Test V_{IH}[1]	V_{IL}[1]	V_{OL}[1]1	V_{OL}[2]	I_L	I_O	V_{IH}[2]	V_{IL}[2]	V_{OH}[2]	V_{OL}[2]
$\overline{AS}$					X		X	X		
$\overline{CRD}$			X			X			X	
$\overline{BM3}$–$\overline{BM0}$	X	X			X					
$\overline{CAS3}$–$\overline{CAS0}$						X			X	X
$\overline{CS/DP3}$	X	X		X		X			X	
$\overline{CS/DP2}$–$\overline{CS/DP0}$	X	X	X			X			X	
CDAL31–CDAL00	X	X	X			X			X	
$\overline{DMG}$					X		X	X		
$\overline{DPE}$	X	X	X			X				
$\overline{DS}$					X		X	X		
$\overline{ERR}$ and $\overline{RDY}$	X	X		X		X			X	
MA9–MA0						X			X	X
MCLKA and MCLKB					X		X	X		
MD38–MD00						X	X	X	X	X
$\overline{MEMERR}$				X		X				
$\overline{NLMR}$			X			X			X	
RAS3–RAS0						X			X	X
$\overline{RESET}$	X	X			X					
SE						X			X	X
$\overline{TRI\ OUT}$	X	X			X					
WE						X			X	X
$\overline{WR}$	X	X			X					

[1]TTL level
[2]MOS level

ac Electrical Characteristics

The ac load circuits used in the measurements of the ac parameters are shown in Figure 10 and summarized in Table 20. The input and output pin capacitance is listed in Table 21. The test conditions are

- Temperature (T_A) = 70°C
- V_{SS} = 0 V
- V_{DD} = 4.75 V (except as noted)

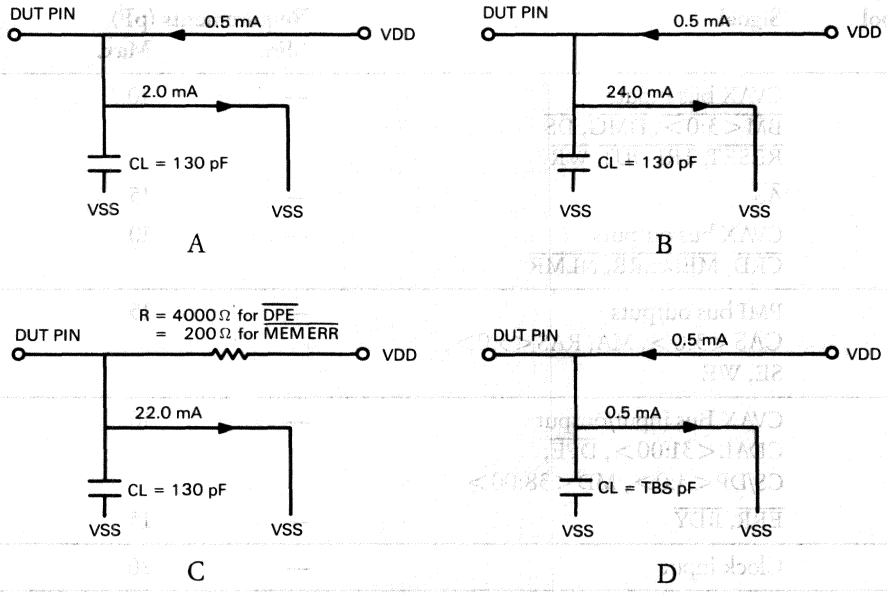


Figure 10 • CVAX 78588 ac Test Circuits

Table 20 • CVAX 78588 ac Test Circuit Summary

Test	Signal	Pin	Load
V_{OL}[1] and V_{OH}[2]	$\overline{CRD}$	56	A
$\overline{CS/DP<2:0>}$	66-64		
	CDAL<31:00>	104-71	
	$\overline{NLMR}$	55	
V_{OL}[2] and V_{OH}[2]	$\overline{CS/DP3}$	69	B
	$\overline{ERR}$	52	
	$\overline{RDY}$	53	
V_{OL}[2] and V_{OH}[2]	$\overline{DPE}$	70	C
	$\overline{MEMERR}$	54	
	CAS<3:0>	32-35	D
	MA<9:0>	20-29	
	MD<38:00>	9-15,105-115, 118-132,3-8	
	RAS<3:0>	38-41	
	SE	45	
	WE	44	

[1]TTL level
[1]MOS level

Table 21 • CVAX 78588 Pin Capacitance

Symbol	Signal	Requirements (pF) Min.	Max.
C_{IC}	CVAX bus inputs $\overline{BM<3:0>}$, $\overline{DMG}$, $\overline{DS}$ $\overline{RESET}$, $\overline{TRI\ OUT}$, $\overline{WR}$	—	10
	$\overline{AS}$	—	15
	CVAX bus outputs $\overline{CRD}$, $\overline{MEMERR}$, $\overline{NLMR}$	—	10
C_{OC}	PMI bus outputs CAS<3:0>, MA, RAS<3:0>, SE, WE	—	15
C_{IOC1}	CVAX Bus input/output CDAL<31:00>, $\overline{DPE}$, CS/DP<3:0>, MD<38:00>	—	10
	$\overline{ERR}$, $\overline{RDY}$	—	15
C_{ICM}	Clock input	—	20

Table 22 lists the signals that can be connected to V_{DD} through an external resistor. Refer to Table 1 for the pin numbers of the signals listed.

Table 22 • CVAX 78588 External Resistor Requirements

Signal	Resistance Ω Min.	Max.	Remarks
$\overline{CS/DP3}$	200	620	Accommodates asynchronous DMA devices that do not drive this pin.
$\overline{DPE}$	4 k		Used as a passive pull-up for external logic that may be wire ORed to this input. Larger value may be used if the passive pullup timing can be met.
$\overline{ERR}$	200		Required for external logic.
$\overline{MEMERR}$	200		Used as a passive pull-up for external logic that may be wire ORed to this input. Larger value may be used if the passive pullup timing can be met.
$\overline{RDY}$	200		Required for external logic.

ac Synchronous Characteristics

A subset of the CMCTL operations is used to specify the phase timing for the CMCTL signals. The ac test conditions are

- Supply voltage (V_{DD}): 4.75 V
- Ambient temperature (T_A): 70°C

- V_{IL} (MOS) = 10% V_{DD}
- V_{IH} (MOS) = 90% V_{DD}
- V_{IL} (TTL) = 0.8 V
- V_{IH} (TTL) = 2.0 V

Table 23 lists the input signal timing parameters and Table 24 list the output signal timing parameters.

Table 23 • CVAX 78588 Synchronous Input Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{AS}	$\overline{AS}$ setup	5.0	—
t_{BS}	$\overline{BM<3:0>}$ setup	25	—
t_{BH}	$\overline{BM<3:0>}$ hold	0	—
t_{CC}	External clock cycle	50	*
t_{CD}	MCLKA to MCLKB delay	$t_{cc}/2-0.5$	$t_{cc}/2+0.5$
t_{CH}	External clock high	15	25
t_{CL}	External clock low	15	25
t_{CR}	External clock rise/fall	0	5
t_{CSS}	$\overline{CS}$ setup	25	—
t_{CSH}	$\overline{CS}$ hold	0	—
t_{DAS}	CDAL address setup	25	—
t_{DAH}	CDAL address hold	0	—
t_{DDS}	CDAL data setup	25	—
t_{DDH}	CDAL data hold	0	—
t_{ERS}	$\overline{ERR}$ and $\overline{RDY}$ setup	15	—
t_{ERH}	$\overline{ERR}$ and $\overline{RDY}$ hold	10	—
t_{PS}	$\overline{DP}$ and $\overline{DPE}$ setup	10	—
t_{PH}	$\overline{DP}$ and $\overline{DPE}$ hold	0	—
t_{PMIS}	MD<38:00> setup	0	—
t_{PMIH}	PMI bus input hold	10	—
t_{RS}	$\overline{Reset}$ input prior to P1 setup	20	$t_{cc}-10$
t_{RW}	$\overline{Reset}$ input width	$4t_{cc}$	—
t_{SS}	Strobe setup	0	—

Symbol	Definition	Requirements (ns) Min.	Max.
t_{SH}	Strobe hold	0	—
t_{SYNS}	$\overline{AS}$ and $\overline{DS}$ synchronizer setup	10	—
t_{SYNH}	$\overline{AS}$ and $\overline{DS}$ synchronizer hold	10	—
t_{WS}	$\overline{WR}$ setup	25	—
t_{WH}	$\overline{WR}$ hold	0	—

*To be continued

Table 24 • CVAX 78588 Synchronous Output Timing Parameters

Symbol	Parameter	Requirements (ns) Min.	Max.
t_{ASERH}	$\overline{AS}$ to $\overline{ERR}$ and $\overline{RDY}$ hold time	0	25
t_{ASND}	$\overline{AS}$ to $\overline{NLMR}$ delay	—	25
t_{CASL}	CAS<3:0> low	70*	—
t_{CMND}	$\overline{CRD}$, $\overline{MEMERR}$ and $\overline{NLMR}$ delay	—	25
t_{DDD}	CDAL<31:00> delay	—	20
t_{DSERD}	$\overline{DS}$ to $\overline{ERR}$ and $\overline{RDY}$ delay	—	25
t_{DSDDH}	$\overline{DS}$ to CDAL<31:00> high impedance	0	25
t_{DSPH}	$\overline{DS}$ to DP and $\overline{DPE}$ high impedance	0	25
t_{ERD}	$\overline{ERR}$ and $\overline{RDY}$ delay	—	30
t_{MACASH}	MA<9:0> to CAS<3:0> hold	150*	—
t_{MAPMIS}	MA<9:0> to strobe setup	20*	—
t_{MARASH}	MA<9:0> to RAS<3:0> hold	25*	—
t_{MDD}	MD<38:00> delay	0	30
t_{MDH}	MD<38:00> hold	0	25
t_{PD}	$\overline{DP}$ and $\overline{DPE}$ delay	—	25
t_{PMID}	PMI bus delay	—	15
t_{RASL}	RAS<3:0> low	120*	—
t_{RSTD}	$\overline{RESET}$ to CDAL<31:00>, CSDP<3:0>, $\overline{DPE}$, $\overline{RDY}$, $\overline{ERR}$, $\overline{MEMERR}$, $\overline{CRD}$, and $\overline{NLMR}$	—	50

*Preliminary values subject to change

Clock Input Timing

Figure 11 shows the MCLKA and MCLKB external clock input timing for synchronous operation of the CMCTL. The timing parameters are listed in Table 24.

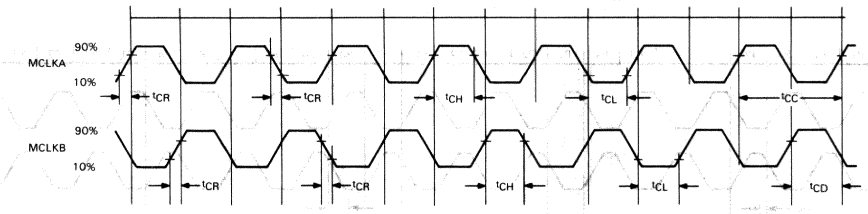


Figure 11 • CVAX 78588 Clock Input Timing

Initialization

Figure 12 shows the $\overline{\text{RESET}}$ input timing for initializing of the CMCTL for synchronous operation.

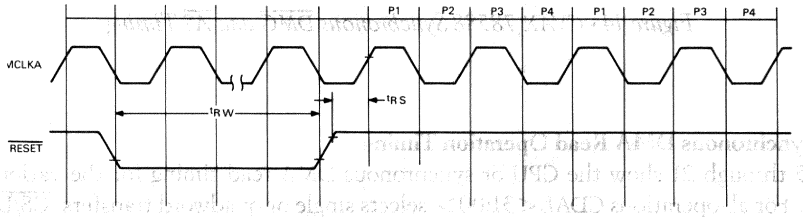


Figure 12 • CVAX 78588 Initialization Timing (Synchronous Operation)

CVAX Bus Reset Timing

Figure 13 shows the relationship of the $\overline{\text{RESET}}$ signal to the CVAX bus signals. When $\overline{\text{RESET}}$ is asserted, CDAL<31:00>, $\overline{\text{CS/DP}}$<3:0>, $\overline{\text{DPE}}$, $\overline{\text{RDY}}$, $\overline{\text{ERR}}$ and $\overline{\text{MEMERR}}$ outputs are asynchronously set to a high impedance and $\overline{\text{CRD}}$ and $\overline{\text{NLMR}}$ are asynchronously deasserted.

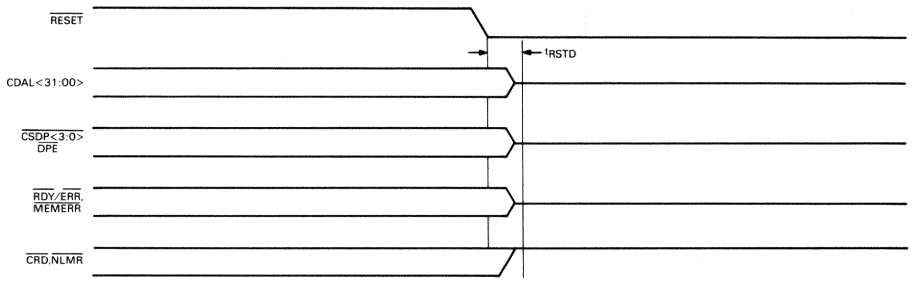


Figure 13 • CVAX 78588 CVAX Bus Reset Timing

Synchronous $\overline{DMG}$ and $\overline{AS}$ Timing

Figure 14 shows the phase timing of the $\overline{DMG}$ and $\overline{AS}$ inputs during a synchronous DMA operation. The timing relation of these signals is the same for a CSR or memory operation. The $\overline{CS/DP3}$ input selects synchronous or asynchronous DMA operation.

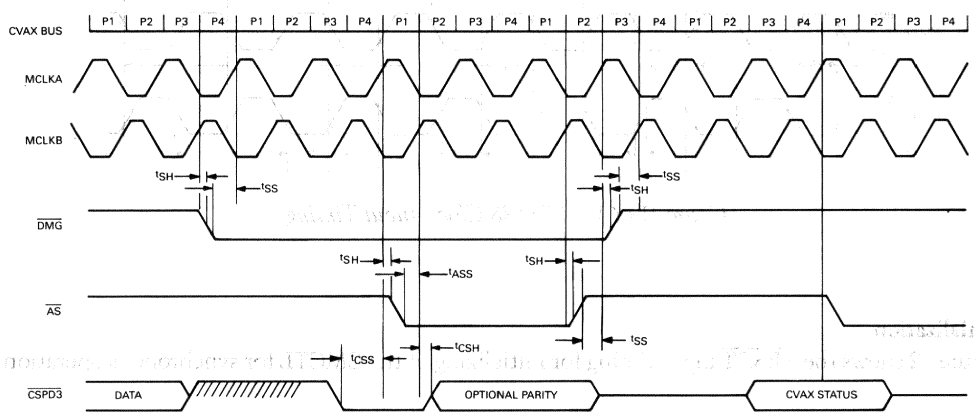


Figure 14 • CVAX 78588 Synchronous $\overline{DMG}$ and $\overline{AS}$ Timing

CPU or Synchronous DMA Read Operation Timing

Figures 15 through 21 show the CPU or synchronous DMA read timing for the various transfer operation. For all operations CDAL<31:00> selects single or quadword transfers, $\overline{CS/DS}$<3:0> selects no read lock and the PMI cycle is 4/2 selected by the control and status register CSR17. For additional information on read lock, refer to the *Operations* section. Refer to the *Register* section for information on the PMI cycle select functions of CSR17.

Single Transfer Read (no memory errors)—Figure 15 shows the phase timing for a single transfer to a CSR or a memory read operation with no errors detected. The phase timing relationship is not shown in the following diagrams unless it is different.

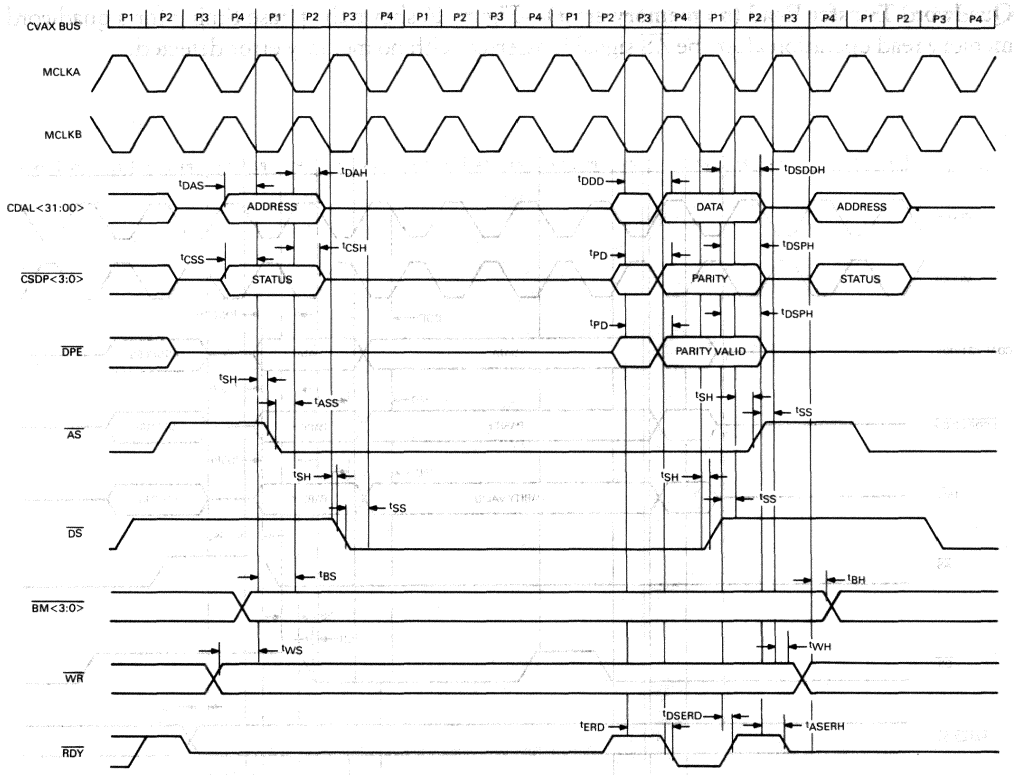


Figure 15 ▪ CVAX 78588 CPU or Synchronous DMA Read Timing (Single—No Memory Error)

Quadword Transfer Read (no memory error)—Figure 16 shows the phase timing for a quadword memory read operation after the $\overline{AS}$ signal is asserted with no memory error detected.

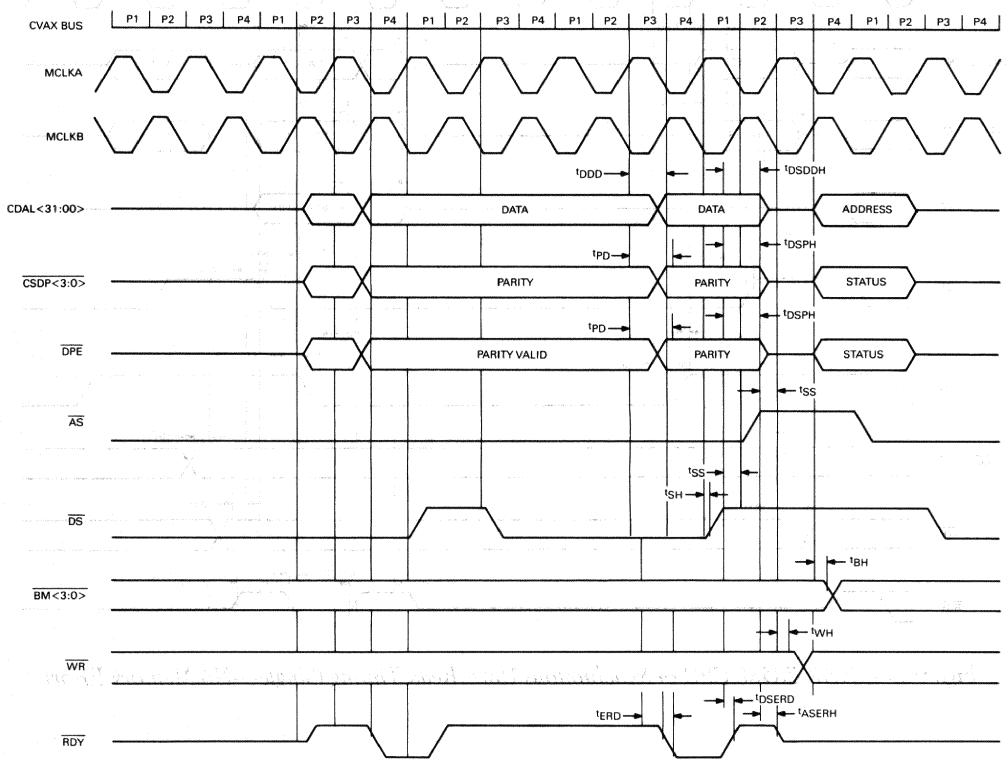


Figure 16 ▪ CVAX 78588 CPU or Synchronous DMA Read Timing (Quadword—No Memory Error)

Single Transfer Read (uncorrectable memory error)—Figure 17 shows the timing for the $\overline{ERR}$ output during a single transfer memory read operation with an uncorrectable memory error. The memory read operation is terminated when the $\overline{ERR}$ signal is asserted indicating that an uncorrectable error was detected. This timing is similiar for quadword, hexword, and octaword transfers.

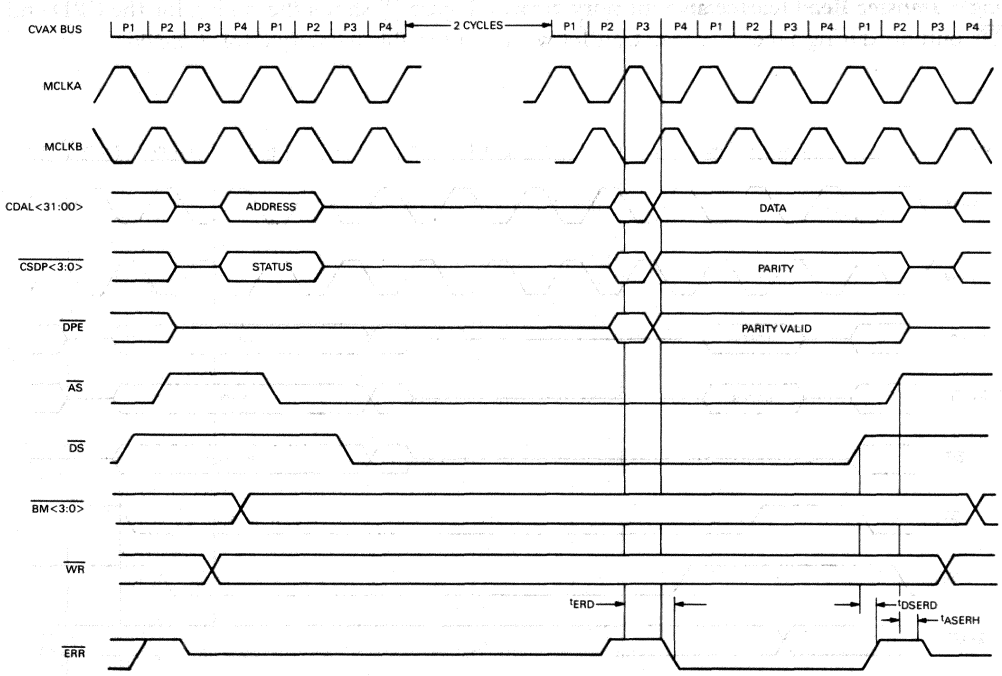


Figure 17 • CVAX 78588 CPU or Synchronous DMA Read Timing (Single—Uncorrectable Memory Error)

Single Transfer Read (correctable memory error)—Figure 18 shows the timing for the $\overline{CRD}$ and $\overline{RDY}$ outputs during a memory read transfer with a correctable memory error detected.

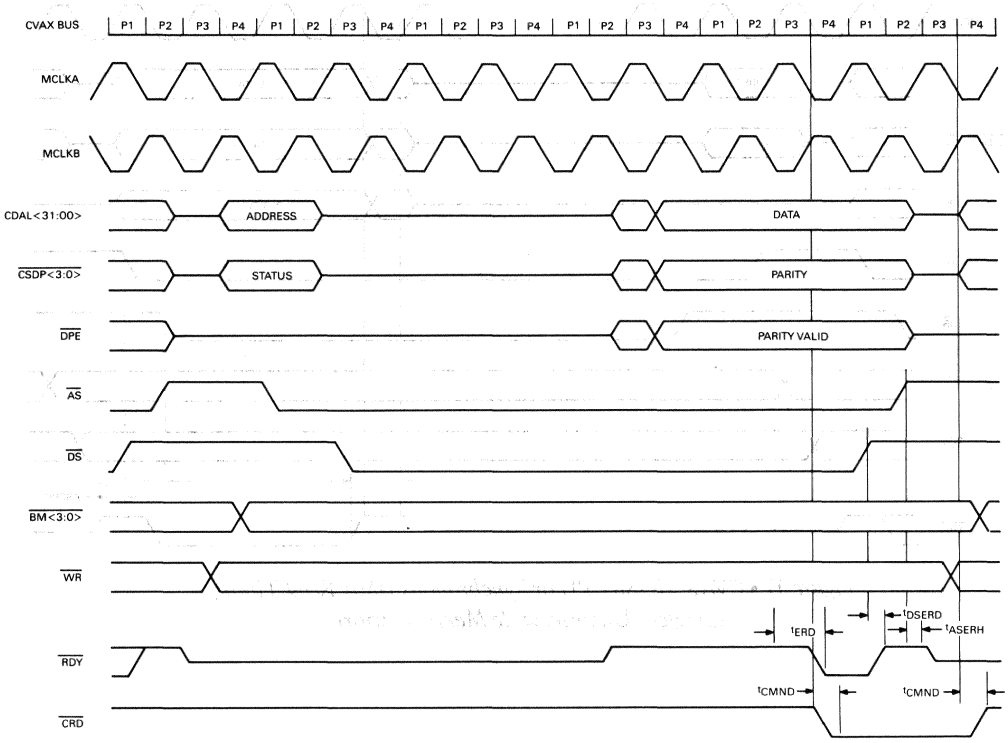


Figure 18 • CVAX 78588 CPU or Synchronous DMA Read Timing (Single—Correctable Memory Error)

Quadword Transfer Read (correctable memory error)—Figure 19 shows the phase timing for a quadword memory read after the $\overline{AS}$ input is asserted with a correctable memory error detected.

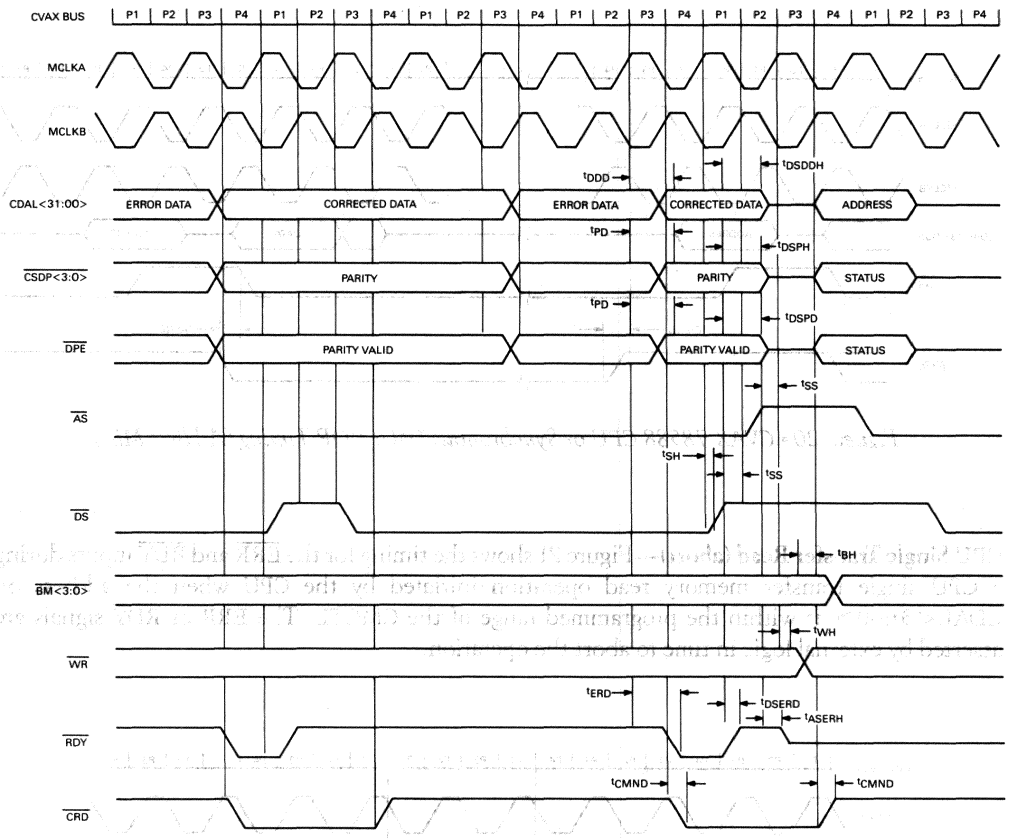
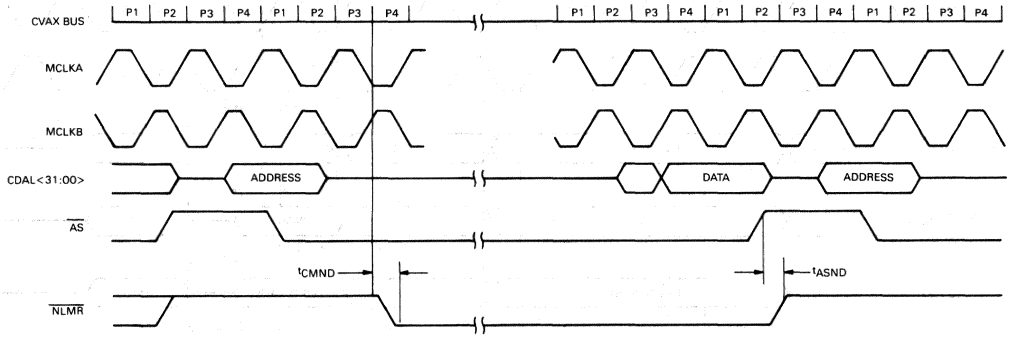


Figure 19 • CVAX 78588 CPU or Synchronous DMA Read Timing (Quadword—Correctable Memory Error)

Single Transfer Read (address miss NOP)—Figure 20 shows the timing for the $\overline{\text{NLMR}}$ output during a CPU or DMA single transfer to a CSR or a memory read operation when the address on CDAL<31:00> is not within the programmed range of the CMCTL.



Figuew 20 • CVAX 78588 CPU or Synchronous DMA NOP Timing (Address Miss)

CPU Single Transfer Read (abort)—Figure 21 shows the timing for the $\overline{\text{ERR}}$ and $\overline{\text{RDY}}$ inputs during a CPU single transfer memory read operation initiated by the CPU when the address on CDAL<31:00> is within the programmed range of the CMCTL. The $\overline{\text{ERR}}$ or $\overline{\text{RDY}}$ signals are asserted by external logic in time to abort the operation.

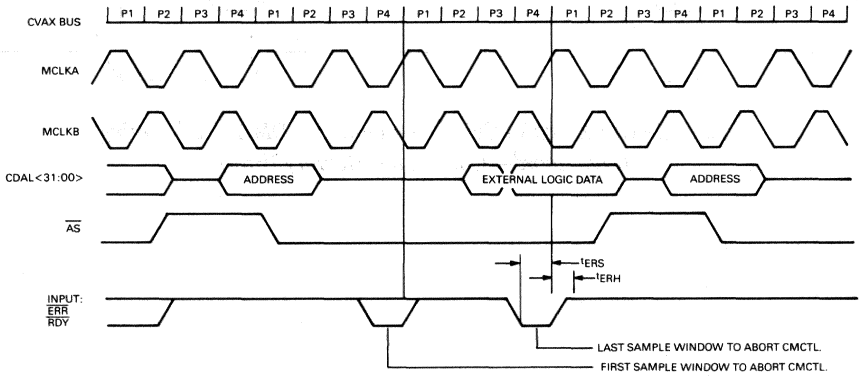
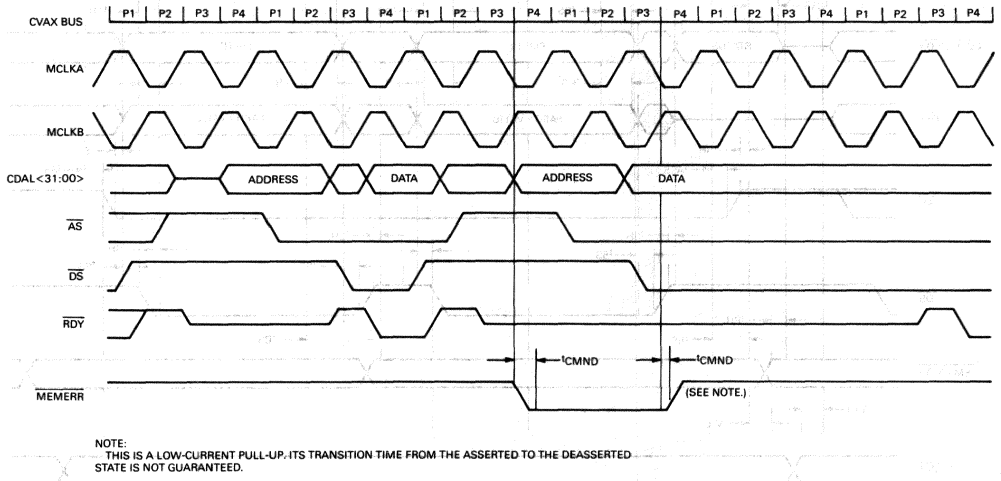


Figure 21 • CVAX 78588 CPU Memory Read Abort Timing (External Logic)

CPU or Synchronous DMA Write Operation Timing

Figures 22 and 23 show the phase timing for the CPU or synchronous DMA write operations.

Single Transfer CPU Write (unmasked)—Figure 22 shows the timing of the $\overline{\text{MEMERR}}$ output. This is an open-drain output and requires an external pullup resistor connected to V_{BB}. Other timing considerations are the same as shown in Figure 23.



NOTE:
THIS IS A LOW-CURRENT PULL-UP. ITS TRANSITION TIME FROM THE ASSERTED TO THE DEASSERTED STATE IS NOT GUARANTEED.

Figure 22 • CVAX 78588 CPU Write Timing (Single—Unmasked)

Quadword Transfer DMA write (no error)—Figure 23 shows the timing for synchronous quadword unmasked write operation when no parity errors are detected on the CDAL < 31:00 > data. When a parity error is detected, the $\overline{\text{ERR}}$ signal is asserted instead of the $\overline{\text{RDY}}$ signal and at the same time. When $\overline{\text{ERR}}$ is asserted, the $\overline{\text{DS}}$ input must remain asserted for an additional cycle before it is deasserted to start the next data transfer. The timing for the $\overline{\text{ERR}}$ signal is the same as the synchronous read operation. The timing for a masked write operation is similiar to the unmasked write timing except that one or more slip cycles occur after $\overline{\text{AS}}$ is asserted and until the $\overline{\text{RDY}}$ or $\overline{\text{ERR}}$ signal is asserted. There are also more error conditions. Table 25 lists the number of slip cycles added in relation to the type of write operation and error condition. The $\overline{\text{CRD}}$, $\overline{\text{ERR}}$, and $\overline{\text{RDY}}$ signals are asserted for the same duration as for a memory read operation.

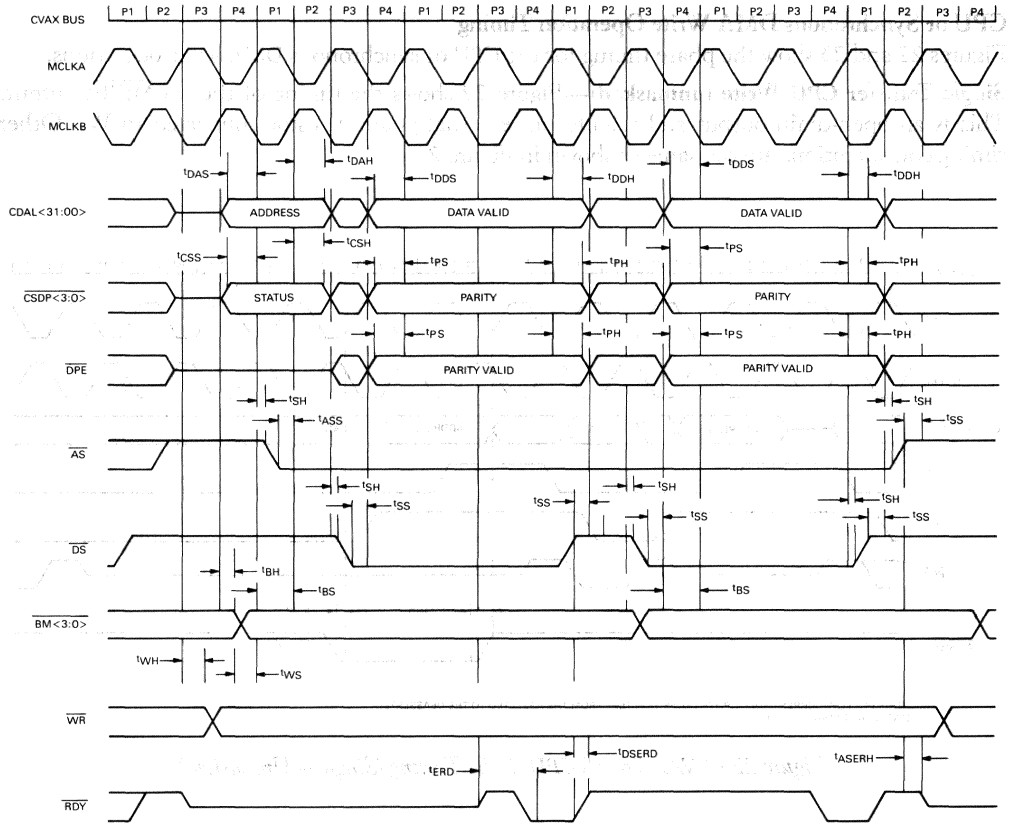


Figure 23 • CVAX 78588 Synchronous DMA Write Timing (Quadword—No Error)

Table 25 • CVAX 78588 Write Operation Slip Cycles

Write operation	Error type	Slip cycles from $\overline{RDY}$ (ns)	Output asserted
Unmasked[1]	none	0	$\overline{RDY}$
	bus parity	0	$\overline{ERR}$
First transfer masked[2]	none	$4t_{cc} = 200$	$\overline{RDY}$
	correctable	$4t_{cc} = 200$ $\overline{RDY}$ and $\overline{CRD}$	
	uncorrectable	$4t_{cc} = 200$ $\overline{ERR}$	
Second transfer masked[2,3]	none	$6t_{cc} = 300$	$\overline{RDY}$
	correctable	$6t_{cc} = 300$ $\overline{RDY}$ and $\overline{CRD}$	
	uncorrectable	$6t_{cc} = 300$ $\overline{ERR}$	
	bus parity	$6t_{cc} = 300$ $\overline{ERR}$	

[1]Add $2t_{cc} = 100$ ns to all transfers except the first if the previous transfer is masked.
[2]Add $2t_{cc} = 100$ ns if the PMI cycle select bit of CSR17 is set.
[3]Add $2t_{cc} = 100$ ns to all transfers except the first if the previous transfer is unmasked.

PMI Bus Timing

Figures 24 through 33 show the timing of the PMI bus during a reset condition and during read and write operations. The timing shown is a result of the PMI cycle select bit being set and no errors being detected. When the cycle select bit is cleared, the CAS<3:0> outputs are asserted for an additional microcycle during a memory read operation or during the read portion of a masked write operation and the RAS<3:0> and SE outputs are asserted for an additional cycle during a memory refresh operation. No timing conditions on the CAS<3:0>, MA<9:0>, RAS<3:0>, and WE outputs for an error condition are not critical and are not specified.

The timing for multiple masked write operations is similar to a single masked write, Figure 28, except that the RAS<3:0> outputs are asserted during the operation. For the read signature operation, the RAS<3:0> outputs are negated.

During memory refresh, Figure 34, the CAS<3:0> and WE ouputs are negated and the MD<38:00> lines provides the command.

The synchronizer timing for asynchronous DMA operations, Figure 34, is used during manufacturing test to determine the asynchronous cycle response time of CMCTL.

RESET—Figure 24 shows the timing of the PMI bus signals with respect to the $\overline{\text{RESET}}$ input. When $\overline{\text{RESET}}$ is asserted, all PMI outputs are asynchronously negated. The RAS<3:0>, CAS<3:0>, WE, and SE outputs are negated and then asserted for the first pass of the CMCTL chip.

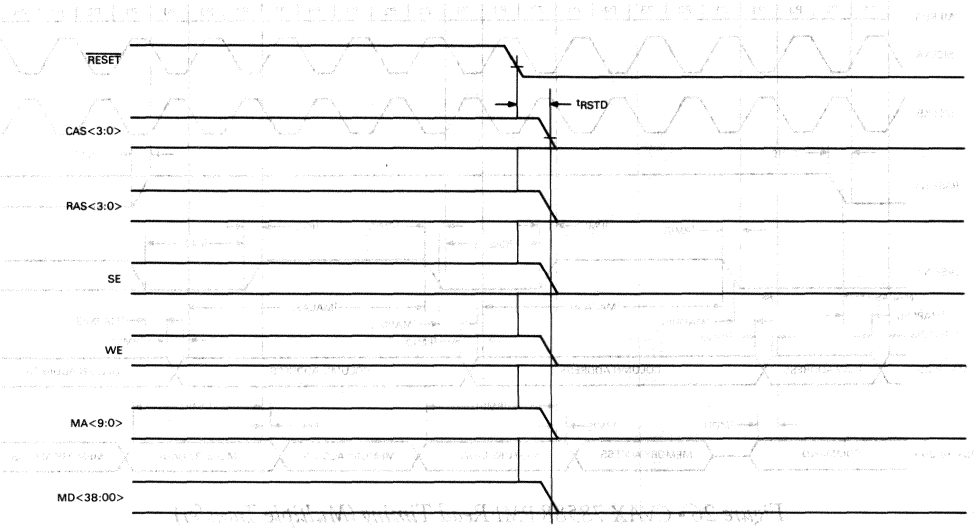


Figure 24 ▪ CVAX 78588 PMI Reset Timing

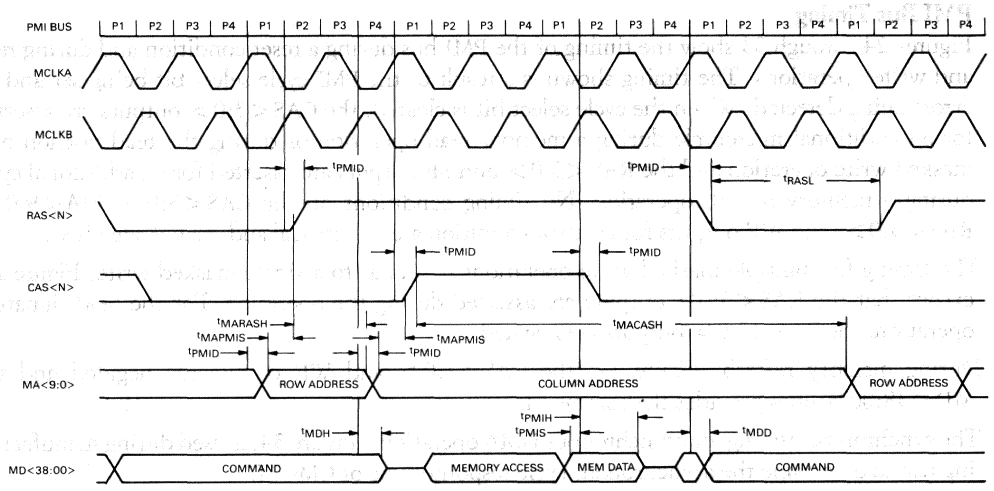


Figure 25 • CVAX 78588 PMI Read Timing (Single Transfer)

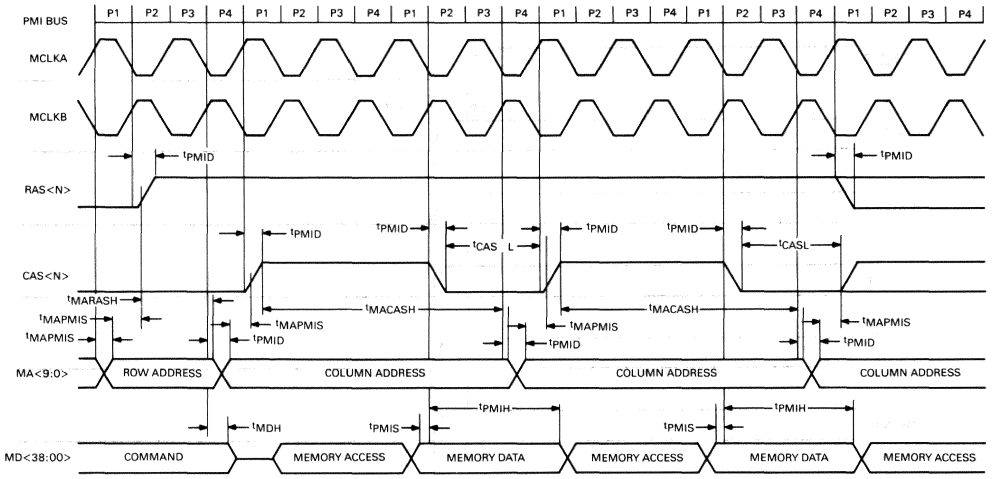


Figure 26 • CVAX 78588 PMI Read Timing (Multiple Transfer)

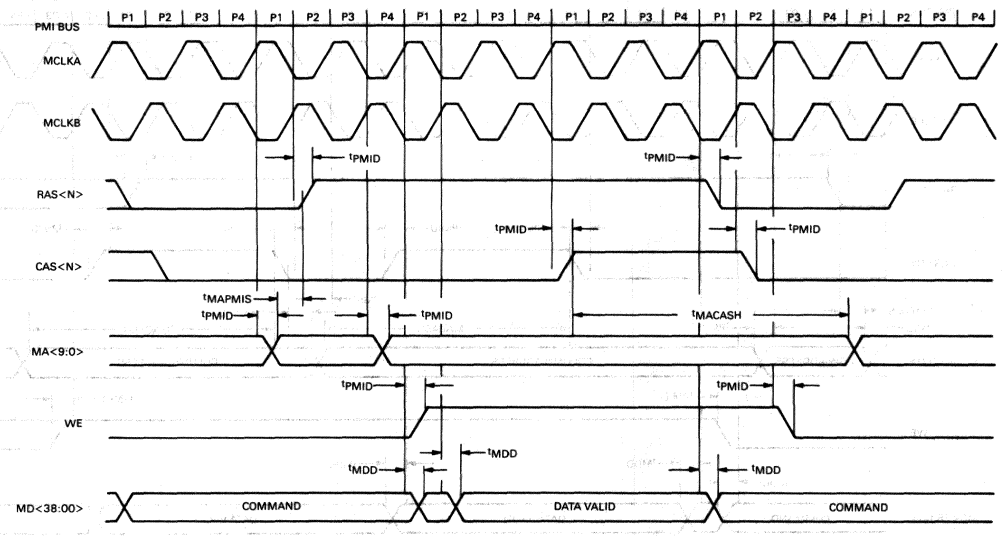


Figure 27 • CVAX 78588 PMI Write Timing (Single Unmasked Transfer)

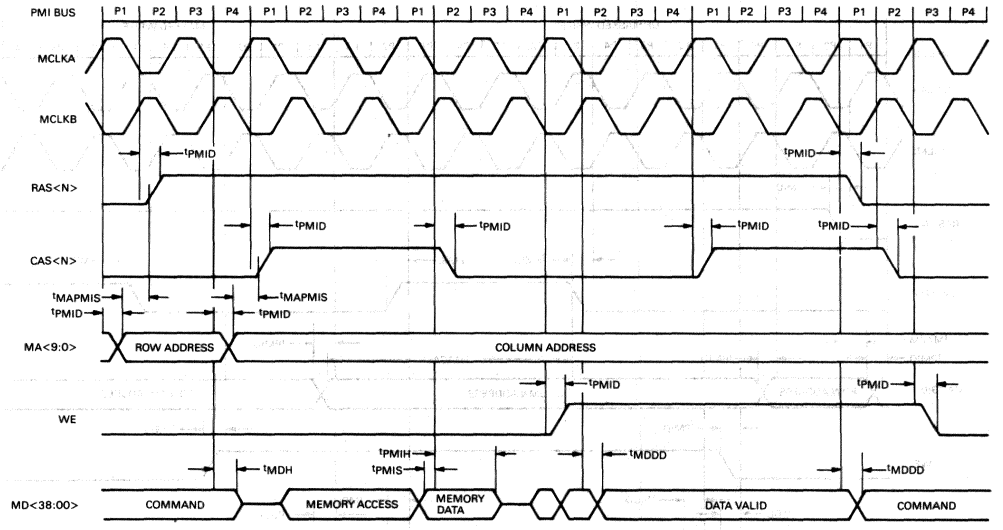


Figure 28 • CVAX 78588 PMI Write Timing (Single Masked Transfer)

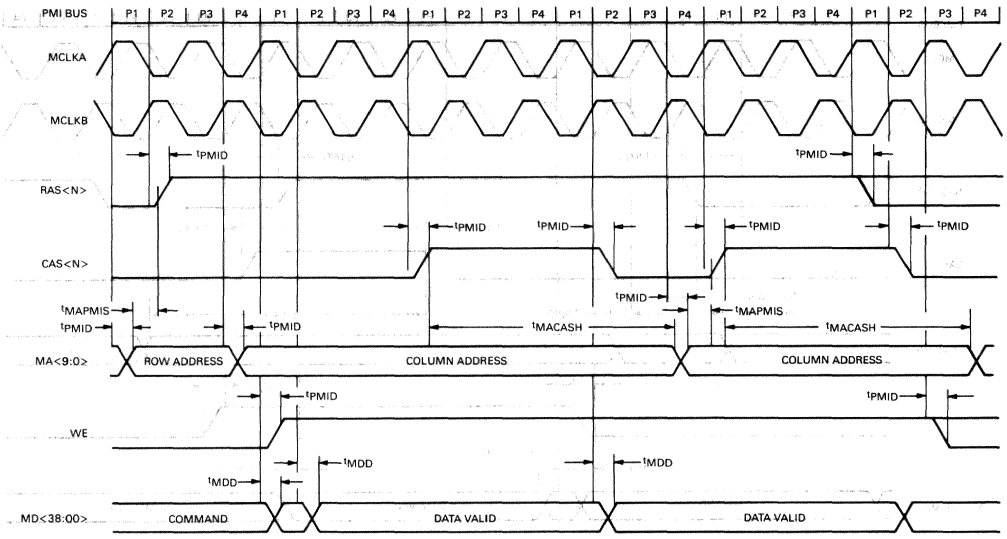


Figure 29 ▪ CVAX 78588 PMI Write Timing (Multiple Unmasked Transfer)

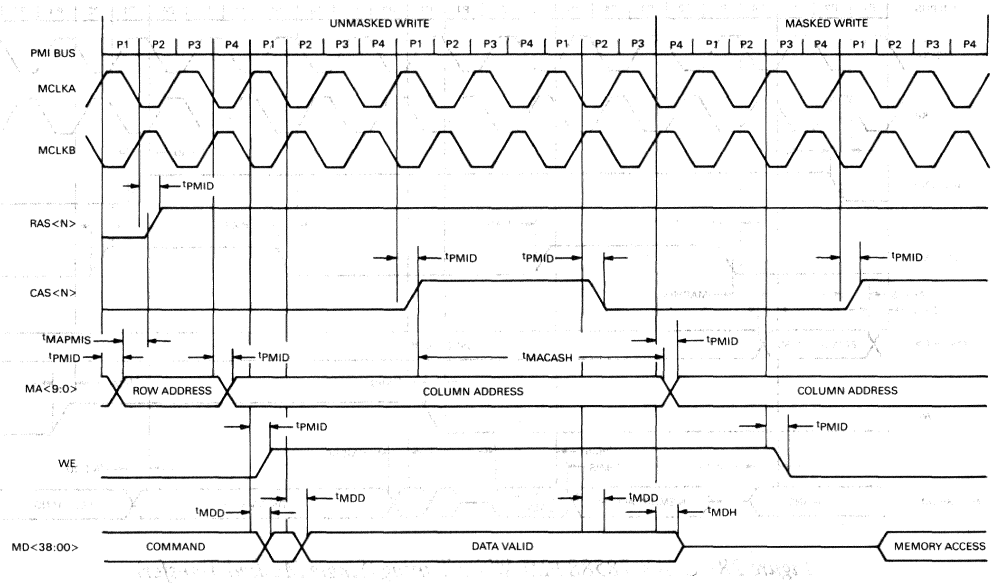


Figure 30 ▪ CVAX 78588 PMI Write Timing (Multiple Unmasked and Masked Transfer)

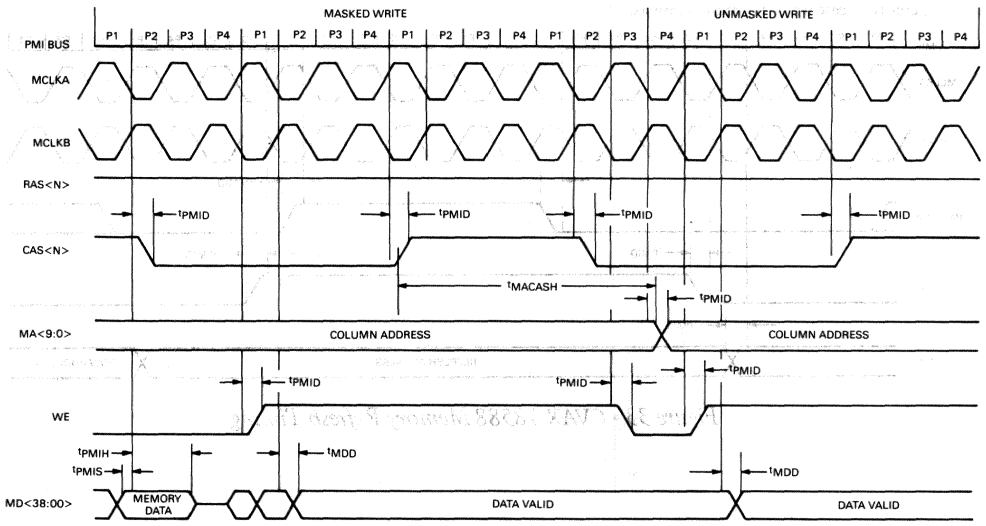


Figure 31 ▪ CVAX 78588 PMI Write Timing (Multiple Masked and Unmasked)

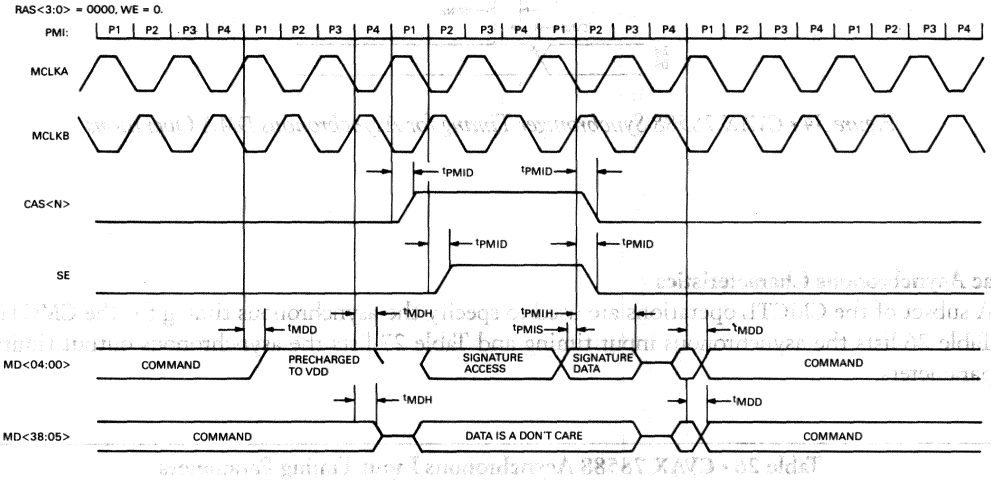


Figure 32 ▪ CVAX 78588 Read Signature Timing

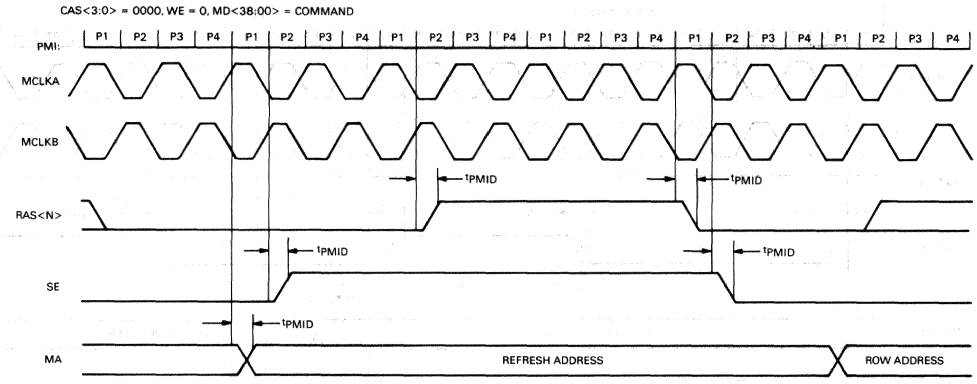


Figure 33 • CVAX 78588 Memory Refresh Timing

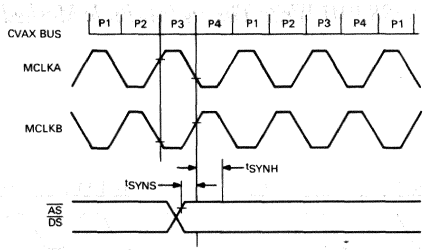


Figure 34 • CVAX 78588 Synchronizer Timing for Asynchronous DMA Operations

ac Asynchronous Characteristics

A subset of the CMCTL operations are used to specify the asynchronous timing for the CMCTL. Table 26 lists the asynchronous input timing and Table 27 lists the asynchronous output timing parameters.

Table 26 • CVAX 78588 Asynchronous Input Timing Parameters

Symbol	Parameter	Requirements (ns)[1] Min.	Max.
t_{ASDML}	$\overline{AS}$ to $\overline{DMG}$ lead time	0	—
t_{ASDSRL}	$\overline{AS}$ to $\overline{DS}$ read lead time	50	$4t_{CC}$[2]
t_{ASDSWL}	$\overline{AS}$ to $\overline{DS}$ write lead time	50	$2t_{CC}$[3]
t_{ASHW}	$\overline{AS}$ high width	100	—
t_{ASLW}[4]	$\overline{AS}$ low width	$3.5t_{CC}$	4000

Symbol	Parameter	Requirements (ns)[1] Min.	Max.
t_{ASBWH}	$\overline{AS}$ to $\overline{BM<3:0>}$ and $\overline{WR}$ hold	0	—
t_{ASIS}	Input to $\overline{AS}$ setup	20	—
t_{ASCS3S}	$\overline{CS/DP3}$ to $\overline{AS}$ setup	40	—
t_{ASIH}	Input to $\overline{AS}$ hold	20	—
t_{DMASL}	$\overline{DMG}$ to $\overline{AS}$ lead time	100	—
t_{DSASL}	$\overline{DS}$ to $\overline{AS}$ lead time	0	—
t_{DSDD}	$\overline{DS}$ to data delay	—	5.0
t_{DSDH}	$\overline{DS}$ to data hold	0	—
t_{DSHW1}	$\overline{DS}$ high width	175	—
t_{DSHW2}	$\overline{DS}$ high width	20	—
t_{DSLW}[5]	$\overline{DS}$ low width	—	4000

[1]t = 50 ns.
[2]The lead time that results in an $\overline{AS}$ memory read access relative to $\overline{DS}$ without synchronization slip cycles.
[3]The lead time that results in an $\overline{AS}$ memory write relative to the $\overline{DS}$ signal without synchronization slip cycles.
[4]The minimum occurs during an address miss and results in assertion of the $\overline{NLMR}$ signal for a minimum of 25 ns ($0.5t_{CC}$). The maximum ensures proper refresh timing for dynamic RAMs
[5]When the $\overline{ERR}$ signal is asserted, the $\overline{DS}$ signal must remain asserted for 100 ns ($2t_{CC}$) before it is deasserted.

Table 27 • CVAX 78588 Asynchronous Output Timing Parameters

Symbol	Parameter	Requirements (ns)[1] Min.	Max.	Error/ Condition[2]	PMI cycle
t_{ASERH}	$\overline{AS}$ to $\overline{ERR}$ and $\overline{RDY}$ hold	0	25		
t_{ASNLD}	$\overline{AS}$ to $\overline{NLMR}$ low delay	$2.5t_{CC}$	$24.5t_{CC}$		
t_{ASNHD}	$\overline{AS}$ to $\overline{NLMR}$ high delay	—	25		
t_{DSERD}	$\overline{DS}$ to $\overline{ERR}$ and $\overline{RDY}$ delay	0	25		
t_{OLW}	Output low width	100	—		
t_{RASOD}	Read $\overline{AS}$ to output delay	$6.5t_{CC}$	$8.5t_{CC}$	N	4/2[3,4]
		$8.5t_{CC}$	$10.5t_{CC}$	C	4/2
		$8.5t_{CC}$	$10.5t_{CC}$	U	4/2
		$8.5t_{CC}$	$10.5t_{CC}$	N	5/3
		$10.5t_{CC}$	$12.5t_{CC}$	C	5/3[10]
		$10.5t_{CC}$	$12.5t_{CC}$	U	5/3[10]

Symbol	Parameter	Requirements (ns)[1] Min.	Max.	Error/ Condition[2]	PMI cycle
t_{RDSOD}	Read $\overline{DS}$ to output delay	$2.5t_{CC}$	$4.5t_{CC}$	N	4/2[1-4]
		$4.5t_{CC}$	$6.5t_{CC}$	C	4/2[7]
		$4.5t_{CC}$	$6.5t_{CC}$	U	4/2
		$2.5t_{CC}$	$4.5t_{CC}$	N	5/3[7,10]
		$4.5t_{CC}$	$6.5t_{CC}$	C	5/3[7,10]
		$4.5t_{CC}$	$6.5t_{CC}$	U	5/3[10]
t_{WASOD}	Write $\overline{AS}$ to output delay	$4.5t_{CC}$	$6.5t_{CC}$	unmasked	[6]
		$8.5t_{CC}$	$10.5t_{CC}$	masked	4/2[9]
		$10.5t_{CC}$	$12.5t_{CC}$	masked	5/3[10]
t_{WDSOD}	Write $\overline{DS}$ to output delay	$2.5t_{CC}$	$4.5t_{CC}$	unmasked	[5,8,11]
		$2.5t_{CC}$	$4.5t_{CC}$	masked 1st transfer	[9,11]
		$8.5t_{CC}$	$10.5t_{CC}$	masked 2nd transfer	4/2[5,11]
		$10.5t_{CC}$	$12.5t_{CC}$	masked 2nd second transfer	5/3[5,11]

[1] t = 50 ns.

[2] N = none, C = correctable, U = uncorrectable

[3] During the first transfer of a read (no lock), the output delay is determined by $\overline{DS}$ when t_{ASDSRL} is greater than $4t_{CC}$, and the output delay is determined by $\overline{AS}$ when t_{ASDSRL} is less than or equal to $4t_{CC}$.

[4] During the first transfer of a read (lock), the output delay is determined by $\overline{DS}$ when t_{ASDSRL} is greater than $6t_{CC}$ and the output delay is determined by $\overline{AS}$ when t_{ASDSRL} is less than or equal to $6t_{CC}$. Add $2t_{CC}$ ns to t_{ASOD} for a read (lock).

[5] During the second, third, or fourth transfer of a multiple transfer operation, the output delay is determined by $\overline{DS}$.

[6] During multiple transfer reads, the specified t_{RDSOD} values are used if the delay from the previous $\overline{RDY}$ assertion to $\overline{DS}$ assertion is greater than $1.5t_{CC}$. If the delay is less than or equal to $1.5t_{CC}$, then the delay from the previous $\overline{RDY}$ to the next $\overline{RDY}$ is equal to $4t_{CC}$.

[7] During multiple transfer reads, the specified t_{RDSOD} values are used if the delay from the previous $\overline{RDY}$ assertion to $\overline{DS}$ assertion is greater than $3.5t_{CC}$. If the delay is less than or equal to $3.5t_{CC}$, then the delay from the previous $\overline{RDY}$ to the following $\overline{RDY}$ is equal to $6t_{CC}$.

[8] During the first transfer of an unmasked write, the output delay is determined by $\overline{DS}$ when t_{ASDSWL} is greater than $2t_{CC}$. The output delay is determined by $\overline{AS}$ when t_{ASDSWL} is less than or equal to $2t_{CC}$.

[9] During the first transfer of a masked write, the output delay is determined by $\overline{DS}$ when t_{ASDSWL} is greater than $6t_{CC}$. The output delay is determined by $\overline{AS}$ when t_{ASDSWL} is less than or equal to $6t_{CC}$.

[10] Add $2t_{CC}$ to the $\overline{AS}$ to $\overline{DS}$ delays in notes [3,4,9] when PMI cycle select bit is set.

[11] Add $2t_{CC}$ to t_{WDSOD} on a multiple transfer write if a masked write is followed by an unmasked write or if an unmasked write is followed by a masked write.

Asynchronous DMA Timing

Figures 35 through 37 show the asynchronous timing for a direct memory access (DMA) transfer with no errors detected. The timing of the $\overline{DMG}$ and $\overline{AS}$ signals is the same as for a read and a write operation.

During the memory read operation of the single transfer memory or CSR read transfer, Figure 36, a single transfer and no read lock is specified and the PMI cycle is 4/2. No memory errors occur.

During the asynchronous quadword unmasked write opeation, Figure 37, if a parity error is detected the $\overline{ERR}$ signal would be asserted instead of the $\overline{RDY}$ signal and at the same. When the $\overline{ERR}$ signal is asserted, the $\overline{DS}$ signal must remain asserted for an additional cycle before it can be deasserted to start the next data transfer. The timing of $\overline{ERR}$ is the same as an asynchronous read operation. During masked write operations, the type of error does not affect the assertion time of $\overline{RDY}$ or $\overline{ERR}$. These signals are asserted at the same time when no error is detected.

Figure 38 shows the timing of the $\overline{NLMR}$ output during a read or write operation where the address on CDAL<31:00> is not within the programmed range of the CMCTL.

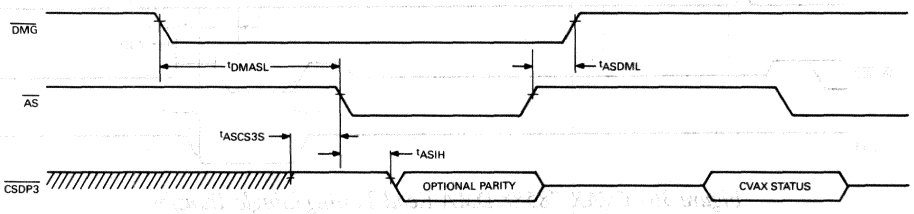


Figure 35 • CVAX 78588 DMA Transfer Timing ($\overline{DMG}$ and $\overline{AS}$ Signals)

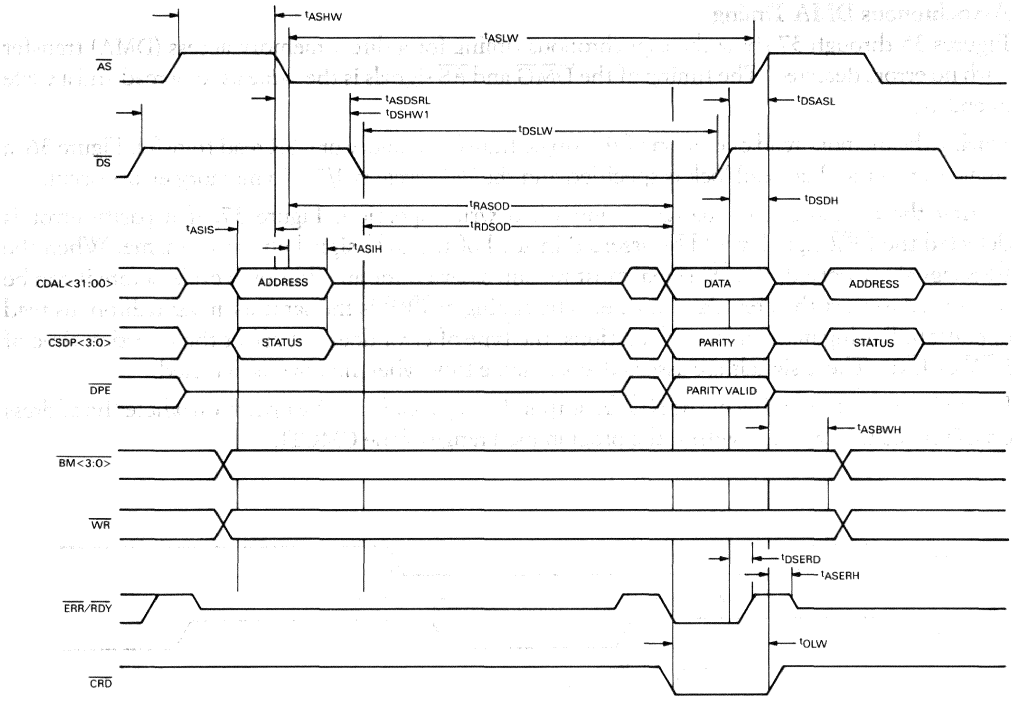


Figure 36 • CVAX 78588 DMA Read Timing (Single Transfer)

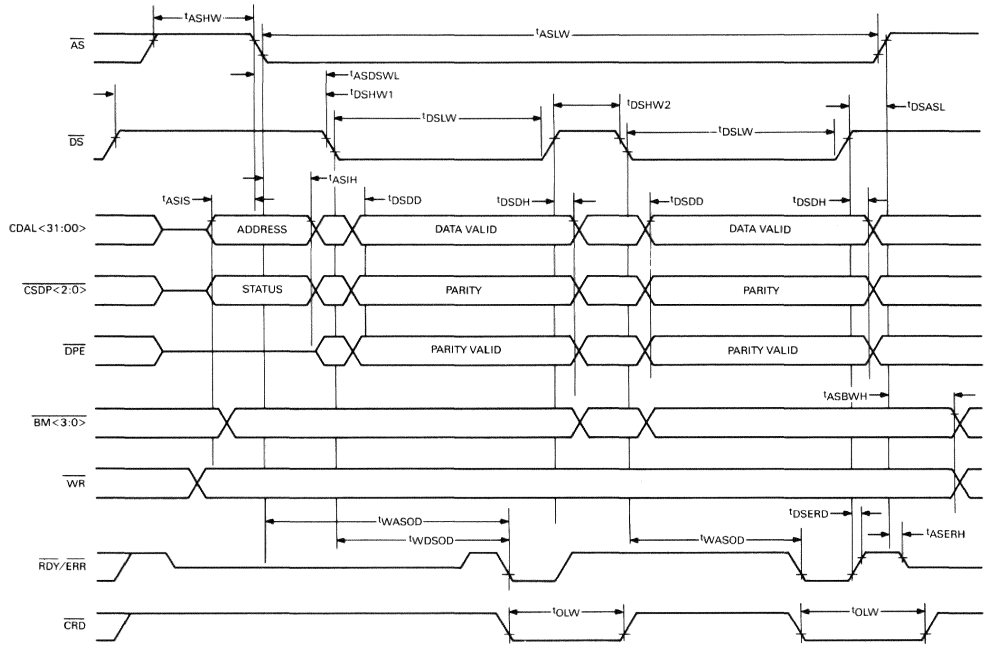


Figure 37 • CVAX 78588 DMA Write Timing

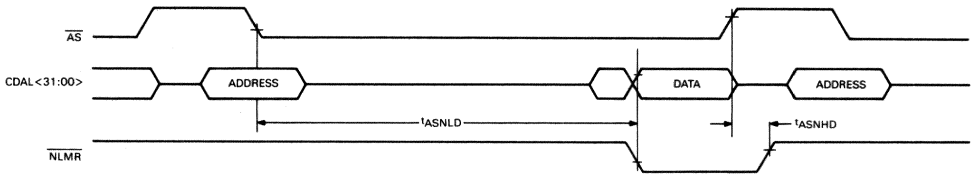
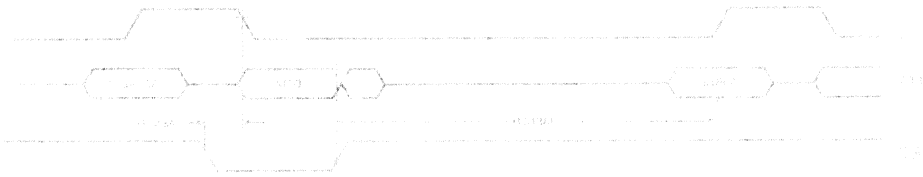


Figure 38 • CVAX 78588 DMA Address Miss Timing (NOP)

Mechanical Configuration

The physical dimensions of the CVAX 78588 132-pin package are contained in the Appendix.



Features

- 32-bit CVAX bus to 16-bit Q22-bus interface
- Integral Q22-bus transceivers
- 16-entry cached copy of the external 8 K longword scatter and gather map
- Performs scatter and gather operations and map control and address translation
- Powerup, initialization, powerfail, and powerdown control
- Integral address decoding for internal registers, scatter and gather map locotions, and Q22-bus references
- MicroVAX II-compatible doorbell register
- Two Q22-bus octaword write buffers and a quadword read buffer
- Transparent alignment of 32-bit and 16-bit data transactions
- Longword, quadword, hexaword, and octaword CVAX bus DMA transactions
- Q22-bus nonblock and block-mode transaction support
- Q22-bus arbiter or auxiliary mode operation
- Single 5-volt power supply

Description

The CVAX 78711 Q22-bus Interface Chip (CQBIC) is an asynchronous interface adapter for use between the 32-bit CVAX 78034 CPU and its internal memory and the 16-bit Q22-bus. Figure 1 is a block diagram of the CVAX 78711 CQBIC.

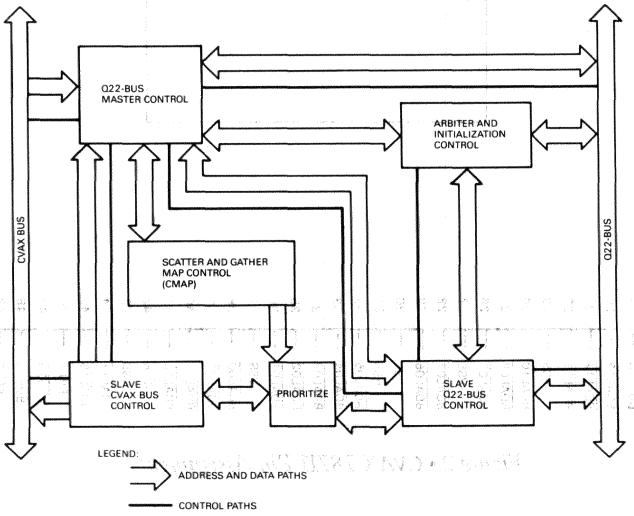


Figure 1 • CVAX 78711 Q22-bus Interface Block Diagram

The CQBIC performs the necessary address mapping and 32-bit and 16-bit data alignment. The CQBIC can function as the Q22-bus arbiter or as an auxiliary device. The CQBIC contains scatter and gather map translation and control logic, a 16-entry cache of the external 8 K longword mapping registers, a system configuration register, a DMA error register with master and slave address error registers, a MicroVAX II-compatible doorbell register and Q22-bus transceivers. The CQBIC uses a single 5-volt power supply, is available in a 132-pin surfacemount ceramic package, and dissipates less than 1.5 watts of power.

• Pin and Signal Descriptions

This section provides a description of the input and output signals and power and ground connections used by the CQBIC. The signal pin assignments are shown in Figure 2 and summarized in Table 1.

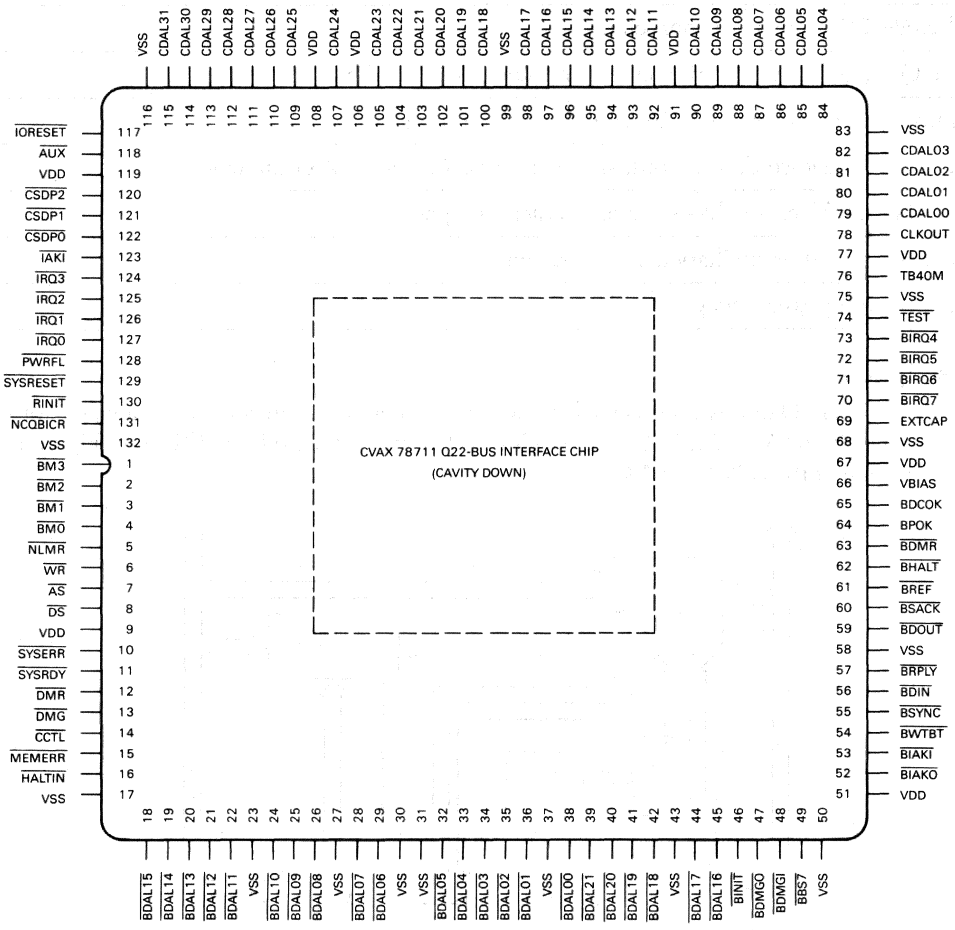


Figure 2 • CVAX 78711 Pin Assignments

Table 1 CVAX 78711 Pin and Signal Summary

Pin	Signal	Input/Output	Definition/Function
18-22 24-26,28, 29,32-36, 38-42,44,45	$\overline{\text{BDAL<21:00>}}$	Input/Output	Q22-bus Data/Address Lines—Time multiplexed, bidirectional data and address lines.
63	$\overline{\text{BDMR}}$	Input/Output	Q22-bus DMA Request—Requests bus mastership for DMA transfers.
62	$\overline{\text{BHALT}}$	Input	Q22-bus Halt—A CPU halt request by way of the $\overline{\text{HALTIN}}$ signal.
61	$\overline{\text{BREF}}$	Input/Output	Q22-bus reference—A strobe used to coordinate block-mode transfers.
65	BDCOK	Input	Q22-bus DC OK—Indicates that the system dc power is stable.
64	BPOK	Input	Power OK—Indicates that the primary ac power to the system power supply is normal.
60	$\overline{\text{BSACK}}$	Input/Output	Q22-bus Slave Acknowledge—Indicates that a DMA device is bus master.
59	$\overline{\text{BDOUT}}$	Input/Output	Q22-bus Data Output Strobe—Indicates a data output transfer with respect to the bus master and valid data is on $\overline{\text{BDAL<15:00>}}$.
57	$\overline{\text{BRPLY}}$	Input/Output	Q22-bus Reply—A strobe to indicate that the slave device has transferred the requested data to the bus or has accepted data from the bus.
56	$\overline{\text{BDIN}}$	Input/Output	Q22-bus Data Input—A strobe to indicate that an input transfer with respect to the bus master is in process or that an interrupt operation is taking place.
55	$\overline{\text{BSYNC}}$	Input/Output	Q22-bus Synchronizer—Indicates the start of a bus transfer and a valid address is on $\overline{\text{BDAL<21:00>}}$.
54	$\overline{\text{BWTBT}}$	Input/Output	Q22-bus Write and Byte Select—Provides bus cycle control. During the address portion of a bus cycle, it indicates an output cycle. During the data portion of a bus cycle, it indicates a byte transfer.
53	$\overline{\text{BIAKI}}$	Input	Q22-bus Interrupt Acknowledge In—Interrupt acknowledge daisychain input.
52	$\overline{\text{BIAKO}}$	Output	Q22-bus Interrupt Acknowledge Out—Interrupt acknowledge daisychain output.

Pin	Signal	Input/Output	Definition/Function
70-73	$\overline{\text{BIRQ}}$<7:4>	Input	Q22-bus Interrupt Request lines—Interrupt request lines for Q22-bus devices.
49	$\overline{\text{BBS7}}$	Input/Output	Q22-bus Bank 7 Select—Indicates an I/O page reference or block-mode transfer.
48	$\overline{\text{BDMGI}}$	Input/Output	Q22-bus DMA Grant In—The DMA grant daisychain input.
47	$\overline{\text{BDMGO}}$	Input/Output	Q22-bus DMA Grant Out—The DMA grant daisychain output.
46	$\overline{\text{BINIT}}$	Input/Output	Q22-bus Initialize—Q22-bus reset signal.
79-82, 84-90, 92-98, 100-107, 109-115	CDAL<31:00>	Input/Output	CVAX Data/Address lines—Time multiplexed, bidirectional data and address bus.
120-122	CSDP<2:0>	Input/Output	Control Status/Data Parity—Provide status information about bus cycle.
7	$\overline{\text{AS}}$	Input/Output	CVAX Address Strobe—CVAX system address strobe.
8	$\overline{\text{DS}}$	Input/Output	CVAX Data Strobe—CVAX system data strobe.
1-4	$\overline{\text{BM}}$<3:0>	Input/Output	CVAX Byte Masks—Identify the bytes of the CVAX bus and parity bits that are valid.
6	$\overline{\text{WR}}$	Input/Output	CVAX Write—Provides read and write control for the bus.
11	$\overline{\text{SYSRDY}}$	Input/Output	CVAX System Ready—Provides normal termination of the current bus cycle. Used with the $\overline{\text{SYSERR}}$ signal to request a retry of the current bus cycle.
10	$\overline{\text{SYSERR}}$	Input/Output	CVAX System Error—Provides abnormal termination of the current bus cycle in the event of an error. Used with $\overline{\text{SYSRDY}}$ to request a retry of the current bus cycle.
12	$\overline{\text{DMR}}$	Output	CVAX DMA Request—Requests the bus for DMA transfers.
13	$\overline{\text{DMG}}$	Input	CVAX DMA Grant—Grants the bus for a DMA transfer.
124-127	$\overline{\text{IRQ}}$<3:0>	Output	CVAX Interrupt Request lines—These lines are used to pass interrupt requests to the CVAX CPU.

Pin	Signal	Input/Output	Definition/Function
15	$\overline{\text{MEMERR}}$	Output	CVAX Memory Error—CQBIC requests an interrupt for a nonexistent memory error.
128	$\overline{\text{PWRFL}}$	Output	CVAX Powerfail—Indicates a powerfail condition on the Q22-bus.
14	$\overline{\text{CCTL}}$	Output	CVAX Cache Control—Provides the means to invalidate CVAX cache entries when the CQBIC accesses local memory.
129	$\overline{\text{SYSRESET}}$	Output	System Reset—Initializes CVAX CPU during the powerup sequence.
16	$\overline{\text{HALTIN}}$	Output	Halt—Halts the CVAX CPU.
118	$\overline{\text{AUX}}$	Input	Auxiliary—Selects the operating mode of the CQBIC.
117	$\overline{\text{IORESET}}$	Input	I/O Reset—Resets devices on the CVAX bus and Q22-bus.
131	$\overline{\text{NCQBICR}}$	Output	Not CQBIC Reference—Indicates that the address on the CVAX bus is not for the CQBIC.
123	$\overline{\text{IAKI}}$	Input	Interrupt Acknowledge In—Enables the CQBIC to respond to an interrupt acknowledge cycle on the CVAX bus.
5	$\overline{\text{NLMR}}$	Input	Not Local Memory Reference—Indicates that the address on the CVAX bus is not a local memory address.
130	$\overline{\text{RINIT}}$	Output	Receive Initialize—Initializes the CVAX bus and brings system to a predetermined state.
76	TB40M	Input	40-MHz Clock—The 40-MHz TTL clock input.
78	CLKOUT	Output	Clock Out—This output is used for test during manufacturing of the CQBIC.
9,51,67, 91,106, 108,119	V_{DD}	Input	Voltage—Power supply voltage.
17,23,27, 31,32,37, 43,50,58, 68,75,83, 99,116,132	V_{SS}	Input	Ground reference.
66	V_{BIAS}	Input	Q22-bus Bias—Provides the bias voltage for Q22-bus transceivers on the CQBIC.

Pin	Signal	Input/Output	Definition/Function
69	EXTCAP	Input	External Capacitor—Provides a 100 ms delay of the BDCOK input when 1.0 μF external capacitor is connected to this pin.
74	TEST	Input	Test—Used for test during manufacturing of the CQBIC.

Q22-Bus Signals

Q22-Bus Data Address Lines ($\overline{BDAL<21:18>}$)—These lines are used to transfer address information between the CQBIC and the Q22-bus.

Q22-Bus Data Address Lines ($\overline{BDAL<17:00>}$)—These time-multiplexed lines are used to transfer address, data, and parity control information between the CQBIC and the Q22-bus. During address protocol, $\overline{BDAL<17:01>}$ transfer address information and $\overline{BDAL00}$ specifies a high or low byte during DATOB and DATIOB cycles. During data protocol, $\overline{BDAL<17:16>}$ transfer parity control information, and $\overline{BDAL<15:00>}$ transfer data.

Direct Memory Access Request ($\overline{BDMR}$)—This line is asserted by the CQBIC or another device on the Q22-bus to request bus mastership.

Processor Halt ($\overline{BHALT}$)—This signal, when asserted, requests a CPU Halt through the $\overline{HALTIN}$ output. The CQBIC asserts the $\overline{HALTIN}$ signal if it is enabled by the SCR register.

Block Mode Reference ($\overline{BREF}$)—This signal is asserted or deasserted with the $\overline{BRPLY}$ signal by block mode slave devices to indicate to the bus master that the slave can accept another block mode data in (DIN) or data out (DOUT) transfer.

DC Power OK (BDCOK)—This is a power supply generated signal that is asserted when sufficient dc voltage is available to allow reliable system operation. It used as part of the powerup and powerdown protocol and boot protocol.

AC Power OK (BPOK)—This signal is asserted by the power supply when the primary ac power is normal. If the signal is deasserted during processor operation, a powerfail trap is initiated. This is part of the powerup and powerdown protocol.

Slave Acknowledge ($\overline{BSACK}$)—This signal is asserted by a DMA device when it becomes Q22-bus master. The device will assert this signal while it is bus master. When the device has completed using the bus, it deasserts $\overline{BSACK}$.

Data Output ($\overline{BDOUT}$)—This signal is asserted by the Q22-bus master to indicate that an output transfer, with respect to the bus master, is in progress and that valid data is on $\overline{BDAL<15:00>}$. This signal is deskewed with respect to the data placed on the bus.

Reply ($\overline{BRPLY}$)—This signal is asserted by a Q22-bus slave device in response to the assertion of the $\overline{BDOUT}$ or $\overline{BDIN}$ signal or by a device responding to an interrupt acknowledge (IAK) transfer. When $\overline{BRPLY}$ is asserted in response to the $\overline{BDOUT}$ signal, the slave device has read the data from $\overline{BDAL<15:00>}$. When it is asserted in response to the $\overline{BDIN}$ signal, the slave device has placed the requested data on $\overline{BDAL<15:00>}$. When $\overline{BRPLY}$ is asserted during an IAK transfer, the device responding to the interrupt has placed a vector on $\overline{BDAL<15:00>}$.

Data Input ($\overline{BDIN}$)—This signal is asserted during the time that $\overline{BSYNC}$ is asserted to indicate that the current bus master requires a response from the addressed slave device. It is also asserted by the interrupt fielding processor and is followed by the assertion of the $\overline{BIACK}$ output to initiate the interrupt service.

Synchronize (BSYNC)—This signal is asserted by the Q22-bus master to start a bus transfer and to indicate that it has transferred an address onto the BDAL<22:00> lines. The transfer continues until this signal is deasserted. For blockmode transfers, the BSYNC signal is asserted until the last transfer cycle is complete.

Write Byte (BWTBT)—This signal is used to control a bus cycle. It is asserted during the address portion of a bus cycle to indicate that a DATO, DATOB, or DATBO output cycle is to follow instead of an input cycle. It is also asserted during the data portion of a DATOB or DATBO cycle to indicate that a byte transfer will follow instead of a word transfer.

Interrupt Acknowledge In (BIAKI) and Interrupt Acknowledge Out (BIAKO)—The processor asserts the BIAKO input to acknowledge an interrupt request according to the interrupt protocol. The bus transmits the status of this signal to the BIAKI input of the next priority device electrically closest to the processor. This device accepts the interrupt acknowledge, if it had previously requested the bus, by asserting one of the interrupt request lines (BIRQ<7:4>) or if it is assigned the highest priority interrupt request on the bus when the BDIN signal was previously asserted. If both of these conditions do not exist, the device asserts the BIAKO output to the next device on the bus. This process continues in a daisychain configuration until the device with the highest interrupt priority receives the BIAKI interrupt acknowledge signal and continues the interrupt protocol sequence.

Q22-Bus Interrupt Request (BIRQ<7:4>)—These signals are asserted by devices on the Q22-bus to request an interrupt. If the CQBIC is in arbiter mode, the CQBIC will transfer these signals to the appropriate CVAX IRQ<3:0> line as indicated in Table 2.

Table 2 • CQBIC 78711 BIRQ to IRQ Mapping

BIRQ line	IRQ line
BIRQ7	IRQ3
BIRQ6	IRQ2
BIRQ5	IRQ1
BIRQ4	IRQ0

Bank 7 Select (BBS7)—This signal is asserted by the bus master to reference the I/O page including the part of the I/O page reserved for nonexistent memory. When BBS7 is asserted, the address on BDAL<12:00> is the I/O page address. During DATBI transfers, the bus master asserts this signal to indicate to the block-mode slave device that subsequent transfers will occur.

DMA Grant In (BDMGI) and DMA Grant Out (BDMGO)—The bus arbiter asserts these signal to grant bus mastership to a device that had requested use of the bus by asserting the BDMR signal. The arbiter grants the use of the bus by asserting the BDMGO signal. This signal is passed in the daisychain configuration through the bus to the BDMGI input of the next priority device electrically closest to the bus. This device accepts the grant if it had asserted the BDMR signal. If it did not, the device passes the DMA grant to the next device on the bus by asserting BDMGO. This sequence continues until the BDMGI input of the requesting device is asserted. The device stops the DMA grant by not asserting its BDMGO output and acknowledges the grant by asserting the BSACK signal after both the BRPLY and BSYNC signals are deasserted.

Initialize (BINIT)—This signal is used to reset the system. When asserted, all the devices on the bus are set to a known state. The initialize process includes clearing registers, disabling bus drivers, and setting the internal logic for an operation. Exceptions to the normal initialization must be documented in programming engineering specifications for the device.

CVAX Bus and System Control

CVAX Data/Address Lines (CDAL<31:00>)—These are bidirectional time-multiplexed lines used to transfer information between the CQBIC and the CVAX CPU or local memory.

During the first part of a read or write bus cycle, CDAL<31:30> indicates the length of the memory operand as listed in Table 3.

Table 3 • CQBIC 78711 Memory Operand Length

CDAL line 31	30	Operand length
0	0	hexaword
0	1	longword
1	0	quadword
1	1	octaword

CDAL<29:02> contain the longword address of the memory operand. The $\overline{BM<3:0>}$ lines specify which byte(s) of the longword address are to be used. The CDAL29 specifies a memory space address or an I/O space address. When CDAL29 is 0, memory space is specified, and when CDAL29 is 1, I/O space is specified. The CDAL<01:00>) are reserved.

During the first part of an interrupt acknowledge cycle, CDAL<06:02> transfer the hexadecimal number of interrupt priority level of the interrupt being acknowledged and CDAL<31:07> and DCAL<01:00> are zeros. During the second part of a read cycle, CDAL<31:00> transfers the incoming data. During the second part of an interrupt acknowledge cycle, CDAL<31:00> transfer the vector required by the CPU. During the second part of a write cycle, CDAL<31:00> transfer the outgoing information.

Control Status/Data Parity ($\overline{CSDP<2:0>}$)—These lines and the $\overline{WR}$ line provide the status of the current bus cycle as listed in Table 4. The $\overline{CSDP<2:0>}$ line information is valid when the $\overline{AS}$ line is asserted. The $\overline{CSDP<2:0>}$ lines are not used with the CQBIC which does not support CVAX bus parity.

Table 4 • CQBIC 78711 Cycle Status*

$\overline{WR}$	$\overline{CSDP}$ 2	1	0	Bus transaction	CQBIC Operation
H	L	L	L	request D-stream read	read
H	L	L	H	reserved	none
H	L	H	L	external IPR read	none
H	L	H	H	interrupt acknowledge	IAK
H	H	L	L	request read (I-stream)	read
H	H	L	H	demand read-lock	read-lock
H	H	H	L	demand read (D-stream, modify intent)	read
H	H	H	H	demand read (D-stream,no modify intent)	read

$\overline{WR}$	$\overline{CSDP}$ 2	1	0	Bus transaction	CQBIC Operation
L	L	L	L	reserved	none
L	L	L	H	reserved	none
L	L	H	L	external IPR write	none
L	L	H	H	reserved for DMA device	none
L	H	L	L	reserved	none
L	H	L	H	write-unlock	write-unlock
L	H	H	L	reserved	none
L	H	H	H	write (D-stream)	write-unlock

*H is a high level, L is a low level

The CQBIC ignores modify intent transactions and considers a write transaction as a write-unlock. When the CQBIC accesses to local memory, the $\overline{CSDP<2:0>}$ lines are set high for all read and write operations.

Address Strobe ($\overline{AS}$)—This bidirectional signal indicates that valid address information is present on the CVAX bus. The $\overline{AS}$ signal is asserted at the beginning of a bus cycle to indicate that valid address and control information is on CDAL<31:00>, $\overline{CSDP<2:0>}$, $\overline{BM<3:0>}$, and $\overline{WR}$ lines. $\overline{AS}$ is deasserted at the conclusion of the bus cycle.

Data Strobe ($\overline{DS}$)—This signal provides timing information for data transfers. During a read or interrupt acknowledge cycle, it is asserted to indicate that the CDAL<31:00> are available to receive incoming data and is deasserted to indicate that the data has been received and latched by the requesting device (CVAX CPU, CQBIC, etc.). During a write cycle, it is asserted to indicate that CDAL<31:00> contain valid outgoing data and is deasserted to indicate that the sending device will remove the data.

Byte Masks ($\overline{BM<3:0>}$)—These signals indicate which bytes of the CVAX bus contain valid data during the second part of a read or write bus cycle as defined in Table 5. During a read cycle, these signals indicate which bytes of the CVAX bus must transfer the data. All other bytes are ignored. During a write cycle, the $\overline{BM<3:0>}$ lines indicate which bytes of the CVAX bus contain valid data. During an interrupt acknowledge bus cycle, all four byte mask lines are asserted. The $\overline{BM<3:0>}$ lines are qualified by the assertion of the $\overline{AS}$ signal. For read bus cycles, all bits of the byte(s) specified by $\overline{BM<3:0>}$ must be set to high or low levels in accordance with the setup times defined in the *Specification* section.

Table 5 • CVAX 78711 Byte Mask Mapping

Byte mask asserted	Valid data
$\overline{BM3}$	CDAL<31:24>
$\overline{BM2}$	CDAL<23:16>
$\overline{BM1}$	CDAL<15:08>
$\overline{BM0}$	CDAL<07:00>

Write (WR)—This signal indicates the direction of the data transfer on the CVAX bus for the current bus cycle. When WR is asserted during a bus cycle, data is transferred to the CVAX bus by the originator of the bus cycle. When WR is deasserted during a bus cycle, the requested data is transferred to the bus by the responding device. WR is qualified by the assertion of the AS signal.

System Ready (SYSRDY)—This bidirectional signal indicates that the normal termination of the current CVAX bus cycle has occurred. It is also used with the SYSERR signal to request a retry of a bus cycle generated by the CVAX CPU. Assertion of this signal during a CVAX bus read or interrupt acknowledge bus cycle indicates that external logic will transfer the requested data onto the bus according to the timing requirements of the bus cycle in progress. Assertion of this signal during a CVAX bus write cycle indicates that external logic will receive the information on the bus according to the timing requirements of a write bus cycle.

System Error (SYSERR)—This bidirectional signal indicates the abnormal termination of the current bus cycle. It is also used with with the SYSRDY signal to request the retry of a bus cycle generated by the CVAX CPU.

DMA Request (DMR)—This signal is asserted by the CQBIC to request use of the CVAX bus and its related control signals.

DMA Grant (DMG)—This signal is asserted by the CPU to grant control of the CVAX bus and its related control signals to external logic. If the CQBIC has asserted the DMR signal, the CQBIC will respond to the assertion of DMG by assuming control of the CVAX bus and its control signals.

Interrupt Request (IRQ<3:0>)—These signals are used to pass interrupt requests to the CVAX CPU. The CQBIC uses IRQ0 to pass a Doorbell register interprocessor interrupt request to the local system. When the CQBIC is in arbiter mode, it maps the interrupt requests from the BIRQ<7:4> of the Q22-bus to IRQ<3:0> as listed in Table 6.

Table 6 • CVAX 78711 Interupt Request Mapping

Q22-bus request	CVAX request	Priority level (hexadecimal)
BIRQ7	IRQ3	IPL 17
BIRQ6	IRQ2	IPL 16
BIRQ5	IRQ1	IPL 15
BIRQ4	IRQ0	IPL 14

Memory Error (MEMERR)—This signal indicates that a nonexistent memory interrupt request to the CVAX CPU has occurred. The CQBIC asserts this signal with the assertion of the AS signal at the beginning of the next CVAX bus cycle. It remains asserted until the DS signal is deasserted at the end of the bus cycle.

Powerfail (PWRFL)—This signal indicates a powerfail condition on the Q22-bus. The CQBIC asserts this signal when the BPOK input is deasserted.

Cache Control (CCTL)—This signal is used to invalidate the cache memory in the CVAX CPU during write transactions to local memory. The CQBIC asserts this signal when it performs a write to local memory and causes the CVAX CPU to perform a cache invalidate cycle.

System Reset (SYSRESET)—This signal is used during the powerup sequence to initialize the CVAX CPU. It is asserted when the BDCOK input is deasserted.

Halt In (HALTIN)—This signal is asserted in response to the assertion of the $\overline{\text{BHALT}}$ input from the Q22-bus. It is controlled by bit 14 of the system configuration register, by bit 08 of the doorbell register and by the operating mode of the CQBIC.

CQBIC and System Control

Auxiliary Mode ($\overline{\text{AUX}}$)—This signal selects the operating mode of the arbitration logic of the CQBIC, the master logic, the powerup and reset sequences, and the doorbell register. The CQBIC samples this signal when the $\overline{\text{AS}}$ signal is asserted to determine its operating mode. When it is deasserted, the arbitration mode of the Q22-bus is selected. The CQBIC is then the arbiter for the Q22-bus. Only one CBIC can be bus arbiter when more than one CBIC are available. When this signal is asserted, auxiliary mode is selected and the CQBIC does not perform arbitration for Q22-bus.

I/O Reset ($\overline{\text{IORESET}}$)—This signal resets the devices on the CVAX bus and the Q22-bus as determined by CQBIC operating mode. When it is asserted, the CQBIC gains control of the CVAX bus and the Q22-bus by first asserting the $\overline{\text{DMR}}$ output and then the $\overline{\text{BDMR}}$ output. When the $\overline{\text{DMG}}$ and $\overline{\text{BDMGI}}$ signals are asserted, the CQBIC asserts the $\overline{\text{RINIT}}$ output to initialize the devices on the CVAX bus. If the CBIC is not in auxiliary mode ($\overline{\text{AUX}}$ deasserted), it asserts the $\overline{\text{RINIT}}$ output to initialize the Q22-bus.

Not CQBIC Reference ($\overline{\text{NCQBICR}}$)—This signal indicates that the address on the CVAX bus is not a CQBIC address. When the CVAX initiates a bus transaction that is not intended for the CQBIC or a bus transaction that is not an interrupt acknowledge cycle, the CQBIC asserts this signal when the $\overline{\text{DS}}$ signal is asserted by the CVAX.

Interrupt Acknowledge Enable ($\overline{\text{IAKI}}$)—This signal is asserted to allow the CQBIC to respond to interrupt acknowledge cycles on the CVAX bus. It is normally connected in a daisychain configuration with the CQBIC as the last device in the chain.

Not Local Memory Reference ($\overline{\text{NLMR}}$)—This signal indicates that the address on the CVAX bus is not a local memory address. When asserted, it notifies the CQBIC of an attempt to access nonexistent local memory from the Q22-bus.

Receive Initialize ($\overline{\text{RINIT}}$)—This signal is used to initialize devices on the CVAX bus and initialize the system to predetermined state. It is asserted during the powerup sequence or in response to the assertion of the $\overline{\text{IORESET}}$ input.

Clock Timing

40-MHz Clock (TB40M)—A 40-MHz clock input for the CQBIC timing. This input is divided by two (20 MHz) for use by the CQBIC. The timing sequence is started by the first rising edge of this input following the deassertion of $\overline{\text{SYSRESET}}$ input.

Clock Output (CLKOUT)—A 20-MHz clock output from the CQBIC. This signal is generated by the CQBIC and is used only during manufacturing test of the CQBIC.

Power and Ground

Voltage (V_{DD})—5-volt power supply input.

Ground (V_{SS})—Ground reference.

Voltage Bias (VBIAS)—The bias voltage for the Q22-bus transceivers. This pin should be connected to ground through a resistor with a tolerance of 1 percent. The value of the resistor is selected to provide a bias current of 300 microamperes.

Miscellaneous

External Capacitor (EXTCAP)—This pin connects to an external 1.0 µF capacitor to generate the 100-millisecond delay for BDCOK output during the powerup sequence.

Test ($\overline{\text{TEST}}$)—Used only for manufacturing test of the CQBIC.

Functional Description

The CQBIC provides the interface between the CVAX 78034 CPU and its local memory and the Q22-bus. The CQBIC can perform Q22-bus arbitration or can function as an auxiliary device on the Q22-bus. The CQBIC contains a 32-bit CVAX bus to 16-bit Q22-bus interface, scatter and gather map translation and control logic, a 16-entry cache register that contains the external 8 K longword mapping registers, a system configuration register, a DMA error register with master and slave address error registers, a MicroVAX II-compatible doorbell register and Q22-bus transceivers.

The CQBIC contains a master, slave, arbiter, and cache section. The master section monitors the CVAX bus addresses and control signals and performs operations directed to the internal registers of the CQBIC. It monitors the Q22-bus master transactions and requests assistance with scatter and gather map operations from the slave section. The slave section monitors the Q22-bus addresses and control signals and performs operations directed to the CQBIC from a Q22-bus master. It controls the doorbell register and performs transfers to and from local memory. The master and slave sections of the CQBIC use the cache registers to validate Q22-bus addresses that are to be mapped into local memory. The cache section uses the slave section to perform the local memory operations directed to the scatter and gather map registers in local memory. The arbiter section resolves conflicts between the master and slave sections that relate to the use of the CVAX bus, the Q22-bus, and Q22-bus DMA and interrupt acknowledge arbitration. It also controls the powerup and powerdown sequence and the initialization protocols.

The CQBIC supports master transactions that are byte, word, and longword transfers from the CVAX bus to the Q22-bus and to the internal registers of the CQBIC. It also supports slave write transactions that are byte and word write transfers, blockmode word write transfers from the Q22-bus to local memory on the CVAX bus, and slave read-modify-write word transactions that are block-mode word read transfers from Q22-bus to local memory. The CQBIC also supports local-miss and global-hit transactions from the CVAX CPU to Q22-bus memory space where the Q22-bus map translates the address back into the local memory space.

Q22-bus Interface

The Q22-bus interface supports nonblock-mode and block-mode transactions. As a Q22-bus slave, the CQBIC supports the following *DEC Standard 160* transactions: DATI, DATIB, DATO, DATOB, DATIO, DATIOB, DATBO, and DATBI. As Q22-bus master, the CQBIC supports the following transactions: DATI, DATIB, DATO, DATOB, DATBO, DATBI, and interrupt acknowledge (IAK) transactions. As Q22-bus master, the CQBIC supports Q22-bus master read parity errors and Q22-bus nonexistent memory timeouts.

When the CQBIC functions as a slave to a Q22-bus block-mode write transaction, it stores up to 16 words and maintains the address alignments. The contents of the storage buffer is then transferred to local memory in two octaword transfers. When a Q22-bus block-mode transfer is not 16-word aligned, the CQBIC stores up to the 16-word boundary, stops replying as a Q22-bus block-mode slave, and performs the transfer to local memory. The CQBIC controls address alignments of block-mode write transfers of two to eight words. When the CQBIC responds as a slave to a Q22-bus read transaction, it stores up to a quadword of data before the Q22-bus block-mode read transfer. The CQBIC performs a read prefetch operation from local memory when the third word of its internal quadword buffer is transferred to the Q22-bus block-mode master that is performing a DATBI transaction.

Address Decoding

The CQBIC performs all address decoding for its internal registers, Q22-bus memory and Q22-bus I/O address spaces, and for the external scatter and gather map registers. It also provides a not-addressed signal ($\overline{\text{NCQBICR}}$) for system use. The CQBIC latches the address from the CVAX bus with the assertion of the $\overline{\text{AS}}$ signal. In addition to the address, the CQBIC latches the $\overline{\text{CSDP}}<2:0>$, $\overline{\text{BM}}<3:0>$, and $\overline{\text{WR}}$ information to determine the type of transaction. The valid physical addresses that are decoded by the CQBIC are listed in Table 7. All other physical addresses cause the CQBIC to assert its $\overline{\text{NCQBICR}}$ signal.

Table 7 • CVAX 78711 CVAX Bus Physical Address Decodes

CVAX bus addresses	Address description
2000 0000 to 2000 1FFF	Q22-bus I/O space
2008 0000 to 2008 0010	CQBIC internal registers
2008 8000 to 2008 FFFC	Q22-bus scatter and gather map registers
3000 0000 to 303F FFFF	Q22-bus memory space

The doorbell register is located in Q22-bus I/O space. Its address is determined by the mode of operation of the CQBIC and bits 03:01 of the system configuration register.

Clock

The CQBIC requires an external 40-MHz TTL clock input that can be asynchronous or synchronous to the clock of the CVAX CPU. This fixed rate clock produces a two-phase internal 20-MHz clock and the required Q22-bus timing in accordance with *DEC Standard 160*. The CQBIC operates with a CVAX bus cycle time of 100 to 80 nanoseconds and a Q22-bus with fixed timing.

Registers

The CQBIC contains mapping registers, error registers, and a configuration register. It uses the doorbell register, located in the I/O page of the Q22-bus address space, for interprocessor communications.

Mapping Registers (MRA)—The Q22-bus scatter and gather map contains 8192 mapping registers. Each MRA maps a page (512 bytes) of Q22-bus address space into a selected page of local memory. The MRA format is shown in Figure 3 and described in Table 8.

Figure 3 • CVAX 78711 Mapping Register Format

Table 8 • CVAX 78711 Mapping Register Description

Bit	Description
31	V (Valid)—When set, this bit indicates that mapping is enabled to a page in Q22-bus address space specified by bits 19:00 of this register. When cleared, the mapping for the selected page in Q22-bus address space is disabled and the CQBIC does not respond.
30:20	MBZ (Must be zeros)—These bits are read as zeros.
19:00	A28 to A09 (Address bits 28:09)—This field contain the physical page address in local memory to which the Q22-bus address is mapped.

The mapping registers are located in the local processors I/O space at physical addresses 2008 8000 through 2008 FFFC (hexadecimal). Each MRA is located on a longword boundary and is byte addressable. The physical longword address of each register is such that bits 14:02 of the physical address are identical to bits 21:09 of the Q22-bus address. The actual location of the scatter and gather map in local memory is determined by the map base register that contain the starting address of an aligned 8 Kblock of local memory. Only the local processor can directly access these registers through the CQBIC. Table 9 shows the relationship of the mapping registers to the Q22-bus addresses.

Table 9 • CVAX 78711 CBIC to Q22-bus Address Mapping

Register address (hexadecimal)	Q22-bus address Mapped (hexadecimal)	Mapped (octal)
2008 8000	00 0000 – 00 01FF	00 000 000 – 00 000 777
2008 8004	00 0200 – 00 03FF	00 001 000 – 00 001 777
2008 8008	00 0400 – 00 05FF	00 002 000 – 00 002 777
2008 800C	00 0600 – 00 07FF	00 003 000 – 00 003 777
2008 8010	00 0800 – 00 09FF	00 004 000 – 00 004 777
2008 8014	00 0A00 – 00 0BFF	00 005 000 – 00 005 777
2008 8018	00 0C00 – 00 0DFF	00 006 000 – 00 006 777
2008 801C	00 0E00 – 00 0FFF	00 007 000 – 00 007 777
•	•	•
•	•	•
•	•	•
2008 FFF0	3F F800 – 3F F9FF	17 774 000 – 17 774 777
2008 FFF4	3F FA00 – 3F FBFF	17 775 000 – 17 775 777
2008 FFF8	3F FC00 – 3F FDFF	17 776 000 – 17 776 777
2008 FFFC	3F FE00 – 3F FFFF	17 776 000 – 17 777 777

Note

The system boot PROM must remove the local memory space used for the scatter and gather map registers from the bit map of good memory. Direct accesses to the local memory copy of the map by a device other than the CQBIC can result in nonvalid data in the cache of map registers of the CQBIC and the results are unpredictable.

At powerup time, the scatter and gather map registers and their valid bits are undefined. These registers are not altered by system or local resets.

Cached Map Registers (CMR)—The CQBIC maintains a 16-entry cache of map registers that performe all mapping functions. If the cache does not contain a valid copy of the map register used to map a Q22-bus address into local memory, the CQBIC obtains the required register information from the scatter and gather map. Only map registers that have their valid bit set are stored in the cache of the CQBIC. The CMR replacement algorithm is first-in/first-out (FIFO). The format of the CMR is shown in Figure 4 and described in Table 10.

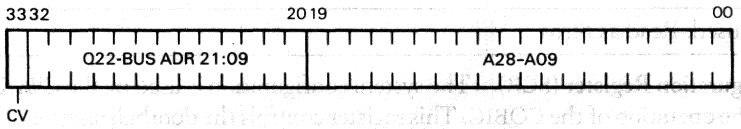


Figure 4 • CVAX 78711 Cached Map Register Format

Table 10 • CVAX 78711 Cached Map Register Description

Bit	Description
33	CV (CAM valid)—When set, this bit indicates that the CMR contains a valid copy of a map register, and mapping is enabled for the page in Q22-bus address space. When cleared, it indicates that the contents of the CMR are not valid and mapping is disabled. The CQBIC must update the cache from the scatter and gather map to determine if mapping is enabled for the Q22-bus address to be mapped.
32:20	Q22-BUS ADR 21:09 (Q22-bus address 21:09)—This field contains the address of the page in Q22-bus address space mapped by the map register address bits stored in bits 19:00 of this register.
19:00	A28-A09 (Address bits 28:09)—This field contains the address of the page in local memory that the associated Q22-bus address is mapped to.

Map Base register—This longword accessible register is located at physical address 2008 0010 (hexadecimal) and contains the starting address of the scatter and gather map in local memory. This address must be located on an aligned 8 K longword block of local memory. The system boot PROM must indicate when this block of memory is unavailable. The only access to the scatter and gather map should be through the CQBIC. A write operation to the map base register clears all the CAM valid bits in the CMR. This removes the cached copy of the map registers when changing the location of the scatter and gather map in local memory. The format of the map base register is shown in Figure 5 and is described in Table 11.

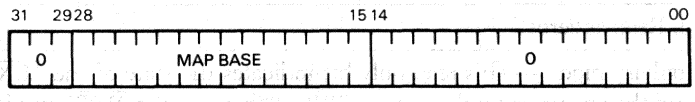


Figure 5 • *CVAX 78711 Map Base Register Format*

Table 11 • CVAX 78711 Map Base Register Description

Bit	Description
31:29	Not used. Read as zeros.
28:15	MAP BASE (Map base address)—These bits are used as physical address bits 28:15 when the CQBIC accesses the scatter and gather map. Bits 14:02 of the map register address are used as physical address bits 14:02. Bits 01:00 are zeros.
14:00	Not used. Read as zeros.

System Configuration Register (SCR)—The system configuration is used by the CVAX 78034 CPU to configure the operation of the CQBIC. This register controls the doorbell register offset address, the enabling and disabling of the Q22-bus $\overline{\text{BHALT}}$ signal and the Power OK (POK) and AUX flag. The SCR is located at CVAX physical address 2008 0000 (hexadecimal). The format of this register is shown in Figure 6 and described in Table 12. The SCR is cleared during the powerup sequence or when the $\overline{\text{SYSRESET}}$ signal is asserted. This register is not affected by a processor programmed reset.

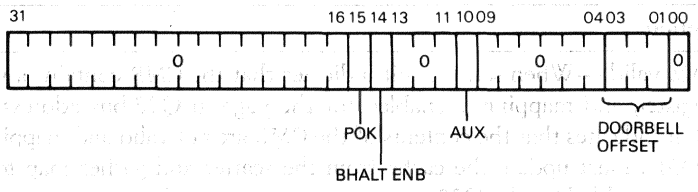


Figure 6 • CVAX 78711 System Configuration Register Format

Table 12 • CVAX 78711 System Configuration Register Description

Bit	Description
31:16	Not used. Read as zeros.
15	POK (Power OK)—This is a read-only bit that indicates the the state of the BPOK signal on the Q22-bus. It is synchronized and latched at each assertion of the $\overline{\text{AS}}$ signal on the CVAX bus and is set to indicate that the power is OK and normal operation is possible. It is cleared to indicate that the power supply has failed or that the supply voltage is below normal.
14	BHALT ENB ($\overline{\text{BHALT}}$ Enable)—This read/write bit is used to enable the transfer of $\overline{\text{BHALT}}$ signal from the Q22-bus to the HALTIN output of the CQBIC. When set, the state of $\overline{\text{BHALT}}$ is transferred to $\overline{\text{HALTIN}}$. When cleared, $\overline{\text{BHALT}}$ has no effect on the $\overline{\text{HALTIN}}$ output.
13:11	Not used. Read as zeros.
10	AUX (Auxiliary mode)—This read-only bit indicates the state of the $\overline{\text{AUX}}$ input to the CQBIC and the operating mode of the CQBIC. When set, the $\overline{\text{AUX}}$ input is asserted and auxiliary mode is selected. When cleared, the $\overline{\text{AUX}}$ input is negated and arbiter mode is selected.

Bit	Description
09:04	Not used. Read as zeros.
03:01	Doorbell—These read/write bits select the auxiliary doorbell register when the $\overline{AUX}$ input of the CQBIC is asserted and auxiliary mode is selected. When $\overline{AUX}$ is not asserted, these bits have no effect on the doorbell register used.
00	Not used. Read as 0.

DMA System Error Register (DSER)—This register is used to report DMA errors to the local system. It is located in the VAX I/O space at address 2008 0004 (hexadecimal). It can be accessed only by the local processor. Other processors on the Q22-bus cannot access this register. This register is cleared when the $\overline{RINIT}$ output is asserted. Individual bits may be cleared by writing a 1 to the respective bit. Writing a 0 to any bit has no effect on the register information. The format of the DSER is shown in Figure 7 and described in Table 13.

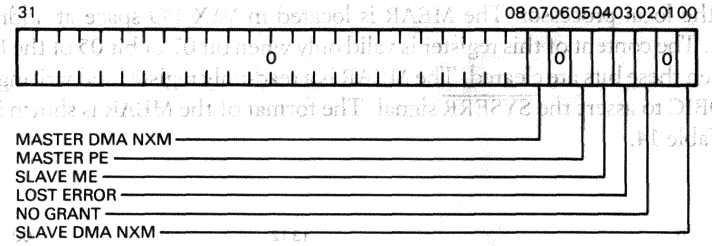


Figure 7 • CVAX 78711 DMA System Error Register Format

Table 13 • CVAX 78711 DMA System Error Register Descriptions

Bit	Description
31:08	Not used. Read as zeros.
07	MASTER DMA NXM (Master DMA nonexistent memory error)—This read/write bit is set for Q22-bus read or write cycles that do not assert the $\overline{BRPLY}$ signal in less than 10 microseconds. This bit is cleared by writing a 1 to it.
06	Not used. Read as 0.
05	MASTER PARITY ERROR—This read/write bit is set when parity errors are detected during Q22-bus read cycles performed by the CQBIC. This bit is cleared by writing a 1 to it.
04	SLAVE MEMORY ERROR—This bit is set when a local memory read access from the Q22-bus, Q22-bus device, or local-miss or global-hit cycles receives a memory error and asserts the $\overline{SYSERR}$ signal. This bit is cleared by writing a 1 to it.
03	LOST ERROR—This bit is set to indicate that an error address has been lost. An error address is lost when the DSER bits 07, DSER bits 05:04, and DSER bit 00 were previously set and another error that would have set one of these bits occurs. This bit is cleared by writing a 1 to it.

Bit	Description
02	NO GRANT TIMEOUT—This read/write bit is set if the Q22-bus does not return a bus grant within 10 milliseconds after a bus request, generated by a local processor demand read or write cycle, has occurred. This bit is cleared by writing a 1 to it.
01	Not used. Read as 0.
00	SLAVE DMA NXM (Slave DMA nonexistent memory—This read/write bit is set when the CQBIC executes a slave cycle that results in the CMCTL asserting the $\overline{\text{NLMR}}$ signal. This includes local-miss or global-hit cycles and map references that are nonexistent memory. This bit is cleared by writing a 1 to it.

Master Error Address Register (MEAR)—This register contains the address of the page in Q22-bus space that resulted in a parity error (DSER bit 05 set) or a bus timeout (DSER bit 07 set) during an access by the local processor. The MEAR is located in VAX I/O space at address 2008 0008 (hexadecimal). The content of this register is valid only when bit 07 or bit 05 of the DSER is set and undefined when these bits are cleared. The MEAR is a read-only register and writing to this register causes the CQBIC to assert the $\overline{\text{SYSERR}}$ signal. The format of the MEAR is shown in Figure 8 and described in Table 14.

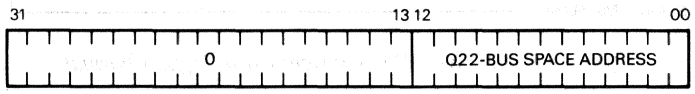


Figure 8 • CVAX 78711 Master Error Address Register Format

Table 14 • CVAX 78711 Master Error Address Register Description

Bit	Description
31:13	Not used. Read as zeros.
12:00	Q22-BUS SPACE ADDRESS—This field contains Q22-bus address bits 21:09.

Slave Error Address Register (SEAR)—This register contains the map translated address of the page in local memory that resulted in a memory error (DSER bit 04 set) or a nonexistent memory error (DSER bit 00 set) during an access by the local processor. The SEAR is located in VAX I/O space at address 2008 000C (hexadecimal). The content of this register is valid only when bit 04 or bit 00 of the DSER is set. The content is undefined when these bits are cleared. The SEAR is a read-only register and writing to this register causes the CQBIC to assert the $\overline{\text{SYSERR}}$ signal. The format of the SEAR is shown in Figure 9 and described in Table 15.

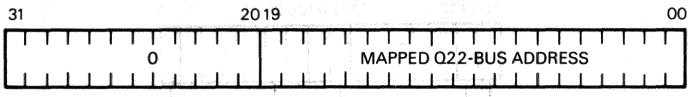


Figure 9 • CVAX 78711 Slave Error Address Register Format

Table 15 • CVAX 78711 Slave Error Address Register Description

Bit	Description
31:20	Not used. Read as zeros.
19:00	MAPPED Q22-BUS ADDRESS—This field contains the map translated address of the page in local memory that resulted in a memory error or nonexistent memory error from the Q22-bus to local memory.

Doorbell Register (DBR)—The DBR is used by the CQBIC interprocessor communication facility to allow other processors on the Q22-bus to request program interrupts from the local processor of the CQBIC without using the Q22-bus interrupt request lines ($\overline{\text{BIRQ}}$<7:4>). The DBR also controls external access from the Q22-bus to local memory and allows other processors to halt an auxiliary CPU. The DBR resides in the I/O Page of Q22-bus address space and is accessed by any device that can become Q22-bus master. The address of the DBR is determined by the arbiter or auxiliary mode of operation of the CQBIC and by bits 03:01 of the System Configuration Register (SCR). Table 16 lists the SCR bits 03:01 and the selected 32-bit address for the DBR. The format of the DBR is shown in Figure 10 and described in Table 17.

Table 16 • CVAX 78711 Doorbell Register Address Selection

SCR bits 03	02	01	Doorbell register	Address (hexadecimal)
0	0	0	arbiter CPU	2000 1F40
0	0	1	auxiliary no.1	2000 1F42
0	1	0	auxiliary no.2	2000 1F44
0	1	1	auxiliary no.3	2000 1F46
1	0	0	auxiliary no.4	2000 1F48
1	0	1	auxiliary no.5	2000 1F4A
1	1	0	auxiliary no.6	2000 1F4C
1	1	1	auxiliary no.7	2000 1F4E

When the CQBIC is in arbiter mode, SCR bits 03:01 are cleared by default and the arbiter CPU doorbell register address is selected. When the CQBIC is in auxiliary mode, clearing the SCR bits 03:01 will disable the doorbell register.

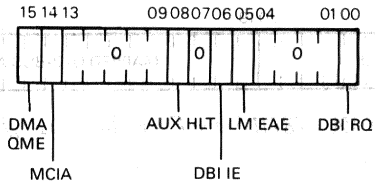


Figure 10 • CVAX 78711 Doorbell Register Format

Table 17 • CVAX 78711 Doorbell Register Description

Bit	Description
15	DMA QME (DMA Q-22 memory error)—This read-only bit is set to indicate an address space memory error when bit 04 of the DMA System Error Register (DSER) is set. It is cleared when bit 04 of DSER is cleared or when the $\overline{\text{RINIT}}$ output is asserted.
14	MCIA (Map cache invalidate all)—This write-only bit is used to invalidate the cache of map entries of the CQBIC. Writing a 1 to this bit clears the CAM valid bit 33 of the CMR for each entry. This bit is read as 0 and writing a 0 to this bit has no effect.
13:09	Not used. Read as zeros.
08	AUX HLT (Auxiliary halt)—This read/write bit is used when the CQBIC is in auxiliary mode. It is typically set by the arbiter CPU and causes the $\overline{\text{HALTIN}}$ output of the CQBIC to be asserted. This bit is cleared by writing a 0 to it or by the assertion of the $\overline{\text{RINIT}}$ output. The $\overline{\text{BHALT}}$ enable bit 14 of the system configuration register has no effect on this bit. When the CQBIC is in arbiter mode, this bit is read-only and is read as 0.
07	Not used. Read as 0.
06	DBI IE (Doorbell interrupt enable)—This bit is set to enable interprocessor doorbell interrupt requests through the DBR bit 00. It is cleared to disable interprocessor doorbell interrupt requests. It is a read/write bit when the CQBIC is Q22-bus master and a read-only bit when another device or CPU is Q22-bus master. It is cleared by the $\overline{\text{RINIT}}$ output.
05	LM EAE (Local memory external access enable)—This bit is set to enable access to local memory from the Q22-bus. This bit is cleared to disable access to local memory from the Q22-bus. It is a read/write bit when the CQBIC is Q22-bus master and a read-only bit when another device or CPU is Q22-bus master. It is cleared by the assertion of the $\overline{\text{RINIT}}$ output when the CQBIC is in auxiliary mode and by the assertion of the $\overline{\text{SYSRESET}}$ signal when the CQBIC is in arbiter mode.
04:01	Not used. Read as zeros.
00	DBI RQ (Doorbell interrupt request)—The function of this bit is enabled and disabled by bit 06 (DBI IE) of this register. When DBI IE is set, writing a 1 to DBI RQ causes the CQBIC to assert the $\overline{\text{IRQ0}}$ output to post an interrupt request to the local processor. When DBI IE is cleared, the CQBIC holds DBI RQ cleared and writing a 1 to this bit has no effect. Writing a 0 to this bit never has an effect. It is cleared by an IPL 14 interrupt acknowledge cycle to the CQBIC or by the assertion of the $\overline{\text{RINIT}}$ output.

Scatter and Gather Map Operation

The CQBIC uses a scatter and gather map to store Q22-bus addresses into local memory. The map consists of 8192 mapping registers stored externally in local memory, a map base register to record the location of the external map registers, and a 16-entry cache containing the most recently used map registers. Each map register selects a 512 byte page of Q22-bus address space in local memory. The map is enabled or disabled by the LM EAE bit 05 in the doorbell register of the CQBIC. All mapping is performed from the internal cache of map registers. Therefore, if the required map register is not present in the cache, the CQBIC updates the cache with the required map register from local memory and then continues the mapping operation. Through the use of a valid bit in each register, the software can selectively enable and disable the mapping of selected pages in Q22-bus address space.

The CQBIC monitors each Q22-bus cycle and responds if the LM EAE bit 05 in the doorbell register is set or if the Valid bit 31 of the selected mapping register is set. The LM EAE bit is ignored for a local miss or hit transaction. Only map registers that have their valid bit set are stored in the CQBIC cache.

During read operations, the mapping register must map the Q22-bus address into an existing local memory, or a bus timeout will occur. During write operations, the CQBIC asserts the $\overline{\text{BRPLY}}$ signal to the Q22-bus before checking for local memory and a bus timeout does not occur.

Figure 11 shows the translation from a Q22-bus address to a local memory address. The sequence is

1. Bits 21:09 of the Q22-bus address are extracted and used to select the map register.
2. The map register is selected and its V bit 31 is checked. If the V bit is not set, the operation terminates.
3. Map register bits 19:00 are used for bits 28:09 of the local memory physical address and bits 08:00 of the Q22-bus address are used for bits 08:00 of the local memory physical address.

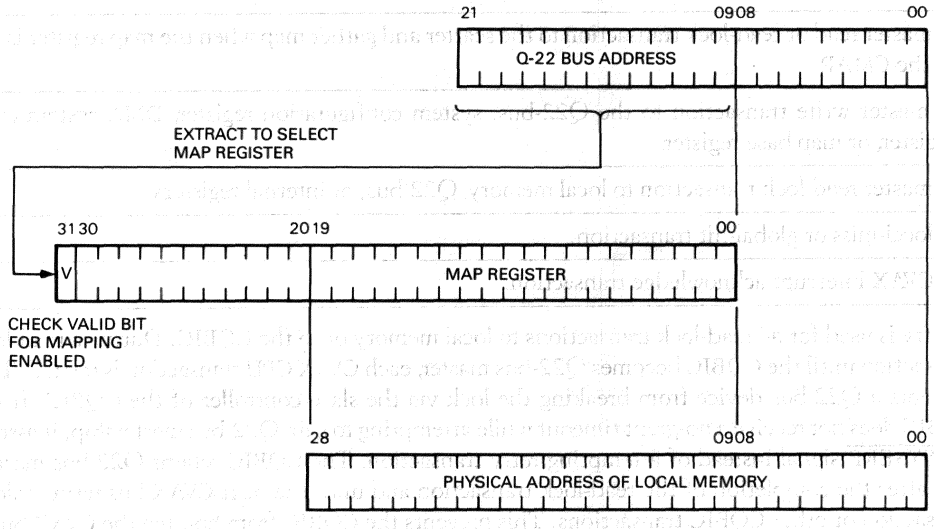


Figure 11 • CVAX 78711 Q22-Bus to Local Memory Address Mapping

Operation Modes

The CQBIC operates in arbiter or auxiliary mode as selected by the $\overline{AUX}$ input. When $\overline{AUX}$ is negated, arbiter mode is selected. When it is asserted, auxiliary mode is selected.

Arbiter mode—During this mode, the CQBIC is the Q22-bus arbiter and controls the Q22-bus DMA arbitration, powerup and powerdown and reset protocols, and powerfail and restart detection. It also supports the no-grant timeouts and no-sack bus grant aborts.

Auxiliary mode—During this mode, the CQBIC is a Q22-bus auxiliary device and controls the powerup and powerdown and reset protocols, and the powerfail and restart detection. It also supports Q22-bus request and mastership protocols, the DMA and IAK daisychain functions, and the nonexistent Q22-bus memory timeouts.

Retry Handling

The CQBIC requests a retry that causes the CVAX to release ownership of the CVAX bus and to retry the current transaction. This allows the CQBIC to obtain bus ownership and to perform a transaction such as fetching a map register from local memory or handling a bus deadlock.

The CQBIC detects a retry request when the $\overline{ERR}$ or $\overline{RDY}$ signal is asserted. The $\overline{SYSERR}$ and $\overline{SYSRDY}$ signals are synchronized by the CVAX clock and become the $\overline{RDY}$ and $\overline{ERR}$ inputs to the CVAX CPU. The CVAX CPU uses two sampling windows to detect an error or a retry request. When $\overline{ERR}$ is asserted in the first sampling window, the CVAX CPU waits for the second sampling window. If $\overline{RDY}$ is asserted in the second window, the CVAX CPU retries the transaction and if $\overline{RDY}$ is not asserted in the second sampling window, the CVAX CPU detects an error. The CQBIC requests a retry by asserting $\overline{SYSERR}$, waiting 50 nanoseconds, and then asserting $\overline{SYSRDY}$. The 50 nanosecond delay ensures that the CVAX clock chip synchronizer does not skew the $\overline{SYSRDY}$ signal so that the retry request is not detected by the CVAX CPU.

Any CQBIC master transaction that is deadlocked because the CQBIC slave controller needs CVAX bus ownership results in the CQBIC issuing a retry request to the CVAX CPU. These transactions are

- A master read transaction to the Q22-bus.
- A master read or read-lock transaction to the scatter and gather map when the map register is not in the CMAP.
- A master write transaction to the Q22-bus, system configuration register, DMA system error register, or map base register.
- A master read-lock transaction to local memory, Q22-bus, or internal registers.
- A local-miss or global-hit transaction.
- A CVAX interrupt acknowledge transaction.

A retry is used for all read-lock transactions to local memory or to the CQBIC. During a read-lock transaction until the CQBIC becomes Q22-bus master, each CVAX CPU transaction is retried. This prevents a Q22-bus device from breaking the lock via the slave controller of the CQBIC. If the CQBIC does not receive a no-grant timeout while attempting to gain Q22-bus mastership, it asserts the $\overline{SYSERR}$ signal instead of attempting retry transaction. The CQBIC retains Q22-bus mastership after the completion of the read-lock transaction and until the next CVAX bus write-unlock transaction or other CQBIC transactions. This prevents the CQBIC from holding the CVAX bus if the CVAX CPU does not recognize the read-lock transaction and does not complete the write-unlock.

Interrupt Handling

The function of the CQBIC during an interrupt request depends on its mode. In arbiter mode, the CQBIC provides the interface between the CVAX interrupt system and the Q22-bus. In auxiliary mode the CQBIC responds as an auxiliary device on the Q22-bus and as a device on the CVAX bus. It blocks interrupt acknowledge cycles from the Q22-bus.

The CQBIC is placed at the end of the daisychain configuration for all four interrupt priority levels used by the CVAX CPU. This is necessary so that when no other device on the CVAX bus has requested an interrupt and when the CQBIC has no outstanding interrupts, it transfers the assertion of $\overline{IAKI}$ signal to the Q22-bus or blocks it and asserts the $\overline{ERR}$ signal.

Arbiter mode—During arbiter mode, the CQBIC transfers Q22-bus interrupt requests directly to the local processor by mapping the Q22-bus interrupt request on the $\overline{BIRQ<7:4>}$ lines to the $\overline{IRQ<3:0>}$ inputs of the local processor. The CVAX responds to an interrupt request on these lines with an interrupt acknowledge cycle. The CQBIC can also request an interrupt by asserting the $\overline{MEMERR}$ signal and/or the $\overline{PWRFL}$ signal. The CVAX does not respond to $\overline{MEMERR}$ or $\overline{PWRFL}$ interrupt requests with an interrupt acknowledge cycle.

The CQBIC responds to a CVAX CPU interrupt acknowledge cycle when the $\overline{IAKI}$ input is asserted. The response is determined by the IPL acknowledged and if the CQBIC has any outstanding interrupts as follows:

- For an interrupt acknowledge at IPL 17, IPL 16, or IPL 15, the CQBIC initiates a Q22-bus IAK transaction. When the CQBIC receives the vector from the interrupting device on the Q22-bus, it appends bits 9 and 0 to the vector and passes the vector to the local processor. Both bits 9 and 0 of the vector are set to force the vector address into unallocated device vector space (>200 hexadecimal) and force the processors interrupt priority level to IPL 17.

- For an interrupt acknowledge cycle at IPL 14 when the CQBIC has a doorbell interrupt request pending, the CQBIC responds by returning vector 204 (hexadecimal) to the CVAX CPU and by asserting the $\overline{RDY}$ output. When no doorbell interrupt request is pending, the CQBIC initiates a Q22-bus interrupt acknowledge transaction. This transaction is processed the same as an interrupt acknowledge cycle at IPL 17, IPL 16, or IPL 15.

Auxiliary mode—During auxiliary mode, the CQBIC blocks interrupt acknowledge cycles in response to IPL 17, IPL 16, and IPL 15 from being transferred to the Q22-bus. The CQBIC processes interrupt requests from its doorbell register and can also request an interrupt by asserting the $\overline{MEMERR}$ and/or $\overline{PWRFL}$ signal. The CVAX does not respond to a $\overline{MEMERR}$ or $\overline{PWRFL}$ interrupt request with an interrupt acknowledge cycle.

The response of the CQBIC to a CVAX CPU interrupt acknowledge cycle when its $\overline{IAKI}$ input is asserted is determined by the IPL acknowledged and if outstanding interrupts are pending as follows:

- For an interrupt acknowledge cycle at IPL 17, IPL 16, or IPL 14, the CQBIC blocks the cycle from the Q22-bus and asserts the $\overline{SYSERR}$ output to end the cycle.

- For an interrupt acknowledge cycle at IPL 14 when the CQBIC has a doorbell interrupt request pending, the CQBIC responds by returning vector 204 (hexadecimal) to the CVAX CPU and by asserting the $\overline{RDY}$ output. When no doorbell interrupt request is pending, the CQBIC blocks the cycle from the Q22-bus and asserts the $\overline{SYSERR}$ signal to end the cycle. When $\overline{SYSERR}$ terminates the cycle, the CVAX CPU ignores the interrupt request and does not take an exception to the termination.

Error Handling

The classes of errors detected and reported by the CQBIC are nonexistent Q22-bus memory and I/O references, nonexistent local memory references, no-grant timeout, no-sack abort, slave memory error reporting, master parity error detection, and local-miss and global-hit nonexistent memory and memory errors. These are grouped into errors processed by the master section of the CQBIC, the slave section of the CQBIC, local-miss and global-hit errors, and CQBIC arbiter errors.

The CQBIC reports errors and error status to the CVAX CPU using the following signals and registers: $\overline{\text{SYSERR}}$, $\overline{\text{MEMERR}}$, $\overline{\text{PWRFL}}$, the DMA System Error Register (DSER), Master Error Address Register (MEAR), and Slave Error Address Register (SEAR). The assertion of the $\overline{\text{SYSERR}}$ signal causes the CPU to terminate the current transaction and to take a machine check for errors that occur on demand read and write transactions. When reporting an error to the CVAX CPU by asserting $\overline{\text{SYSERR}}$, the CQBIC sets CDAL<31:00> to valid logic levels. The assertion of the $\overline{\text{MEMERR}}$ and $\overline{\text{PWRFL}}$ signals are recognized as interrupt requests by the CVAX CPU.

All parity and memory error flags and error addresses are latched and held until cleared by the CVAX CPU. Additional parity or memory errors that occur will set the Lost Error bit 03 in the DSER.

Master section errors—The CQBIC processes nonexistent memory errors as follows:

- During demand read transactions, the CQBIC asserts the $\overline{\text{SYSERR}}$ signal to terminate the transaction, sets the NXM flag bit 07 in the DSER, and latches the address in the MEAR.
- During a request read or interrupt acknowledge transaction, the CQBIC asserts the $\overline{\text{SYSERR}}$ signal to terminate the transaction and no error information is logged.
- During a write transaction, the CQBIC sets the NXM flag bit 07 in the DSER and asserts the $\overline{\text{MEMERR}}$ signal to post a write timeout interrupt request. $\overline{\text{MEMERR}}$ is asserted with the next assertion of the $\overline{\text{AS}}$ signal and deasserted with the next assertion of the $\overline{\text{DS}}$ signal.

Multiple longword transfer to Q22-bus—If the CVAX CPU attempts to perform a multiple longword transfer to the Q22-bus, the CQBIC asserts $\overline{\text{SYSERR}}$ to terminate the transaction. Because the Q22-bus address space is located in the I/O space of the CPU, only longword transfers with byte masks to this space are legal.

No-grant timeout—If the CVAX CPU attempts to obtain Q22-bus mastership and does not succeed within 10 milliseconds, the CQBIC terminates the transaction by asserting $\overline{\text{SYSERR}}$. If the transaction is a demand read, the No Grant Timeout bit 02 is set in the DSER.

Master parity error—The CQBIC processes master parity errors as follows:

- During a demand read transaction from the Q22-bus, the CQBIC asserts $\overline{\text{SYSERR}}$ to terminate the transaction, sets the Master Parity Error bit 05 in the DSER, and latches the address in the MEAR.
- During a request read transaction, the CQBIC asserts $\overline{\text{SYSERR}}$ to terminate the transaction and no error information is logged.

Slave section errors—A slave read or write transaction that results in a nonexistent memory error causes the CQBIC to set the SLAVE DMA NXM flag bit 00 in the DSER, latch the error address in the SEAR, and assert $\overline{\text{MEMERR}}$ to post an interrupt to the CVAX CPU.

A slave read or write transaction that results in an error with the the $\overline{\text{SYSERR}}$ signal asserted is as follows:

- A slave read transaction that results in a parity error causes the CQBIC to set the DMA QME bit 15 in the doorbell register, set Slave Memory Error bit 04 in the DSER, and latch the translated error address into the SEAR. The CQBIC then reports the error to the Q22-bus by asserting $\overline{BDAL<17:16>}$ during the data transfer of the transaction.

- A slave write transaction that results in an error causes the CQBIC to set the DMA QME bit 15 in the doorbell register and the Slave Memory Error bit 04 in the DSER, latch the translated error address in the SEAR and assert $\overline{MEMERR}$ to post an interrupt to the CVAX CPU. The CQBIC does not inform the Q22-bus of the error.

- A slave read or write transaction to the scatter and gather map that results in an error causes the CQBIC to set the Slave Memory Error bit 04 in the DSER, latch the translated error address into the SEAR, and assert $\overline{MEMERR}$ to post an interrupt to the CVAX CPU.

Local-miss and global-hit errors—During local-miss and global-hit read transaction, the CQBIC issues a retry request to the CPU by asserting $\overline{SYSERR}$ and $\overline{SYSRDY}$ and latches the mapped address. The CQBIC performs a read transaction from local memory and stores the data. When the CPU tries again, the CQBIC returns the data. If an error was detected during the read transfer from local memory, the CQBIC asserts $\overline{SYSERR}$ to notify the CPU of the error, latches the address in the SEAR, and if the transaction is a demand read, it sets the Slave Memory Error bit 04 or Slave DMA NXM bit 00 in the DSER.

During local-miss and global-hit write transactions, the CQBIC latches the address and write data and asserts $\overline{SYSRDY}$. The CQBIC performs a write transaction to write to local memory. If an error occurs during the transfer to local memory, the CQBIC latches the address into the SEAR, sets the slave memory error bit 04, or Slave DMA NXM bit 00 in the DSER, and asserts the $\overline{MEMERR}$ signal to post an interrupt to the CPU.

Arbiter errors—When the CQBIC arbiter grants the Q22-bus by asserting the $\overline{BDMGO}$ signal and does not receive the assertion of $\overline{BSACK}$ within 10 microseconds, it removes the grant and no errors are reported. The arbiter waits 500 nanoseconds for the $\overline{BDMGO}$ daisychain to clear before beginning arbitration again.

Initialization

When the $\overline{IORESET}$ input is asserted, the CQBIC asserts the $\overline{DMR}$ and $\overline{BDMR}$ outputs to gain ownership of both buses before the assertion of any reset signals. Once the CQBIC has been granted ownership of both buses ($\overline{DMG}$ and $\overline{BDMGI}$ asserted), it asserts the $\overline{RINIT}$ output for 10 microseconds. The assertion of $\overline{RINIT}$ can be used to clear local devices or registers, local interrupt enable bits, and pending local interrupts. The doorbell register and DMA system error register are reset. If the CQBIC is Q22-bus arbiter, it asserts $\overline{BINIT}$ for a minimum of 10 microseconds to clear the Q22-bus. If the CQBIC is in auxiliary mode, $\overline{BINIT}$ is not asserted. After 10 microseconds, $\overline{RINIT}$ and $\overline{BINIT}$ are deasserted and are followed by the deassertion of $\overline{DMR}$ and $\overline{BDMR}$. The system is now initialized and normal system operation can follow.

The response of the CQBIC to the assertion of $\overline{BINIT}$ on the Q22-bus is determined by its mode of operation. In arbiter mode, the assertion of $\overline{BINIT}$ is ignored and in auxiliary mode the assertion of $\overline{BINIT}$ causes the CQBIC to assert $\overline{RINIT}$ and $\overline{SYSRESET}$ to reset the local processor. The $\overline{RINIT}$, $\overline{DMR}$, and $\overline{BDMR}$ signals are asserted when $\overline{IORESET}$ is asserted.

During powerup, the CQBIC sets the CVAX bus and Q22-bus lines to a high impedance.

Interfacing Requirements

Figure 12 shows a CVAX CPU system using the CQBIC as an interface to the Q22-bus.

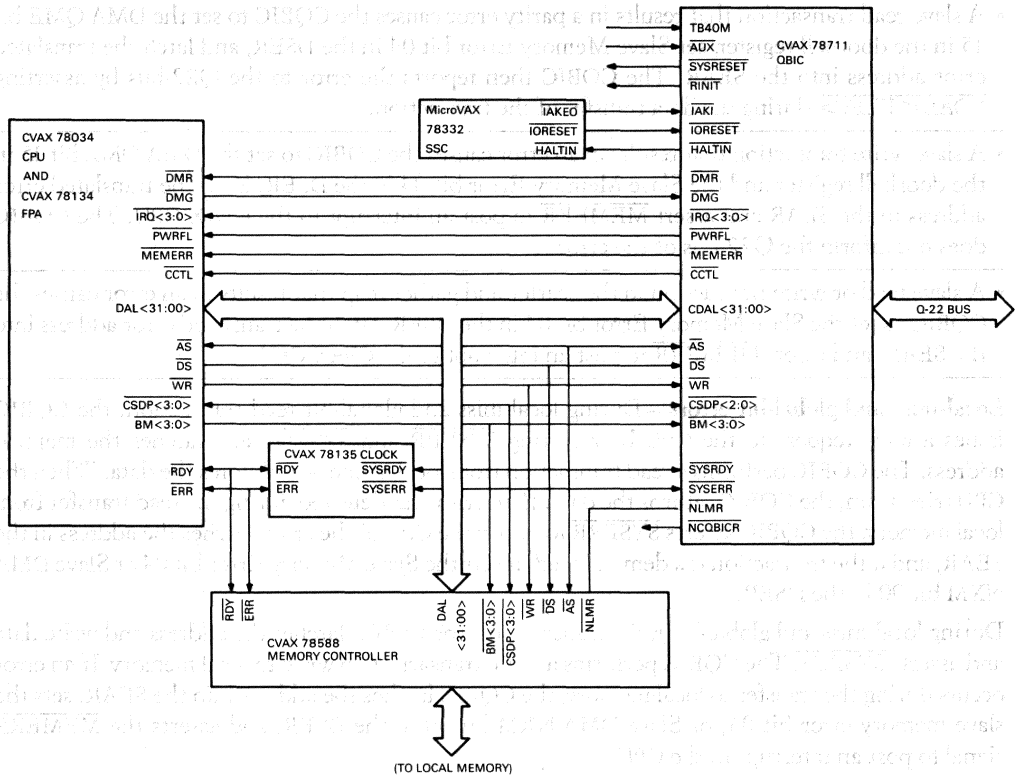


Figure 12 • CVAX 78711 System Interconnect Diagram

• Specifications

The mechanical, electrical, and environmental characteristics and specifications for the CQBIC are described in the following paragraphs. The test conditions for the electrical values are as follows unless specified otherwise.

- Ambient temperature (T_A): 0°C to 70°C
- Power supply voltage (V_{DD}): 4.75 V to 5.25 V

Mechanical Configuration

The physical dimensions of the CVAX 78711 132-pin cerquad package are contained in the Appendix.

Absolute Maximum Ratings

Stresses greater than the absolute maximum ratings may cause permanent damage to the device. Exposure to the absolute maximum ratings for extended periods may adversely affect the reliability of the device.

- Storage temperature (T_S): −55°C to 125°C

- Active temperature (T_A): 0°C to 70°C
- Power supply voltage (V_{DD}): −0.5 V to 7.0 V
- Input or output voltage applied: −0.5 V to V_{DD} + 0.5 V

dc Electrical Characteristics

The dc electrical characteristics for the CQBIC are grouped into Q22-bus signals and CVAX bus and CQBIC specific signals. Table 18 lists the electrical specifications for the Q22-bus signals and Table 19 lists the electrical specifications for the CVAX bus and CQBIC specific signals. The specifications for the dc tests are

- Power dissipation: < 1.5 watts
- Minimum airflow: 100 linear ft/min
- Temperature (T_A): 70°C
- Power supply voltage (V_{DD}): 4.75 V (except where noted)
- Ground (V_{SS}): 0 V

Table 18 • CVAX 78711 Q22-bus dc Parameters

Symbol	Parameter	Requirements Min.	Max.	Units	Test Conditions
V_{IH}	High-level input voltage	1.9	—	V	V_{in} = 5.25 V
		1.72	—	V	V_{in} = 4.75 V
V_{IL}	Low-level input voltage	—	1.66	V	V_{in} = 5.25 V
		—	1.5	V	V_{in} = 4.75 V
V_{OLD}	Low-level output voltage (open drain)	0.9	—	V	I_{out} = 100 mA
I_{IL}	Input leakage current	−10	10	μA	0 < V_{in} < 5.25 V
I_{OL}	Output leakage current	−50	50	μA	0 < V_{in} < 5.25 V
C_{in}	Input capacitance	—	10	pF	
C_{out}	Output capacitance	—	10	pF	

Table 19 • CVAX 78711 CQBIC and CVAX Bus dc Parameters

Symbol	Parameter	Requirements Min.	Max.	Units	Test Conditions
V_{IH}	High-level input voltage	2.0	—	V	
V_{IL}	Low-level input voltage	—	0.8	V	

Symbol	Parameter	Requirements Min.	Max.	Units	Test Conditions
V_{OH}	High-level output voltage	2.4	—	V	$I_{OH} = -400\ \mu A$
V_{OL}	Low-level output voltage	—	0.4	V	$I_{OL} = 2.0$ mA
V_{OLD}	Low-level output voltage (open drain)	—	0.2	V	$I_{OL} = 20$ mA
V_{OHM}	High-level output voltage (MOS signal)	3.0	—	V	$I_{OH} = -100\ \mu A$
V_{OLM}	Low-level output voltage (MOS signal)	—	0.2	V	$I_{OL} = 1.0$ mA
I_{IL}	Input leakage current	-10	10	μA	$0 < V_{in} < 5.25$ V
I_{ILS}	Input leakage current (sustainer)	0.2	1.5	mA	$V_{in} = 0.4$ V
I_{OL}	Output leakage current	-10	10	μA	$0 < V_{in} < 5.25$ V
I_{DD}	Active supply current	—	220	mA	$I_{out} = 0$, $T_A = 0°C$
C_{in}	Input capacitance	—	7.0	pF	
C_{out}	Output capacitance	—	10	pF	
C_{io}	Bidirectional capacitance	—	20	pF	

ac Electrical Characteristics

The Q22-bus ac charateristics are measured under the following test conditions except where noted.

- Ambient temperature (T_A): 70°C
- Power supply voltage (V_{DD}): 4.75 V
- Capacitive load (C_L): 15 pF/330 pF
- Pullup resistor (R1): 91Ω
- Pulldown resistor (R2): 200Ω
- V_{out}: 1.60 V
- V_{IL}: 1.50 V
- V_{IH}: 1.72 V
- Input rise and fall time: 10 ns (1.2 to 2.2 V), (0.8 to 2.6 V)

The following notes apply to Figures 13 through 53 and their associated timing Tables 20 through 27.

- All times are in nanoseconds except where noted.
- The TB40M clock input is TTL-compatible and intended to operate at 40 MHz, ±0.01%. All Q22-bus timing parameters are derived from this clock.

- With TB40M = 40 MHz, the CVAX bus per phase timing can vary from 20 to 33 nanoseconds assuming that the CVAX 78588 Memory Controller retry delay is 4.0 microcycles (CMCTL 5/3 mode memory cycle bit set). For CVAX bus per phase timing greater than 33 nanoseconds, the TB40M clock is reduced by 0.13 × CVAX clock input. Example: CVAX clock in = 25 MHz, TB40M = 40 MHz – (0.13 × 25 MHz) = 36.7 MHz.
- P = CQBIC internal clock period. With TB40M = 40 MHz, this period is 50 nanoseconds.
- Nx = number of internal clock periods (P) required for Q22-bus mastership or to wait until the master and slave logic of the CQBIC becomes idle.
- Ny = number of internal clock periods (P) required to wait until the CQBIC does not require Q22-bus mastership.

Table 20 • CVAX 78711 Powerup/Powerdown and Initialization Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{DCHA}	BDCOK deassertion to $\overline{\text{HALTIN}}$ assertion	—	46.3
t_{DCHN}	BDCOK reassertion to $\overline{\text{HALTIN}}$ deassertion	—	62
t_{PDBI}	BDCOK deassertion to $\overline{\text{BINIT}}$ assertion	15	85
t_{PDOWN}	BDCOK deassertion to $\overline{\text{SYSRESET}}$, $\overline{\text{RINIT}}$, and $\overline{\text{DMR}}$ assertion	15	75
t_{PFLA}	BPOK deassertion to $\overline{\text{PWRFL}}$ assertion	1P – 10	1P + 105
t_{PFLN}	BPOK assertion to $\overline{\text{PWRFL}}$ deassertion	1P – 10	1P + 105
t_{PU1PF}*	VEXTCAP gets VSTMR to $\overline{\text{PWRFL}}$ negation	98304300	98304500
t_{PU1SR}*	VEXTCAP gets VSTMR to $\overline{\text{SYSRESET}}$ negation	98304300	98304500
t_{PU2PF}	BPOK assertion to $\overline{\text{PWRFL}}$ deassertion	2P – 10	2P + 105
t_{PU2SR}	BPOK assertion to $\overline{\text{SYSRESET}}$ deassertion	2P – 10	2P + 105

*VSTMR (VT+) is the EXTCAP positive-going threshold voltage that activates the internal timer: 2.5 V (min.), 3.0 V (typ.), and 3.7 V (max.)
Negative-going threshold voltage (VT-) of EXTCAP Schmitt trigger: 1.9 V (typ.), 1.5 V (min.), and 2.4 V (max.)

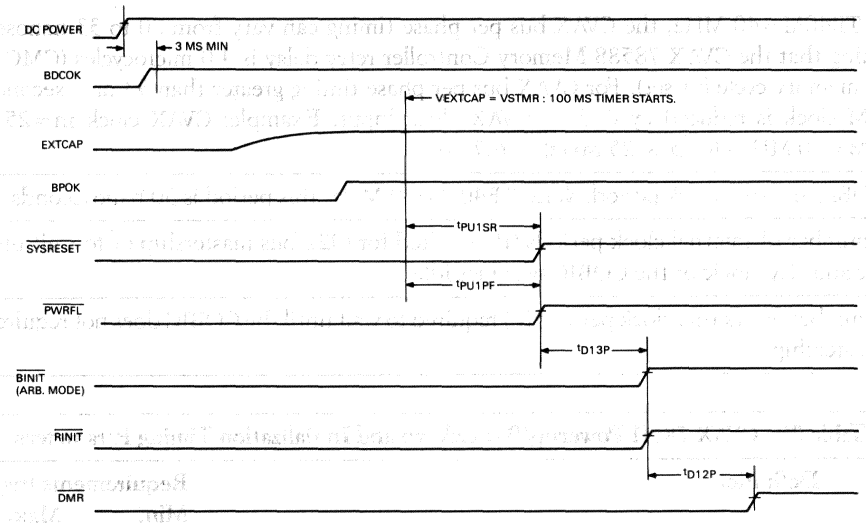


Figure 13 • CVAX 78711 DEC Standard Powerup (ac Power Normal) Timing

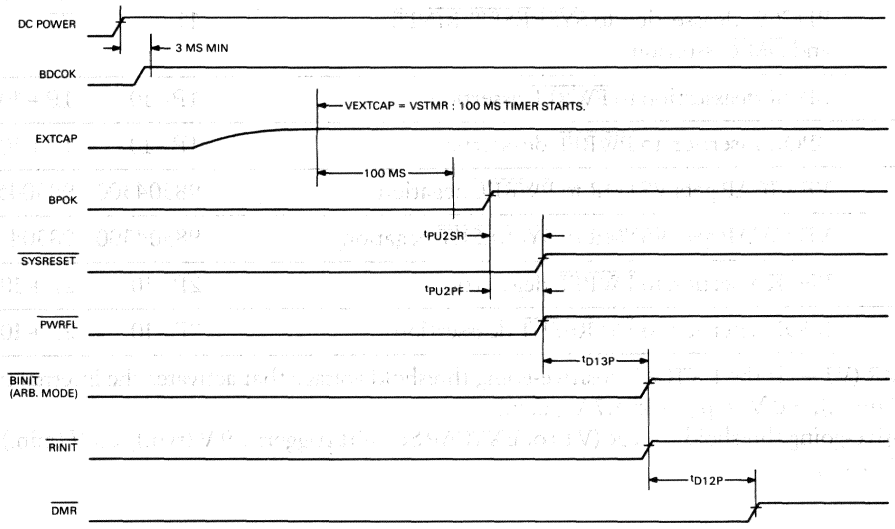


Figure 14 • CVAX 78711 DEC Standard Powerup (ac Power Failure) Timing

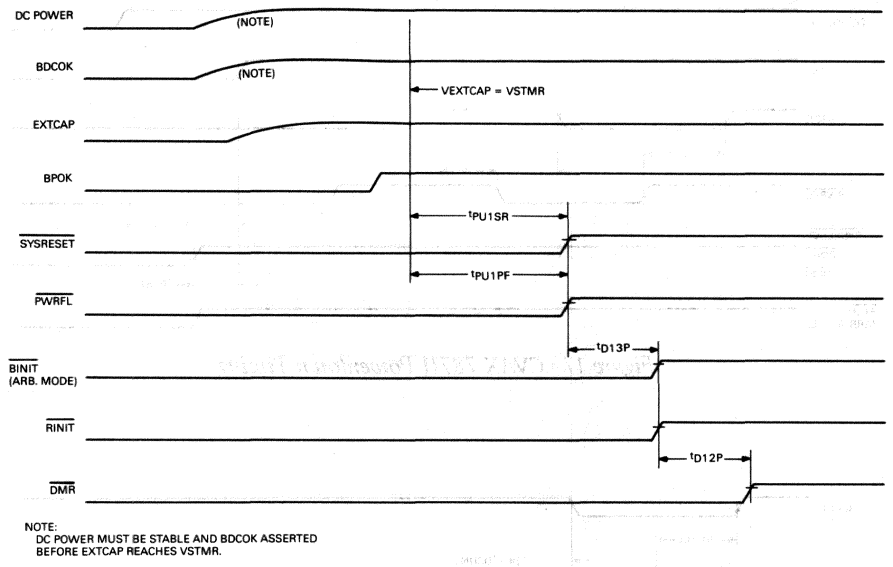


Figure 15 • CVAX 78711 Unsequenced Powerup (ac Power Normal) Timing

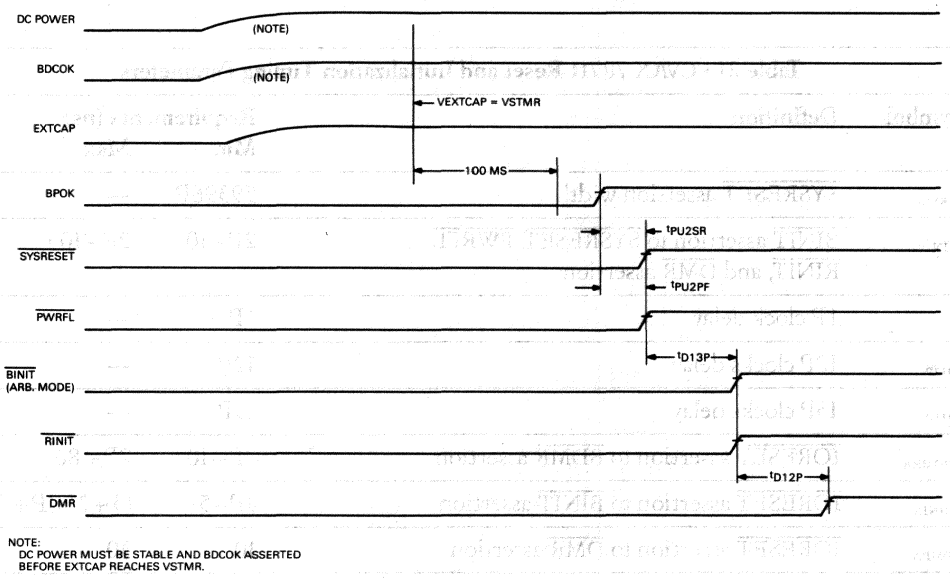


Figure 16 • CVAX 78711 Unsequenced Powerup (ac Power Failure) Timing

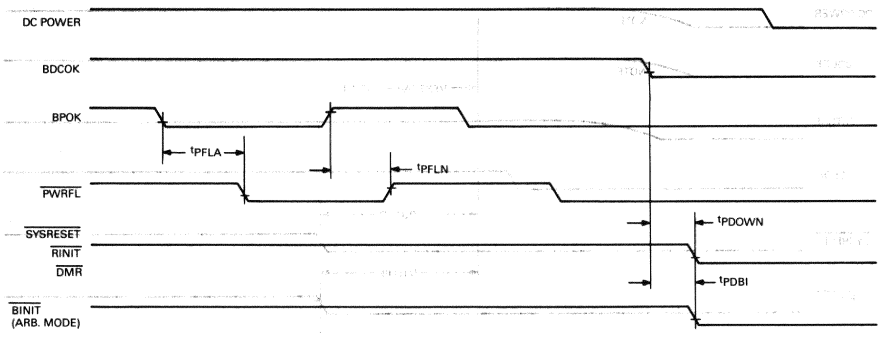


Figure 17 • CVAX 78711 Powerdown Timing

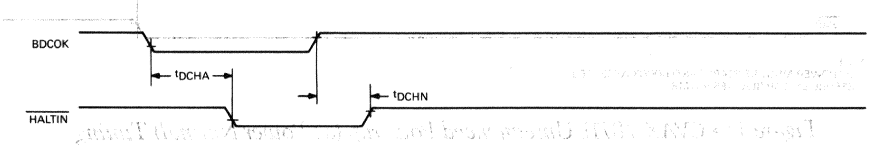


Figure 18 • CVAX 78711 Restart Timing

Table 21 • CVAX 78711 Reset and Initialization Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{ARESET}	SYSRESET assertion width	59396P	—
t_{BIN}	BINIT assertion to SYSRESET, PWRFL, RINIT, and DMR assertion	2P – 10	2P – 105
t_{D1P}	1P clock delay	1P	—
t_{D12P}	12P clocks delay	12P	—
t_{D13P}	13P clocks delay	13P	—
t_{IBDMRA}	IORESET assertion to BDMR assertion	2P – 10	2P + 80
t_{IBINIA}	IORESET assertion to BINIT assertion	3P – 5	(3 + Nx)P + 110*
t_{IDMRA}	IORESET assertion to DMR assertion	10	30
t_{INITW}	BINIT and RINIT assertion width	206P	—
t_{IORSTW}	IORESET assertion width	7P	—
t_{IRINIA}	IORESET assertion to RINIT assertion	3P – 10	(3 + Nx)P + 105*

*Nx is the number of CQBIC internal clock periods required to obtain bus mastership or to wait for the master or slave logic to become idle.

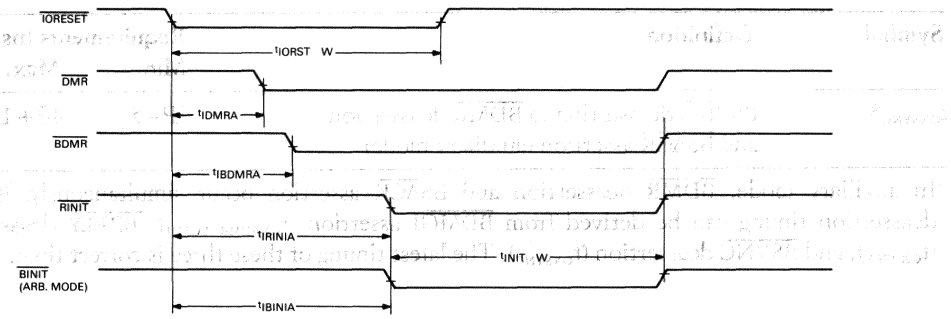


Figure 19 ▪ *CVAX 78711 I/O Reset Signal Timing*

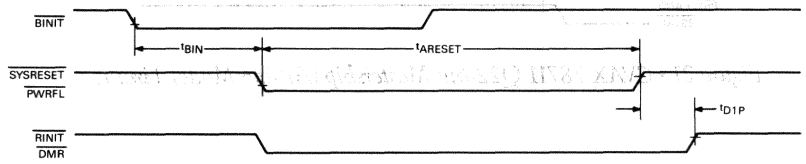


Figure 20 ▪ CVAX 78711 Q22-bus Initialize (Auxiliary Mode) Timing

Table 22 ▪ CVAX 78711 Arbiter Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{DMGOA1}	$\overline{BDMR}$ assertion to $\overline{BDMGO}$ assertion (arbiter mode)	2P-5	2P+110
t_{DMGON1}	$\overline{BSACK}$ assertion to $\overline{BDMGO}$ deassertion (arbiter mode)	2P-5	2P+110
t_{DMGOA2}	$\overline{BDMGI}$ assertion to $\overline{BDMGO}$ assertion (auxiliary mode)	2P-5	2P+110
t_{DMGON2}	$\overline{BDMGI}$ deassertion to $\overline{BDMGO}$ deassertion (auxiliary mode)	20	70
$t_{GABDMR1}$*	$\overline{BSACK}$ deassertion to $\overline{BDMR}$ deassertion (arbiter mode)	2P-5	2P+110
$t_{GXBDMR\ SACK1}$*	$\overline{BDMGI}$ assertion to $\overline{BDMR}$ deassertion and $\overline{BSACK}$ assertion (auxiliary mode)	3P-5	3P+110
t_{GXRPLY}*	$\overline{BRPLY}$ deassertion to $\overline{BDMR}$ deassertion and $\overline{BSACK}$ assertion (auxiliary mode)	2P-5	2P+110

Symbol	Definition	Requirements (ns) Min.	Max.
t_{GXSYNC}*	$\overline{BSYNC}$ deassertion to $\overline{BDMR}$ deassertion and $\overline{BSACK}$ assertion (auxiliary mode)	4P-5	4P+110

*In auxiliary mode, $\overline{BDMR}$ deassertion and $\overline{BSACK}$ assertion occur simultaneously. $\overline{BDMR}$ deassertion timing can be derived from $\overline{BDMGI}$ assertion ($t_{GXBDMR\ SCAK1}$), $\overline{BRPLY}$ deassertion (t_{GXRPLY}), and $\overline{BSYNC}$ deassertion (t_{GXSYNC}). The latest timing of these three is correct time.

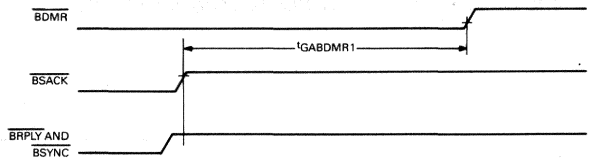


Figure 21 • CVAX 78711 Q22-bus Mastership (Arbiter Mode) Timing

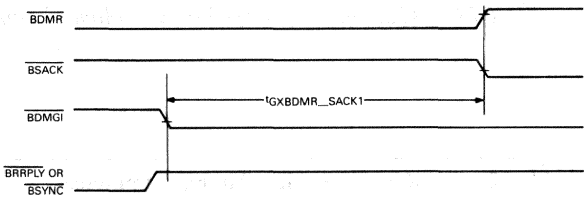


Figure 22 • CVAX 78711 Q22-bus Mastership (Auxiliary Mode with No Reply and No Bus Synch and a DMA Grant In) Timing

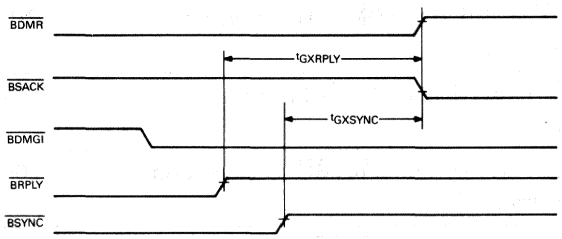


Figure 23 • CVAX 78711 Q22-bus Mastership (Auxiliary Mode with Bus Reply and Bus Synch and a DMA Grant In) Timing

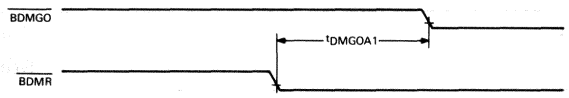


Figure 24 • CVAX 78711 DMA Grant Out Asserted (Arbiter Mode) Timing

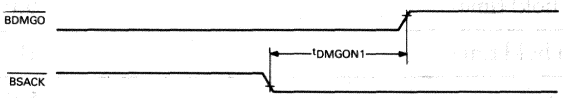


Figure 25 • CVAX 78711 DMA Grant Out Deasserted (Arbiter Mode) Timing

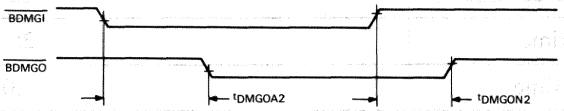


Figure 26 • CVAX 78711 DMA Grant In/Out Daisychain (Auxiliary Mode) Timing

Table 23 • CVAX 78711 CVAX Bus to CQBIC Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{DDALHZ}	DAL high-impedance delay	0	20
t_{DERRA}	$\overline{\text{SYSERR}}$ assertion delay	0	35
t_{DERRN}	$\overline{\text{SYSERR}}$ deassertion delay	0	10
t_{DHZ}	DAL high-impedance delay	0	40
t_{DMEA}	$\overline{\text{MEMERR}}$ assertion delay	0	45
t_{DMEN}	$\overline{\text{MEMERR}}$ deassertion delay	0	115
t_{DNQBA}	$\overline{\text{NCQBICR}}$ assertion delay	0	40
t_{DNQBN}	$\overline{\text{NCQBICR}}$ deassertion delay	0	60
t_{DRDAT}	Read data valid delay time	0	40
t_{DRDYA}	$\overline{\text{SYSRDY}}$ assertion delay	0	35
t_{DRDYN}	$\overline{\text{SYSRDY}}$ deassertion delay	0	10
t_{DWDAT}	Write data valid delay time	0	20
t_{HADR}	Address hold time	11	—

Symbol	Definition	Requirements (ns) Min.	Max.
t_{HBM}	$\overline{BM}$ hold time	5.0	—
t_{HCS}	$\overline{CS}$ hold time	11	—
t_{HIAKI}	$\overline{IAKI}$ hold time	0	—
t_{HRDAT}	Read data hold time	5.0	—
t_{HWDAT}	Write data hold time	11	—
t_{HWR}	$\overline{WR}$ hold time	5.0	—
t_{IAS}	$\overline{AS}$ deassertion to assertion	45	—
t_{IDS}	$\overline{DS}$ deassertion to assertion	100	—
t_{SADR}	Address setup time	22	—
t_{SAS}	$\overline{AS}$ setup time	26	—
t_{SBM}	$\overline{BM}$ setup time	2.0	—
t_{SCS}	$\overline{CS}$ setup time	22	—
t_{SDS}	$\overline{DS}$ setup time	20	—
t_{SIAKI}	$\overline{IAKI}$ setup time	17	—
t_{SWR}	$\overline{WR}$ setup time	22	—

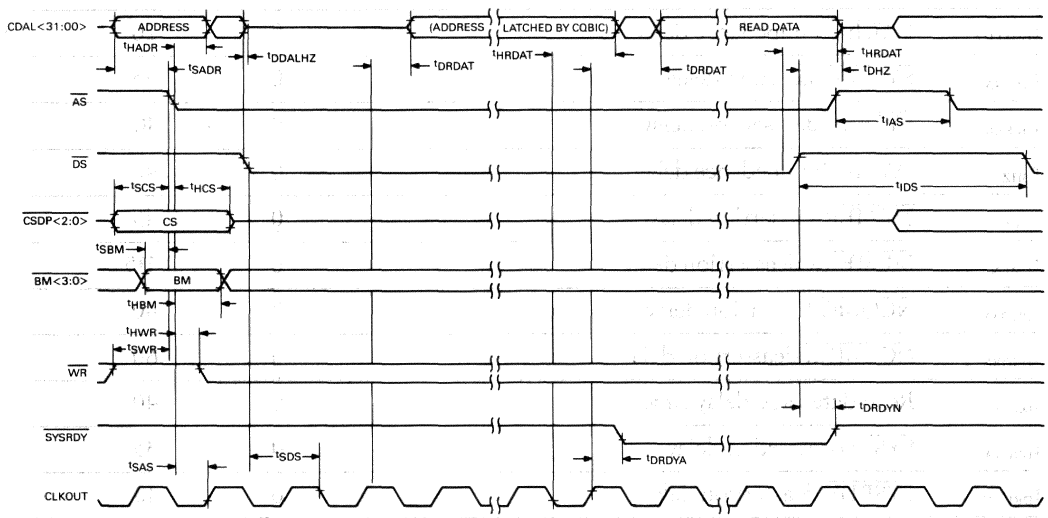


Figure 27 • CVAX 78711 CPU to CQBIC Read Cycle (System Ready) Timing

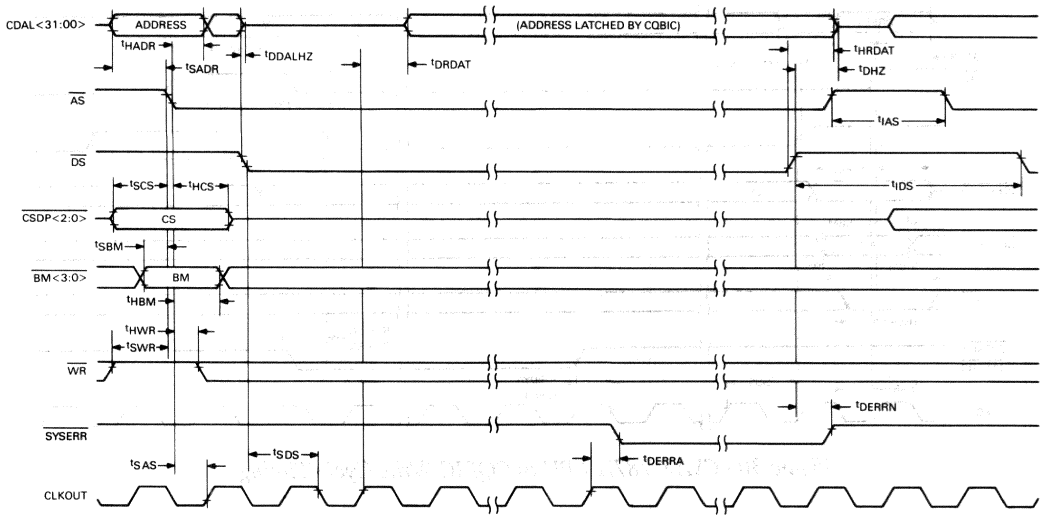


Figure 28 • CVAX 78711 CPU to CQBIC Read Cycle (System Error) Timing

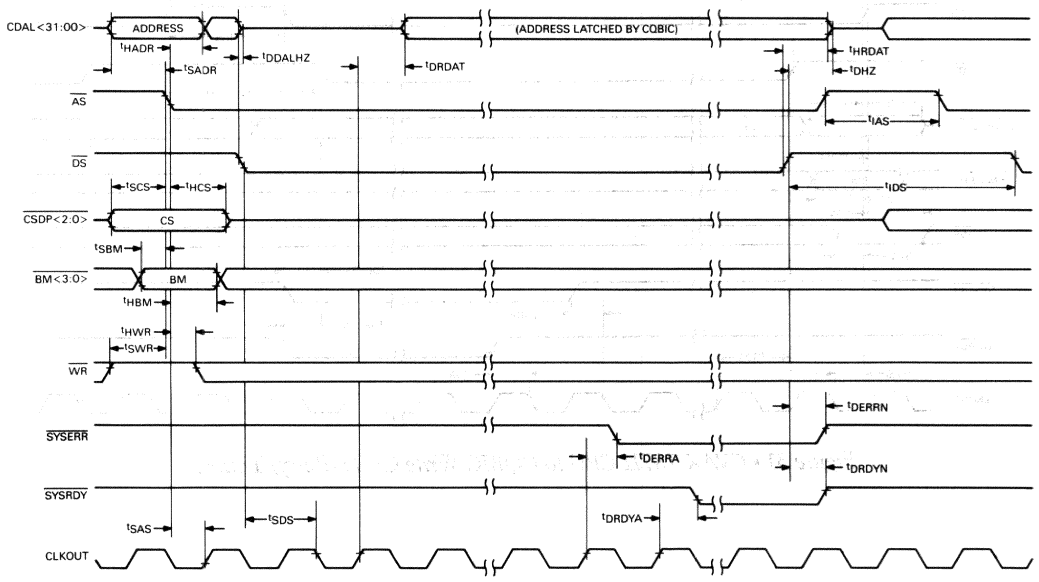


Figure 29 • CVAX 78711 CPU to CQBIC Read Cycle (Retry) Timing

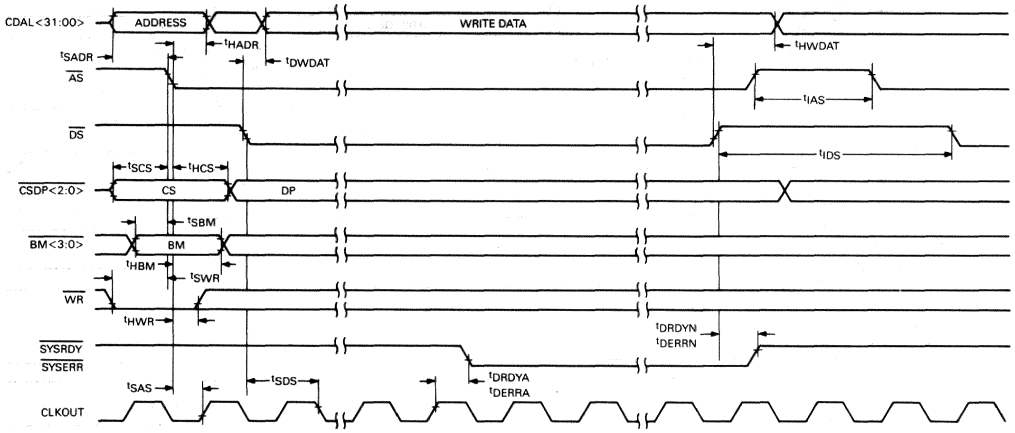


Figure 30 ▪ CVAX 78711 CPU to CQBIC Write Cycle Timing

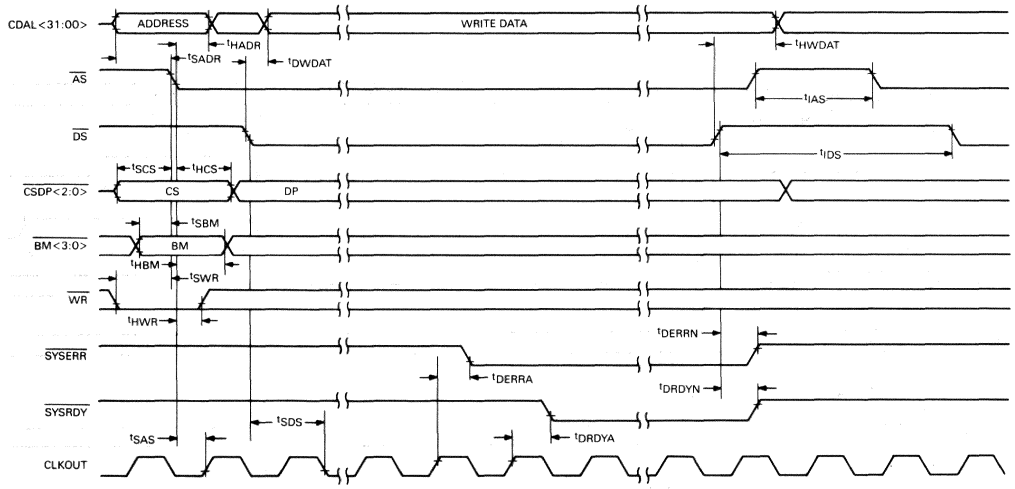


Figure 31 ▪ CVAX 78711 CPU to CQBIC Write Cycle (Retry) Timing

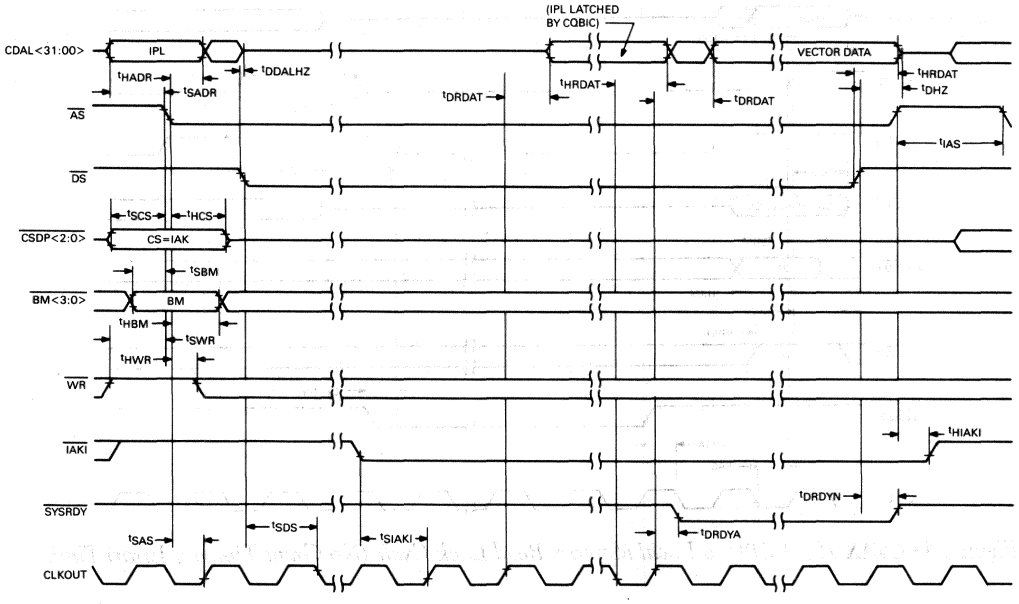


Figure 32 • CVAX 78711 CPU to CQBIC IAK Cycle Timing

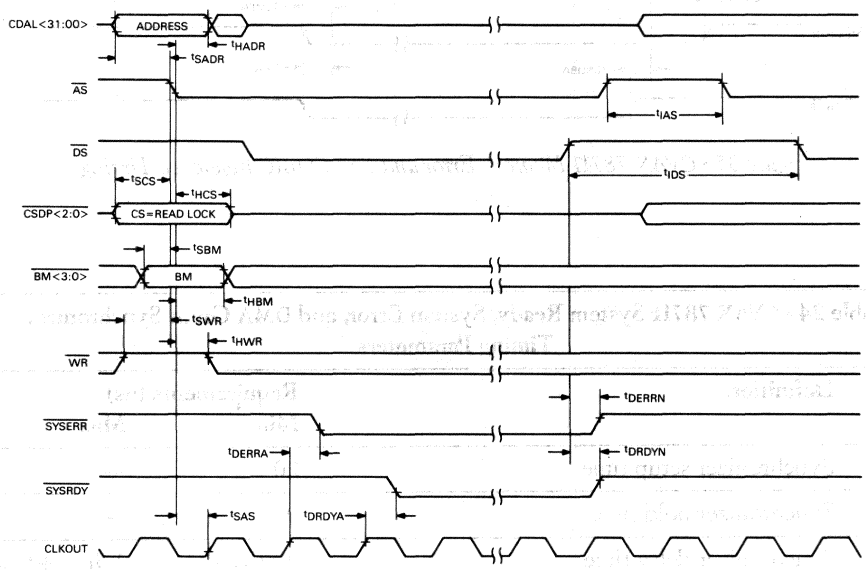


Figure 33 • CVAX 78711 CPU to Local Memory Read Lock Cycle (No Q22-Bus Mastership Retry) Timing

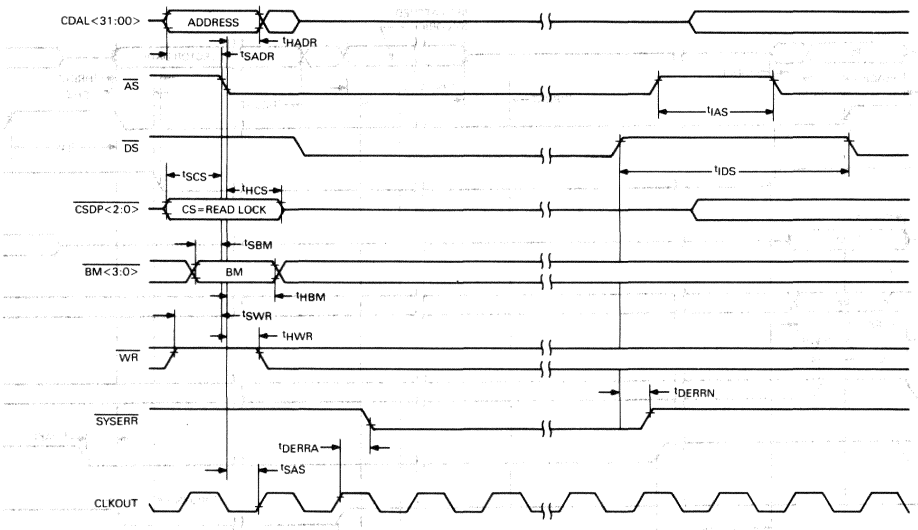


Figure 34 • CVAX 78711 CPU to Local Memory Read Lock Cycle (No Grant Timeout Error) Timing

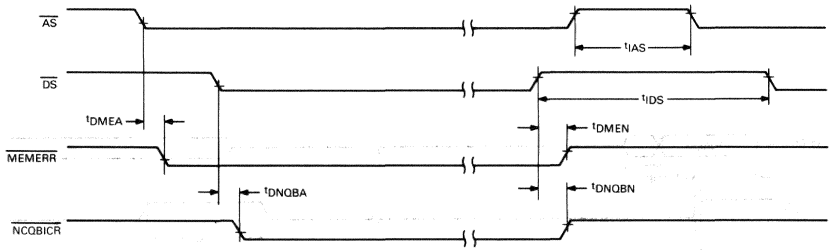


Figure 35 • CVAX 78711 Memory Error and Not CQBIC Reference Timing

Table 24 • CVAX 78711 System Ready, System Error, and DMA Grant Synchronizer Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{SYNS}	Synchronizer setup time	20	—
t_{SYNH}	Synchronizer hold time	0	—
t_{SYNC}	Synchronizer delay time	$0.5t_{CYC} + t_{SYNS}$	$1.5t_{CYC} + t_{SYNS}$

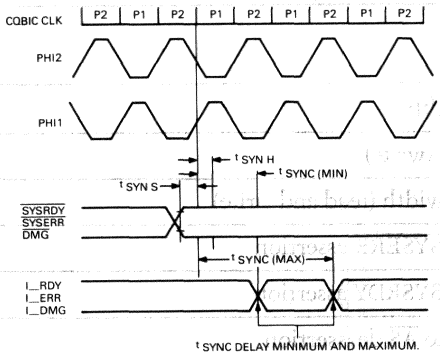


Figure 36 • CVAX 78711 CQBIC Clock to Internal Timing Synchronization

Table 21 • CVAX 78711 CVAX Bus Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{ASAW}	$\overline{AS}$ assertion width	200[1,2]	—
t_{ASDSR}	$\overline{AS}$ assertion to $\overline{DS}$ assertion (read)	25	70
t_{ASDSW}	$\overline{AS}$ assertion to $\overline{DS}$ assertion (write)	40	70
t_{ASNLMR}	$\overline{AS}$ assertion to $\overline{NLMR}$ assertion	100	—
t_{ASNW}	$\overline{AS}$ deassertion width	100	—
t_{ASDLY}	$\overline{AS}$ assertion to $\overline{CCTL}$ assertion	10	40
t_{ASNBWZ}	$\overline{BM<3:0>}$ high-impedance delay	—	30
$t_{ASNNLMRN}$	$\overline{AS}$ deassertion to $\overline{NLMR}$ deassertion	0	—
t_{ASOH}	Address hold time	20	70
t_{ASOS}	Address setup time	30	—
$t_{CCTLCYC}$	$\overline{CCTL}$ cycle time	445	—
t_{CCTLW}	$\overline{CCTL}$ assertion width	40	—
t_{DMGAS}	$\overline{DMG}$ assertion to $\overline{AS}$ assertion	60	250
t_{DMRG}	$\overline{DMR}$ assertion to $\overline{DMG}$ assertion	0	—
t_{DMRNW}	$\overline{DMR}$ deassertion width	130	—
t_{DMRNGN}	$\overline{DMR}$ deassertion to $\overline{DMG}$ deassertion	0	300
t_{DMRNZ}	$\overline{DMR}$ deassertion to output high-impedance	—	100[1]

Symbol	Definition	Requirements (ns) Min.	Max.
t_{DSAW}	$\overline{DS}$ assertion width	200	—
t_{DSDW}	Data setup time (write)	5.0	—
t_{DSNW}	$\overline{DS}$ deassertion width (read and write)	40	—
t_{DSSERR}	$\overline{DS}$ assertion to $\overline{SYSERR}$ assertion	200 (0)[3]	—
t_{DSSRDY}	$\overline{DS}$ assertion to SYSRDY assertion	200 (0)[3]	—
t_{DSNASN}	$\overline{DS}$ deassertion to $\overline{AS}$ deassertion	30	130
t_{DSNDH}	Data hold time (read)	0	—
$t_{DSNDMRN}$	$\overline{DS}$ deassertion to $\overline{DMR}$ deassertion	0	—
t_{DSNDN}	Data hold time (write)	0	—
$t_{DSNSERRN}$	$\overline{DS}$ deassertion to $\overline{SYSERR}$ deassertion	0[4]	—
$t_{DSNSRDYN}$	$\overline{DS}$ deassertion to $\overline{SYSRDY}$ deassertion	0[4]	—
$t_{NLMRDSN}$	$\overline{NLMR}$ assertion to $\overline{DS}$ deassertion	130	170
t_{SERRD}	$\overline{SYSERR}$ assertion to data delay	—	5.0
$t_{SERRDSN}$	$\overline{SYSERR}$ assertion to $\overline{DS}$ deassertion	100	—
$t_{SERRSRDY}$	$\overline{SYSERR}$ assertion to $\overline{SYSRDY}$ assertion	10	45
t_{SRDYD}	$\overline{SYSRDY}$ assertion to data delay	—	5.0
$t_{SRDYDSN}$	$\overline{SYSRDY}$ assertion to $\overline{DS}$ deassertion	0	—

[1]Valid for all CVAX bus signals driven by CQBIC during DMA.
[2]500 + 4 ($\overline{DS}$ to $\overline{SYSERR}$): octaword write to LM
[3]200 nanoseconds is required to satisfy the minimum $\overline{CCTL}$ cycle time ($t_{CCTLCYC}$). The CQBIC operates properly with 0 nanoseconds minimum cycle time except that the $\overline{CCTL}$ cycle time is not satisfied.
[4]$\overline{SYSERR}$ and $\overline{SYSERR}$ must be deasserted a minimum of 100 nanoseconds for synchronization.

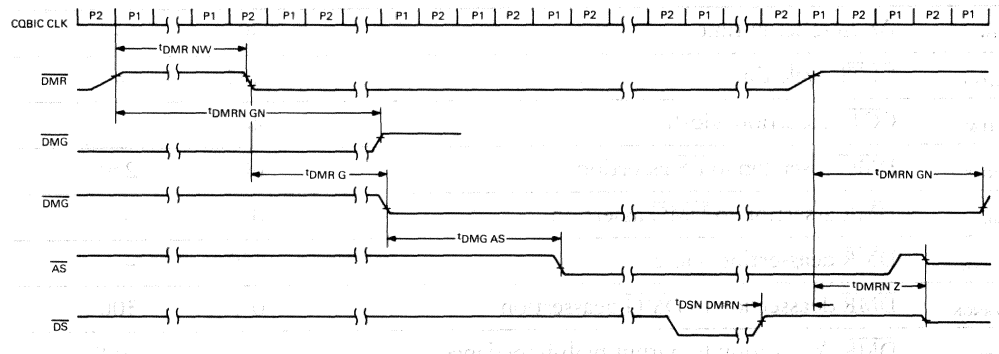
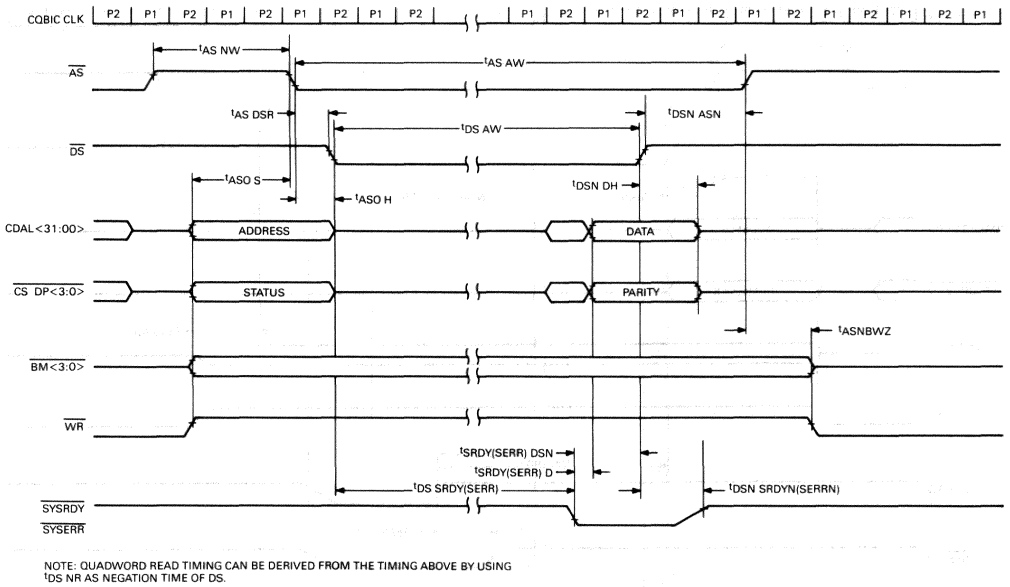


Figure 37 • CVAX 78711 Bus Arbitration Timing



NOTE: QUADWORD READ TIMING CAN BE DERIVED FROM THE TIMING ABOVE BY USING tDS NR AS NEGATION TIME OF DS.

Figure 38 • CVAX 78711 CQBIC to Local Memory Read Timing

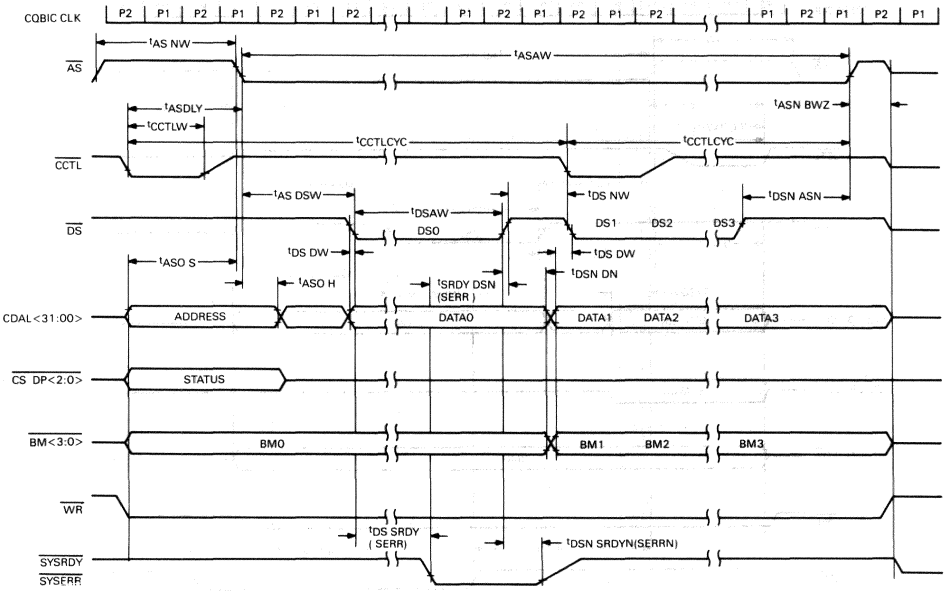


Figure 39 • CVAX 78711 CQBIC to Local Memory Write Timing

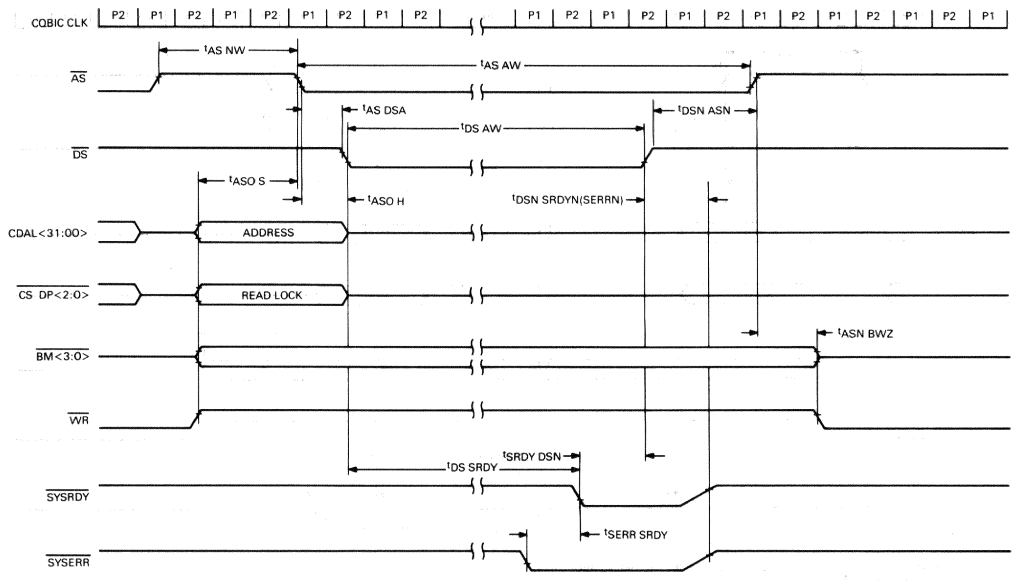


Figure 40 • CVAX 78711 CPU Retry Timing

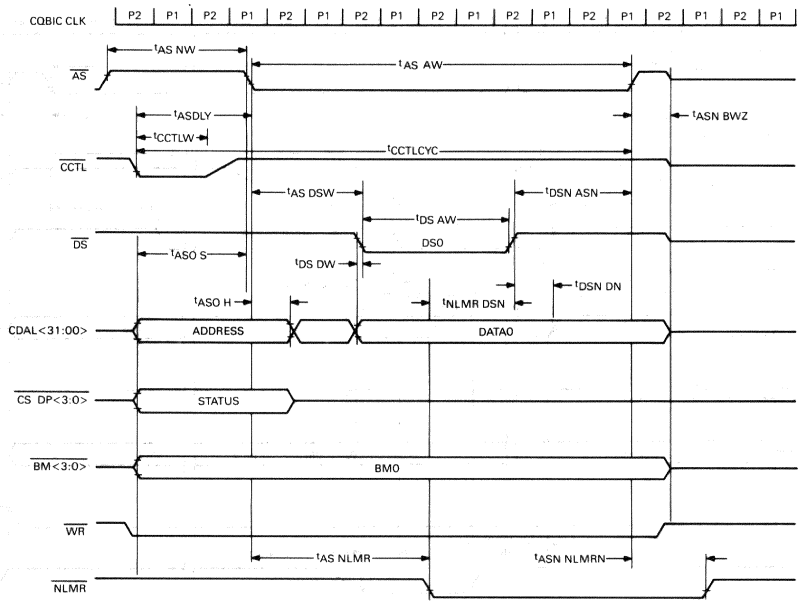


Figure 41 • CVAX 78711 Not Local Memory Reference Timing

Table 26 • CVAX 78711 CQBIC to Q22-bus Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{DBS7A}	$\overline{BBS7}$ assertion delay	0	37
t_{DBS7N}	$\overline{BBS7}$ deassertion delay	0	35
t_{DDINA}	$\overline{BDIN}$ assertion delay	0	42
t_{DDINN}	$\overline{BDIN}$ deassertion delay	0	45
t_{DDOUN}	$\overline{BDOUT}$ deassertion delay	0	40
t_{DDOUTA}	$\overline{BDOUT}$ assertion delay	0	42
t_{DIAKA}	$\overline{BIAKO}$ assertion delay	0	43
t_{DIAKN}	$\overline{BIAKO}$ deassertion delay	0	43
t_{DQDALA}	$\overline{BDAL}$ data assertion delay	0	60
t_{DQDLN}	$\overline{BDAL}$ data deassertion delay	0	55
t_{DSYNCA}	$\overline{BSYNC}$ assertion delay	0	37
t_{DSYNN}	$\overline{BSYNC}$ deassertion delay	0	35
t_{DWTBN}	$\overline{BWTBT}$ deassertion delay	0	55
t_{DWTBTA}	$\overline{BWTBT}$ assertion delay	0	52
t_{HQDAL}	$\overline{BDAL}$ data hold time	0	—
t_{HQRDAT}	$\overline{BDAL}$ read data hold time	10	—
t_{HREF}	$\overline{BREF}$ hold time	10	—
t_{HRPLY}	$\overline{BRPLY}$ hold time	10	—
t_{SREF}	$\overline{BREF}$ setup time	82	—
t_{SRPLY}	$\overline{BRPLY}$ setup time	57	—
t_{SQRDAT}	$\overline{BDAL}$ read data setup time	82	—

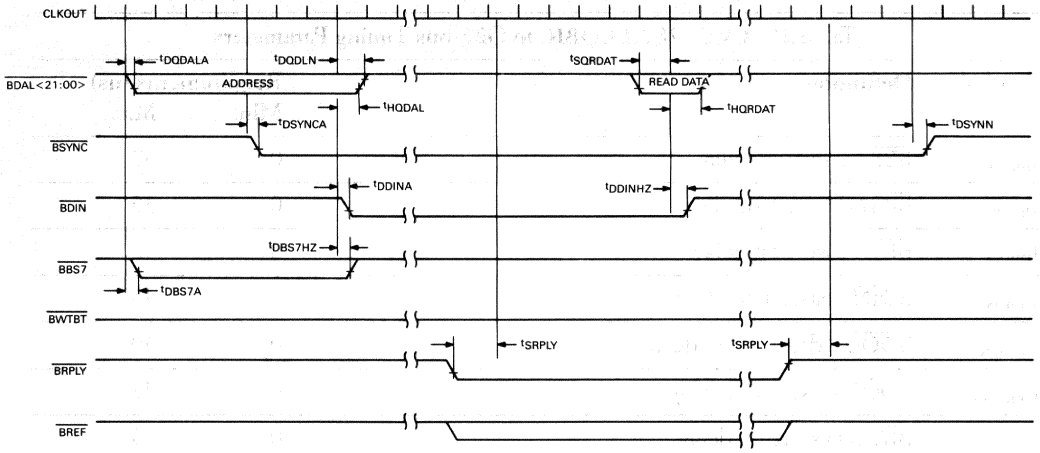


Figure 42 ▪ CVAX 78711 CQBIC to Q22-bus Single Transfer Read Timing

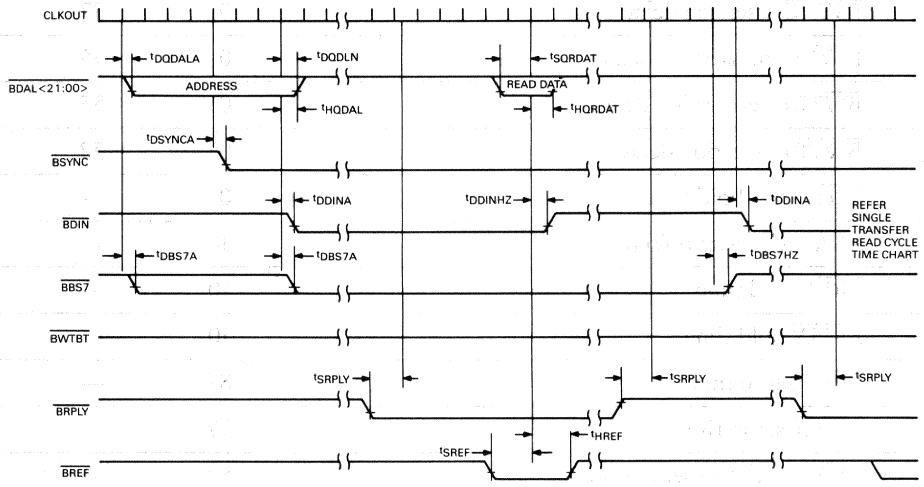


Figure 43 ▪ CVAX 78711 CQBIC to Q22-bus Block-mode Multiple Transfer Read Timing

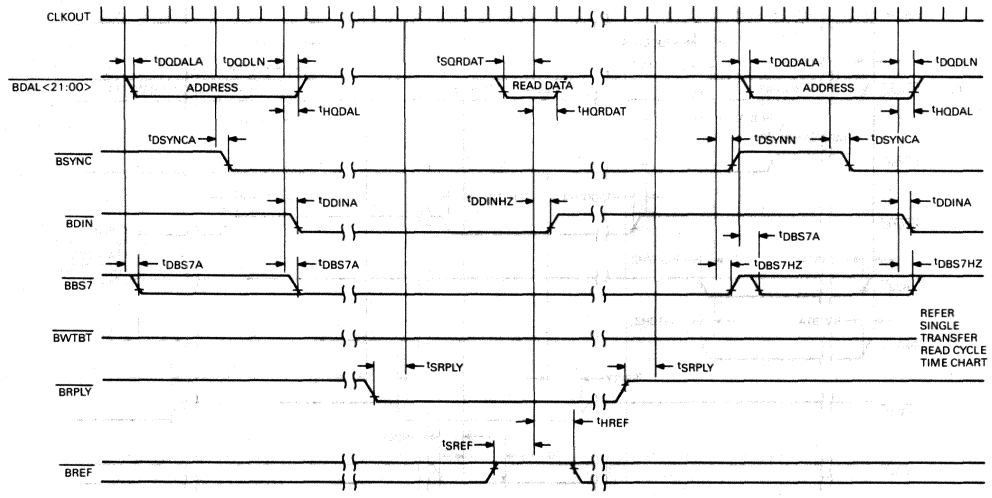


Figure 44 • CVAX 78711 CQBIC to Q22-bus Nonblock-mode Multiple Transfer Read Timing

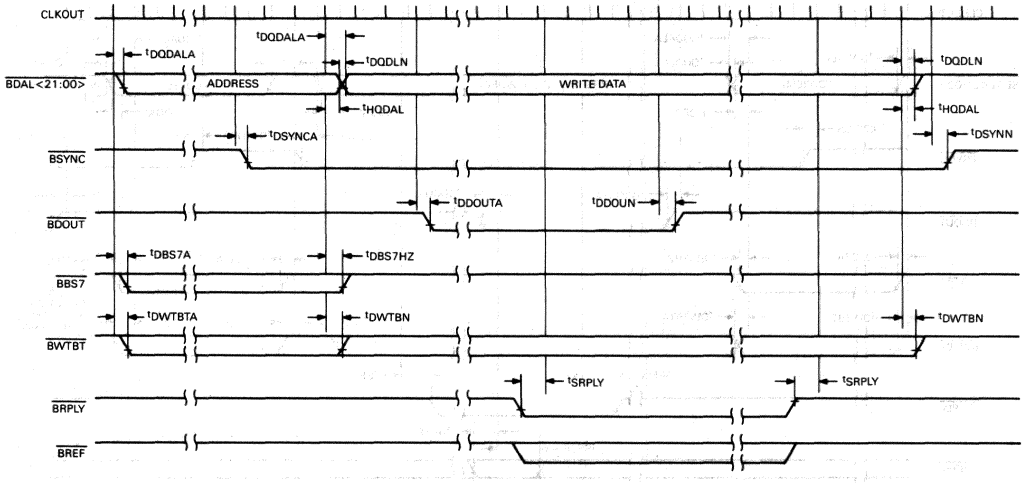


Figure 45 • CVAX 78711 CQBIC to Q22-bus Single Transfer Write Timing

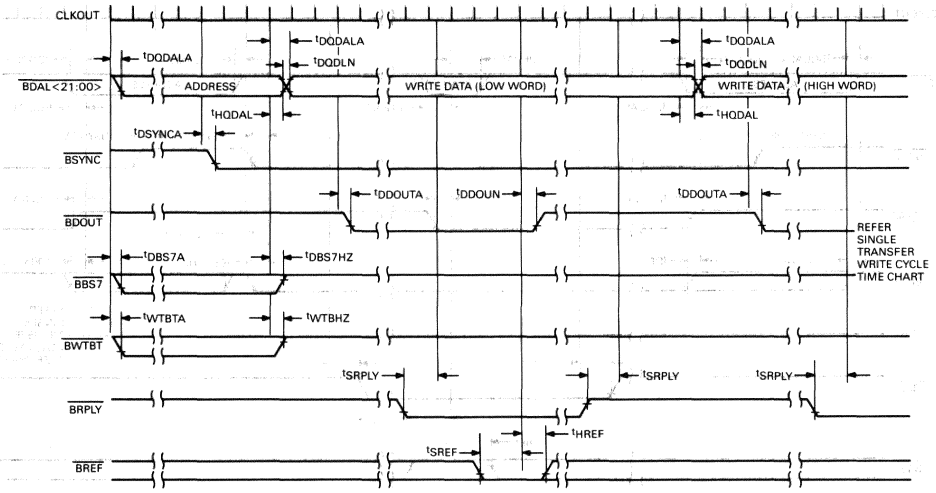


Figure 46 • CVAX 78711 CQBIC to Q22-bus Block-mode Write Timing

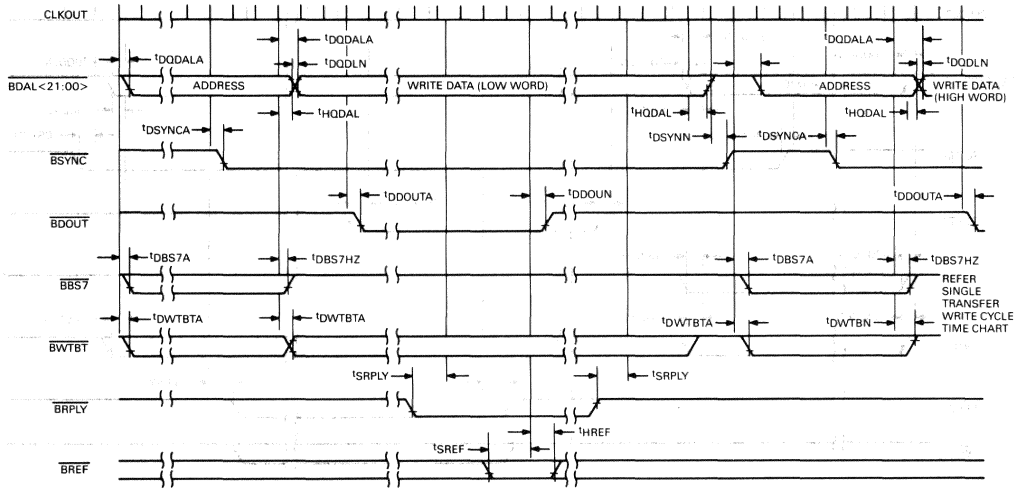


Figure 47 • CVAX 78711 CQBIC to Q22-bus Nonblock Mode Write Timing

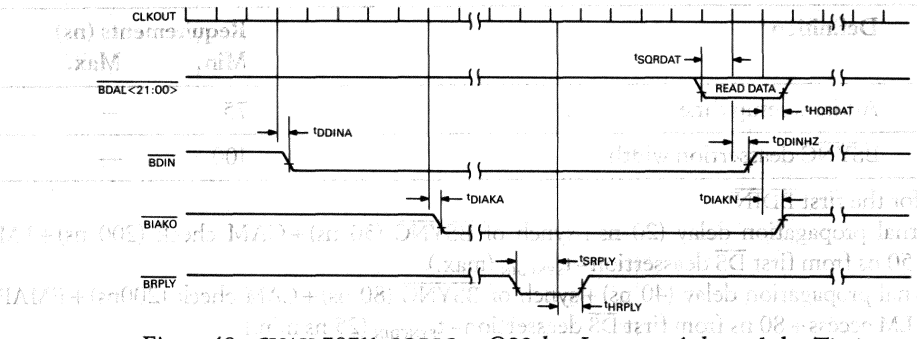


Figure 48 • CVAX 78711 CQBIC to Q22-bus Interrupt Acknowledge Timing

Table 27 • CVAX 78711 Q22-bus to CQBIC Timing Parameters

Symbol	Definition	Requirements (ns) Min.	Max.
t_{DDOUT}	Data setup time	25	—
$t_{DINNRPLYN}$	$\overline{BDIN}$ deassertion to $\overline{BRPLY}$ deassertion	30	130
$t_{DINRPLY}$	$\overline{BDIN}$ assertion to $\overline{BRPLY}$ assertion	30	130
$t_{DINRPLY0}$	$\overline{BDIN}$ assertion to $\overline{BRPLY}$ assertion[1]	A	B
$t_{DINRPLY1}$	$\overline{BDIN}$ assertion to $\overline{BRPLY}$ assertion[2]	30	160
$t_{DINRPLY2}$	$\overline{BDIN}$ assertion to $\overline{BRPLY}$ assertion[3]	C	D
$t_{DOUTNDN}$	Data hold time	25	—
$t_{DOUTRPLY}$	$\overline{BDOUT}$ assertion to $\overline{BRPLY}$ assertion	30	130
$t_{DOUTRPLY0}$	$\overline{BDOUT}$ assertion to $\overline{BRPLY}$ assertion[4]	E	F
$t_{DOUTRPLY1}$	$\overline{BDOUT}$ assertion to $\overline{BRPLY}$ assertion[5]	305	160
$t_{DOUTNRPLYN}$	$\overline{BDOUT}$ deassertion to $\overline{BRPLY}$ deassertion	30	130
t_{RPLYD}	$\overline{BRPLY}$ assertion to valid data	0	60
$t_{RPLYNDIN}$	$\overline{BRPLY}$ deassertion to $\overline{BDIN}$ deassertion	150	—
$t_{RPLYDINN}$	$\overline{BRPLY}$ assertion to $\overline{BDIN}$ deassertion	200	—
$t_{RPLYNDN}$	Data hold time	0	30
$t_{RPLYNDOUT}$	$\overline{BRPLY}$ deassertion to $\overline{BDOUT}$ assertion	150	—
$t_{RPLYNSYNC}$	$\overline{BRPLY}$ deassertion to $\overline{BSYNC}$ assertion	300	—
$t_{RPLYDOUTN}$	$\overline{BRPLY}$ assertion to $\overline{BDOUT}$ deassertion	150	—
$t_{SYNCDIN}$	$\overline{BSYNC}$ assertion to $\overline{BDIN}$ assertion	25	—
t_{SYNCH}	Address hold time	25	—

Symbol	Definition	Requirements (ns) Min.	Max.
t_{SYNCS}	Address setup time	75	—
t_{SYNCNW}	$\overline{BSYNC}$ deassertion width	100	—

[1] $\overline{BRPLY}$ for the first $\overline{BDIN}$
A = internal propagation delay (20 ns + synch of $\overline{BSYNC}$ (30 ns) + CAM check (200 ns) + LM access + 50 ns from first $\overline{DS}$ deassertion – $t_{SYNCDIN}$ (max.)
B = internal propagation delay (40 ns) + synch of $\overline{BSYNC}$ (80 ns) + CAM check (200ns) + EMAP access + LM access + 80 ns from first $\overline{DS}$ deassertion – $t_{SYNCDIN}$ (25 ns min.)
[2] $\overline{BRPLY}$ after the second $\overline{BDIN}$ and before crossing quadword boundary
[3] $\overline{BRPLY}$ for the first $\overline{BDIN}$ after crossing quadword boundary
C = synch of $\overline{BDIN}$ (30 ns) + LM access + 150 ns from $\overline{AS}$ deassertion – two $\overline{BDIN}$
D = synch or $\overline{BDIN}$ (80 ns) + LM access + 180 ns from $\overline{AS}$ deassertion – two $\overline{BDIN}$ CQBIC starts to read LM when third $\overline{BDIN}$ is asserted, therefore – two $\overline{BDIN}$
[4] $\overline{BRPLY}$ for the first $\overline{BDOUT}$
E = internal propagation delay (20 ns) + synch of $\overline{BSYNC}$ (30 ns) + CAM check (200 ns) + LM access + 150 ns from $\overline{AS}$ deassertion – $t_{SYNCDOUT}$ (max.)
F = internal propagation delay (40 ns) + synch of $\overline{BSYNC}$ (80 ns) + CAM check (200 ns) + EMAP access + LM access + 180 ns from $\overline{AS}$ deassertion – $t_{SYNCDOUT}$ (25 ns min.)
[5] $\overline{BRPLY}$ after the second $\overline{BDOUT}$

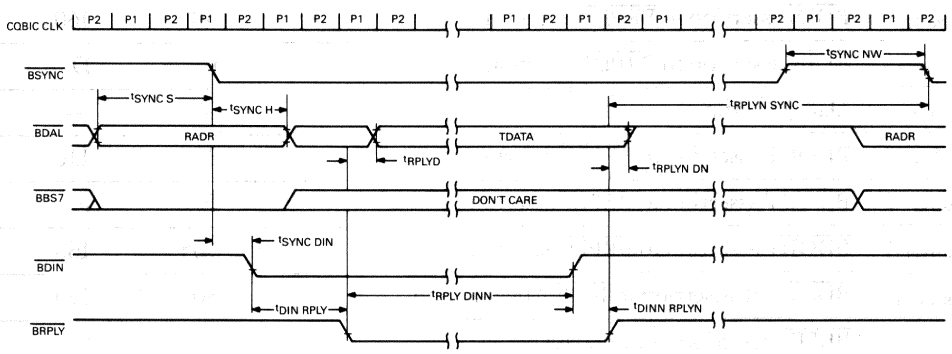


Figure 49 • CVAX 78711 Q22-bus DATI to CQBIC Doorbell Register Timing

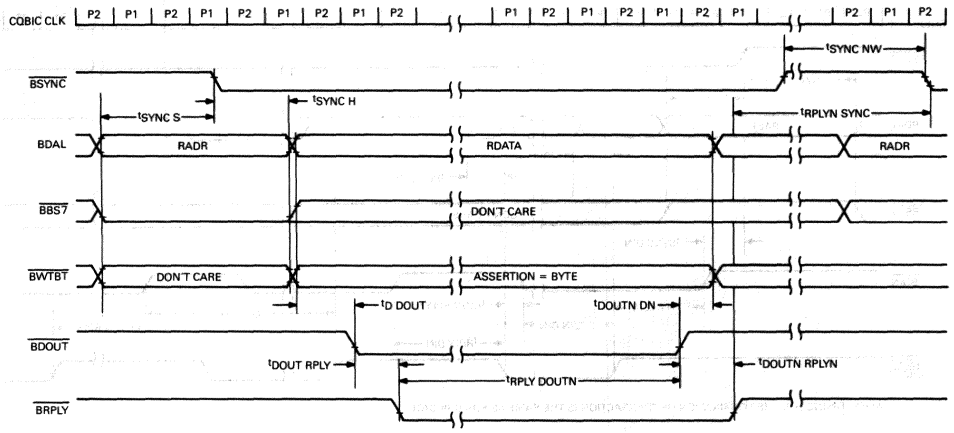


Figure 50 ▪ CVAX 78711 Q22-bus DATO and DATOB to CQBIC Doorbell Register Timing

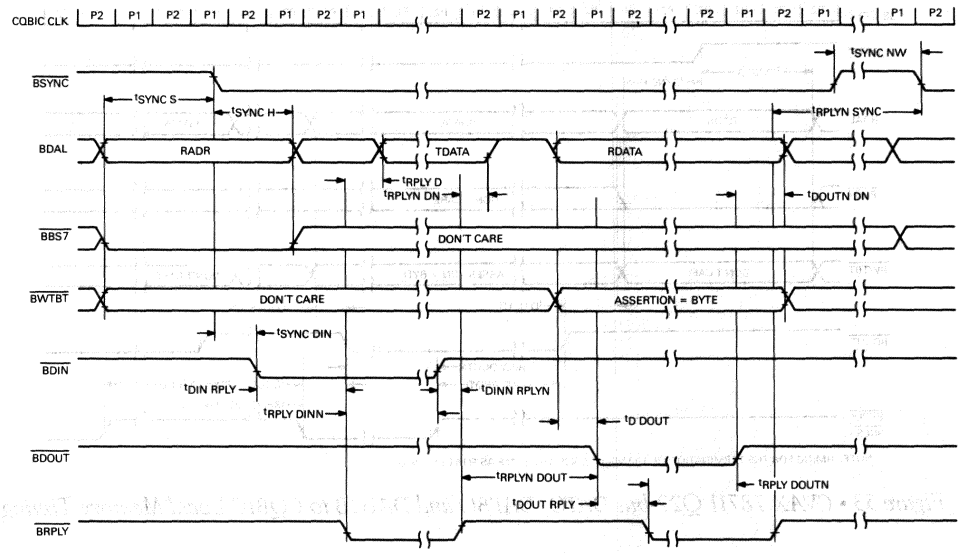
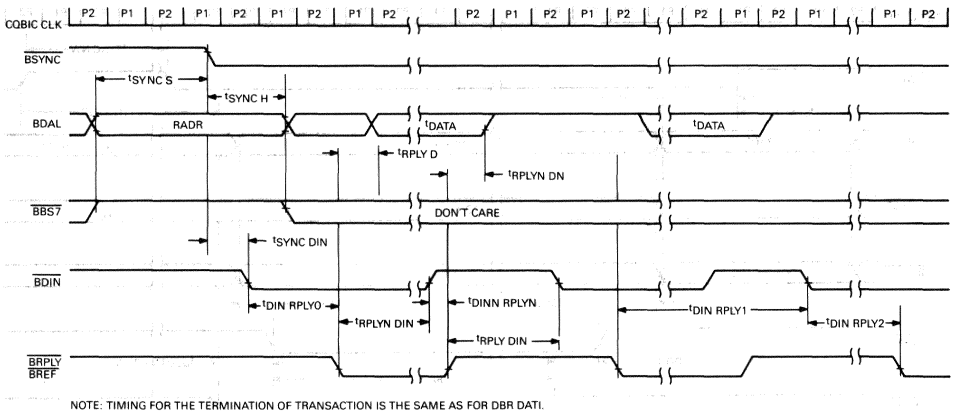
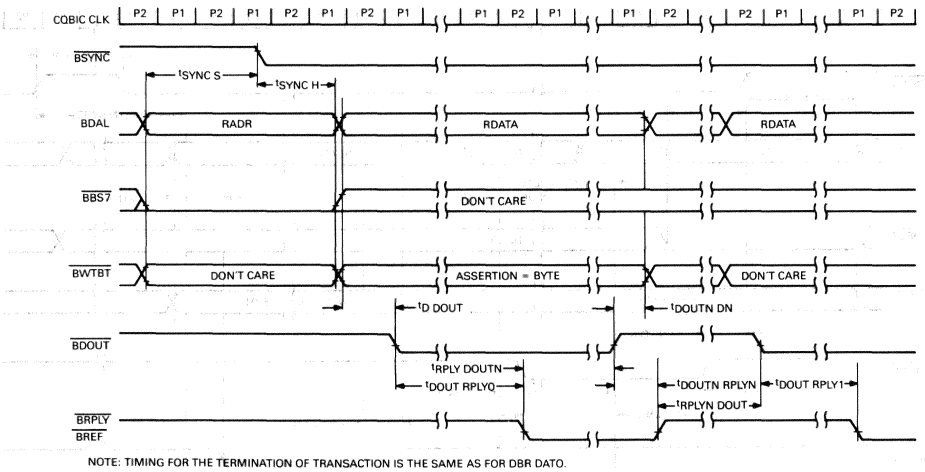


Figure 51 ▪ Q22-bus DATIO, DATIOB to CQBIC Doorbell Register or Local Memory Timing



NOTE: TIMING FOR THE TERMINATION OF TRANSACTION IS THE SAME AS FOR DBR DATI.

Figure 52 • CVAX 78711 Q22-bus DATI and DATBI to CQBIC Local Memory Timing



NOTE: TIMING FOR THE TERMINATION OF TRANSACTION IS THE SAME AS FOR DBR DATO.

Figure 53 • CVAX 78711 Q22-bus DATO, DATBO and DATOB to CQBIC Local Memory Timing

Features

- Supports VAXBI bus features of low interface cost, less than 800-nanosecond data access time, and high data integrity
- High-level integration reduces module area required
- Extensive error detection
- Complete VAXBI bus arbitration, address decoding, and matching logic to reduce hardware and software protocol
- Single 5-volt supply

Description

The DC514 CMOS VAXBI Bus Interface Chip (CBIC) is a 133-pin integrated circuit that combines the functionality of the VAXBI 78743 BCAI and VAXBI 78732 BIIC without the BCI bus lines. The CBIC is the interface between Digital's VAXBI bus and a user-developed interface of a node. It functions as a buffer file, performs bus transactions, and decodes and matches addresses. Figure 1 is a functional block diagram of the CBIC.

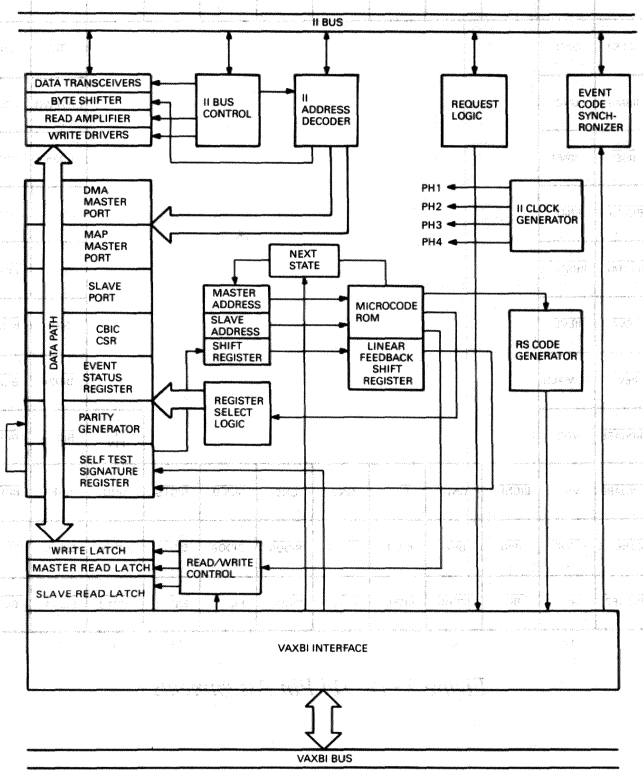


Figure 1 • DC514 VAXBI Bus Interface Chip Block Diagram

The CBIC operates with the VAXBI bus, which is a 32-bit, general purpose synchronous bus that can be used with single a processor or multi processor systems based on the VAX processors or other 32-bit processors or compatible devices. The VAXBI bus has a maximum length of 1.5 meters and connects up to 16 intelligent nodes. The combined throughput rate of the nodes is 13.3 Mbytes/second. This document assumes the reader has an understanding of the VAXBI bus and its operation. Refer to the *VAXBI System Reference Manual* (document number EK-VBISY-RM-001) for information relating to its operation.

• Pin and Signal Description

This section describes the input and output signals and power and ground connections used by the CBIC. The signal and pin assignments are shown in Figure 2 and summarized in Table 1. The signals that communicate with the user interface through the integrated circuit interconnect bus are prefixed with II. Signals that communicate with the VAXBI bus are prefixed with BI.

	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
P	BIACLO	IIP1	IIP3	IID01	IID02	IID05	IID06	IID17	IID18	IID21	IID23	IID10	IID11	VSS	P
N	IIRWEN	IIACLO	IIP0	IIP2	IID00	IID04	IID07	IID16	IID19	IID22	IID09	IID13	IID14	IID26	N
M	IICLKB	IICS	IIAH0	IIDEN	IIAH1	IID03	GND	GND	IID20	IID08	IID12	IID15	IID24	IID27	M
L	IIBM2	IICLKA	GND									VDD	IID28	IID30	L
K	IIBM0	IIBM3	GND									DRVPWR	IID29	IIAH6	K
J	IIDCLO	IISEL	IIBM1									IID25	IID31	IIAH4	J
H	IISTOP	BIDCLO	IIRAK									IIAH5	IIAH3	IIAH2	H
G	IIINCENA	IIESTAT	IIBSTAT									GND	BID29	BID31	G
F	IIRQ1	IIEV4	IIEV2									GND	BID28	BID30	F
E	IIRQ0	IIEV1	IIDMAEN									BID24	BID26	BID27	E
D	IIEV3	BIPHASE	VDD								KEY PIN	BID19	BID22	BID25	D
C	IIEV0	BINOARB	VSS	BICNF0	GND	BII3	GND	GND	BID09	BID16	GND	GREF	BID20	BID23	C
B	BITIME	BIBROKE	BICNF1	BII0	BII2	BID01	BID04	BID05	BID08	BID11	BID13	BID15	BID18	BID21	B
A	BIBUSY	BICNF2	BII1	BIP	BID00	BID02	BID03	BID06	BID07	BID10	BID12	BID14	BID17	VREF	A
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	

Figure 2 • DC514 Pin Assignments

Table 1 • DC514 Pin and Signal Summary

Pin	Signal	Input/Output	Definition/Function
J2,L1,K2,L2, M1,N1,J3,M2, P4,N5,P5,M6, N6,P6,P7,N7, M3,N2,N3,M4, P2,P3,N4,M5, N8,P8,P9,N9, M9,P10,P11, N10	IID<31:00>	Input/Output[1]	II Data Bus—Transfers data to or from the processor bus interface.
P12,N11,P13, N12	IIP<3:0>	Input/Output[1]	II Parity—Indicates parity the four bytes on the IID<31:00> lines.
K1,H3,J1,H2, H1,M10,M12	IIAH<6:0>	Input[1]	II Address—Controls the selection of the CBIC data buffer file (DBF) registers.
K13,L14,J12 K14	IIBM<3:0>	Input[1]	II Byte Mask—Specifies which IID<31:00> and IIP<3:0> lines contain valid data during a transfer.
L13	IICLKA	Input[1]	II Clock A—0 to 10 MHz external clock.
M14	IICLKB	Input[1]	II Clock B—0 to 40 MHz external clock.
M13	$\overline{\text{IICS}}$	Input[1]	II Chip Select—Initiates data transfers to or from the DBF.
M11	$\overline{\text{IIDEN}}$	Input[1]	II Data Enable—Enables the transfer of data on lines IID<31:00> and parity on lines IIP<3:0> during II bus read operations of the DBF.
N14	IIRWEN	Input[1]	II Read/Write Enable—Initiates an II bus read or write access operation to the DBF.
E12	IIDMAEN	Input[1]	II DMA/Map Enable—If asserted when a VAXBI bus request is pending, the CBIC executes an octaword transaction accessing the master-port DMA register. If deasserted when the VAXBI bus request is pending, the CBIC executes a longword transaction accessing the master-port map registers.
G14	$\overline{\text{IIINCENA}}$	Input[1]	II Increment Enable—When asserted during a VAXBI bus DMA increment enable transaction, it allows a pipelined increment of the address in the master-port DMA or map address register.

Pin	Signal	Input/Output	Definition/Function
F14,E14	$\overline{\text{IIRQ<1:0>}}$	Input[1]	II Request<1:0>—Requests a VAXBI bus transaction.
H12	$\overline{\text{IIRAK}}$	Output[1]	II Request Acknowledge—Indicates that a requested VAXBI bus master-port transaction has been initiated.
G13	$\overline{\text{IIESTAT}}$	Output[1]	II Event Status—Asserted when the ESR receives the first unmasked event code to be generated since the register was previously read.
G12	$\overline{\text{IIBSTAT}}$	Output[1]	II Bus Status—Asserted when a bit in the VAXBI bus error register is set.
J13	$\overline{\text{IISEL}}$	Output[1]	II Select—Informs the slave port that it has been selected by a VAXBI bus transaction.
H14	$\overline{\text{IISTOP}}$	Output[1]	II Stop—Informs the slave port that it has been selected by a VAXBI bus Stop transaction.
F13,D14,F12, E13,C14	$\overline{\text{IIEV<4:0>}}$	Output[1]	II Event—Indicate that a significant event in the CBIC or on the VAXBI bus has occurred.
N13	$\overline{\text{IIACLO}}$	Output[1]	II ac Low—Asserted when the line voltage is below a specified minimum level.
J14	$\overline{\text{IIDCLO}}$	Output[1]	II dc Low—Indicates an impending loss of dc power. Also used for initialization during powerup.
B13	$\overline{\text{IIBROKE}}$	Input[2]	II Broke—Used during self-test to indicate a node has failed and when to light the LED status indicators of the node.
G1,F1,G2,F2, E1,E2,D1,E3, C1,D2,B1,C2, D3,B2,A2,C5, B3,A3,B4,A4, B5,A5,C6,B6, A6,A7,B7,B8, A8,A9,B9,A10	$\overline{\text{BID<31:00>}}$	Input/Output[3]	BI Data<31:00>—Transfers data and address information to and from the VAXBI bus and performs arbitration.
C9,B10,A12, B11	$\overline{\text{BII<3:0}}$	Input/Output[3]	BI Information—Transfers commands, master identification, read status, and write masks.
A11	$\overline{\text{BIP}}$	Input/Output[3]	BI Parity—Indicates parity for the BID<31:00> and BII<3:0> lines.
C13	$\overline{\text{BINOARB}}$	Input/Output[3]	BI No Arbitration—Inhibits arbitration on lines BID<31:00>. Used during self-test to prevent a node from starting until all nodes are ready.

Pin	Signal	Input/Output	Definition/Function
A14	$\overline{\text{BIBUSY}}$	Input/Output[3]	BI Busy—Indicates a transactionis in progress.
A13,B12,C11	$\overline{\text{BICNF<2:0>}}$	Input/Output[3]	BI Confirmation—Indicates a response to command and data cycles.
P14	$\overline{\text{BIACLO}}$	Input[3]	BI ac Low—Indicates that the ac line voltage of a critical bus component is below a safe limit.
H13	$\overline{\text{BIDCLO}}$	Input[3]	BI dc Low—Indicates that the dc voltages are not within their specified limits.
B14	$\overline{\text{BITIME}}$	Input[2]	BI Timing—20 MHz square wave generated by a separate differential ECL receiver at each node. Used with the $\overline{\text{BIPHASE}}$ signal to generate VAXBI bus timing.
D13	$\overline{\text{BIPHASE}}$	Input[2]	BI Phase—5 MHz timing square wave generated by a separate differential ECL receiver at each node.
A1	VREF	Input	Voltage Reference—Reference generator resistor to V_{CC}.
C3	GREF	Input	Ground Reference—Referencegenerator resistor to ground (GND).
D12,L3	V_{DD}	Input	Voltage—5-Vdc power supply.
P1,C12	V_{SS}	Input	Ground—Common ground.
K3	DRVPWR	Input	Driver Power—5-Vdc VAXBI bus driver power.
C4,C7,C8,C10, F3,G3,K12, L12,M7,M8	GND	Input	Driver Ground—VAXBI bus driver ground.

[1]TTL compatible
[2]Open drain
[3]Standard TTL levels

II Bus Interface Signals

II Data (IID<31:00>) II Data—Three-state data lines used to transfer data between the II bus master and the CBIC data buffer file (DBF) registers.

II Parity (IIP<3:0>)—When data is transferred between the II bus master and the DBF, one parity bit for each byte of data on IID<31:00> is transferred on the IIP<3:0> lines. The data bytes and their associated parity bits are shown in Table 2. In a data read transaction, the CBIC generates and transfers parity. During a data write operation, the II bus master generates and transfers the parity. These are three-state lines.

Table 2 • DC514 II Data Parity Bit Assignments

Data byte	Parity bit
IID<31:24>	IIP3
IID<23:16>	IIP2
IID<15:08>	IIP1
IID<07:00>	IIP0

II Address (IIAH<6:0>)—During II bus read or write accesses to the DBF, the II bus master transfers a 7-bit address on theses lines to select the register to be accessed. When accessing the dual-octaword data buffer, this address selects the first byte of a longword access. The dual-octaword data buffer registers can be accessed on any even or noneven longword boundary. A noneven aligned access to the eighth register wraps to the first register in the dual-octaword space, providing a circular address space. When accessing any other DBF register, only the upper five address bits are significant. The lower two bits are assumed to be zeros.

II Byte Mask (IIBM<3:0>)—During II bus read and write accesses to the CBIC DBF, these inputs specify which bytes of the data lines (IID<31:00>) and which bits of the IIP<3:0> lines contain valid information as listed in Table 3. In write accesses, any bytes that are not specified as being valid will not be written. During read accesses, bytes not specified as being valid appear as zeros on lines IID<31:00>, with correct parity generated on lines IIP<3:0>. By using the information on the IIBM<3:0> and IIAH<6:0> lines, every byte in the DBF can be accessed. Therefore, 8-, 16-, or 32-bit processors can be easily interfaced to the CBIC.

Table 3 • DC514 II Byte Mask Assignments

IIBM Line*				Valid data	Valid parity
3	2	1	0		
L	L	L	L	IID<31:00>	IIP<3:0>
H	H	H	L	IID<07:00>	IIP0
H	H	L	H	IID<15:08>	IIP1
H	L	H	H	IID<23:16>	IIP2
L	H	H	H	IID<31:24>	IIP3

*L=low level, H=high level. All other binary input combinations that specify the validity of two or three bytes on IID<31:00> are allowed.

II Clock A (IICLKA)—Input clock frequency that must be provided by the user. The CBIC is fully static, therefore the clock frequency requirement is from 0 to 10 MHz maximum. The IICLKA input generates the internal four-phase clock of the CBIC, which controls the II bus interface. This signal is synchronous with the IICLKB input and with all II bus accesses to the CBIC.

II Clock B (IICLKB)—An input clock frequency of four times the frequency of IICLKA that must be provided by the user. This frequency is from 0 to 40 MHz maximum. The IICLKB input generates the internal four-phase clock of the CBIC that controls the II bus interface. This signal is synchronous with the IICLKA input and with all II bus accesses to the CBIC.

II Chip Select ($\overline{\text{IICS}}$)—The II bus master asserts this input to initiate the II bus read and write accesses to the DBF. In addition, the IICLKA input cycle of from t_0 to t_{100} immediately preceding the assertion of this input defines the address subcycle of an II bus access. The t_{100} of the address subcycle defines the deasserting edge of an address strobe signal and latches the address and byte mask information on lines IIAH<6:0> and IIBM<3:0> by the II bus master.

II Data Enable (IIDEN)—The II bus master asserts this input during II bus read accesses to the DBF to enable the CBIC to transfer data and parity to lines IID<31:00> and IIP<3:0>. During II bus write accesses to the DBF, the input IICLKA cycle of from t_0 to t_{100} immediately preceding the assertion of the IIDEN input, defines the data subcycle of an II bus access. The t_{100} of that data subcycle defines the deasserting edge of a data strobe, latching data and parity values placed on lines IID<31:00> and IIP<3:0> by the II bus master.

II Read/Write Enable (IIRWEN)—During t_{100} of an address subcycle, the II bus master asserts this input to initiate an II bus read access to the DBF and deasserts it for an II bus write access.

II DMA/Map Port Enable (IIDMAEN)—If asserted during a VAXBI bus transaction request to the II bus master, the CBIC executes an octaword VAXBI bus transaction to access the master-port DMA registers for the data, address, and command information. If this signal is deasserted during a VAXBI bus transaction request to the II bus master, the CBIC executes a longword VAXBI bus transaction accessing the master-port map registers data, address, and command information.

II Increment Enable (IIINCENA)—When asserted during the request and execution of a DMA VAXBI bus transaction, this input enables a pipelined increment of the address in the master-port DMA address register to occur. The next octaword transaction to be requested and executed by the CBIC accesses the next sequential octaword in VAXBI bus memory. This operation eliminates the need for the II bus master to update the master-port DMA address register for each octaword transaction of a block move operation. When asserted during a map VAXBI bus transaction, this signal performs a similar function with the master-port map address register. The address of the next master-port map transaction is incremented by a longword instead of resulting in an octaword increment.

II Request (IIRQ<1:0>)—These inputs are asserted by the II bus master to request a VAXBI bus transaction that executes a CBIC transaction. When the IIRQ1 input is asserted, a loopback transaction is requested. This is used only when accessing a CBIC node or user CSR space through the longword master-port map read or write VAXBI bus transactions. Asserting the IIRQ1 and IIRQ0 inputs selects the CBIC diagnostic mode.

II Request Acknowledge (IIRAK)—The CBIC asserts this output to indicate that a requested VAXBI bus master-port transaction has been initiated. This output is deasserted when the transaction has been completed. Transaction requests for the next VAXBI bus transaction can be initiated before the deassertion of the acknowledgment of the current VAXBI bus transaction. This output is synchronous with the VAXBI bus clocks.

II Event Status (IIESTAT)—This output is asserted when the event status register has captured the first unmasked event code since the previous reading of the register and deasserted when the event status register is read. The CBIC synchronizes the assertion and deassertion of this output with the IICLKA and IICLKB clock signals.

II Bus Status (IIBSTAT)—This input is asserted when an error is detected during a loopback or VAXBI bus transaction causing a bit to be set in the VAXBI Bus Error Register (BER). It is deasserted when the BER is cleared. The BER can be cleared by the II bus master or by another node on the VAXBI bus. The II bus master clears the BER by performing a master-port map loopback longword transaction. The assertion and deassertion of this input is synchronized with the IICLKA and IICLKB clock inputs. The BER is described in the *VAXBI System Reference Manual*.

II Select (IISEL)—The CBIC asserts this output when a VAXBI bus transaction selects the slave-port interface on the II bus. The BCI Control and Status Register (BCICSR) in the CBIC allows the user interface to create a customized subset of VAXBI bus transactions that select the slave port in

this node. As an example, nodes that are not to respond to multicast space read- or write-type commands can clear the MSEN bit 15 in BCICSR. Refer to the *VAXBI System Reference Manual* for a description of the BCICSR. Therefore, the $\overline{\text{IISEL}}$ is asserted upon the receipt of a CBIC transaction that addresses multicast space. It is asserted for the following conditions:

- When a read- or write-type command has been received whose address is in the range of the starting and ending address registers as defined in the *VAXBI System Reference Manual.*
- A read- or write-type command has been received whose address is in the range of multicast space and the MSEN bit 15 in the BCICSR is set.
- An IDENT command has been received and the IDENTEN bit 11 in the BCICSR is set.
- A BDCST command directed to this node has been received and the BDCSTEN bit 17 in the BCICSR is set.
- A Stop command directed at this node has been received, and the STOPEN bit 13 in the BCICSR is set. In this case, the $\overline{\text{IISTOP}}$ output is also asserted simultaneously with the $\overline{\text{IISEL}}$ output.
- A Reserved command is received and the RESEN bit 12 in the BCICSR is set.
- An IPINTR command directed at this node and matching the IPINTR Mask register has been received and the IPINTREN bit 05 in the BCICSR is set.
- An INTR command directed at this node has been received and the INTREN bit 06 in the BCICSR is set.
- An INVAL command or a write-type command not directed to the range of addresses defined by the starting and ending address registers has been received and the INVALEN bit 10 or WINVALEN bit 09 in the BCICSR is set.
- A read- or write-type command matches the user interface CSR space of this node and the UCSREN bit 08 in the BCICSR is set.

If the SCSYNC bit 26 of the CBIC CSR is not set, the $\overline{\text{IISEL}}$ output is synchronously asserted by the CBIC with respect to the IICLKA and IICLKB clock inputs and remains asserted for one or more succeeding IICLKA cycles.

If the SCSYNC bit 26 is set, the $\overline{\text{IISEL}}$ output is synchronously asserted with respect to the VAXBI bus and remains asserted for one VAXBI bus cycle. The user must synchronize to the node's clock. This mode can be used when the system clock is significantly slower than the VAXBI bus $\overline{\text{BIPHASE}}$ clock.

II Stop ($\overline{\text{IISTOP}}$)—This output is asserted when a Stop command has been received and the STOPEN bit 13 is set in the BCICSR. The $\overline{\text{IISTOP}}$ output is asserted for one or more succeeding IICLKA cycles or one VAXBI bus cycle depending on the state of the SCSYNC bit 26 in the CBIC CSR. It is coincident with the $\overline{\text{IISEL}}$ output.

II Event ($\overline{\text{IIEV}}$<4:0>)—These outputs indicate significant events have occurred in the CBIC or on the VAXBI bus. The event codes are described in the *VAXBI System Reference Manual.* The octal code on the $\overline{\text{IIEV}}$<4:0> lines correspond to the bit position in the event status register (ESR). (Example: Octal code 30 represents the event defined by bit 30 of the ESR described in this document.)

If the EVSYNC bit 27 of the CBIC CSR is not set, the information on lines $\overline{\text{IIEV}}$<4:0> is synchronized with the IICLKA and IICLKB inputs by the CBIC. The information remains for one or more succeeding IICLKA cycles. If the EVSYNC bit is set, the event information is generated

synchronously with the VAXBI bus information and remains for one VAXBI bus cycle. The user must provide synchronization of the node clock. This mode can be used for nodes when the system clock is slower than the VAXBI bus BIPHASE clock input.

II ac Low (IIACLO)—This output is asserted when the line voltage is below the minimum level specified. It performs the same function as the BCIACLO signal of the BIIC which is defined in the *VAXBI System Reference Manual.*

II dc Low (IIDCLO)—This output is asserted to indicate that a dc power loss will occur. It is used for initialization during the powerup sequence. It performs the same function as the BCIDCLO signal of the BIIC which is defined in the *VAXBI System Reference Manual.*

VAXBI Bus Interface Signals

The following signals connect to the VAXBI bus. Most signals can be connected directly and the VAXBI bus provides a pullup resistor for each signal. The signals that require an open-drain circuit between the signal and VAXBI bus are indicated. Refer to the *VAXBI System Reference Manual* for a more complete description of these signals.

BI Data (BID<31:00>)—These bidirectional lines are the primary information path of the VAXBI bus. All address and data transfers and arbitration sequences occur on these lines.

BI Information (BII<3:0>)—These bidirectional lines transfer commands, encoded master identification, read status codes, and write masks. Commands can be directed to one or more nodes depending on the type of command. The command codes and types are listed in Table 4.

Table 4 • DC514 Command Code Assignments

BII Line 3	2	1	0	Type*	Command/Description
H	H	H	H	—	Reserved
H	H	H	L	SR	Read
H	H	L	H	SR	RCI/Read with cache intent
H	H	L	L	SR	IRCI/Interlock read with cache intent
H	L	H	H	SR	Write
H	L	H	L	SR	WCI/Write with cache intent
H	L	L	H	SR	UWMCI/Unlock write mask with cache intent
H	L	L	L	SR	WMCI/Write mask with cache intent
L	H	H	H	MR	INTR/Interrupt
L	H	H	L	SR	IDENT/Identify
L	H	L	H	—	Reserved
L	H	L	L	—	Reserved
L	L	H	H	MR	Stop
L	L	H	L	MR	INVAL/Invalidate
L	L	L	H	MR	BDCST/Broadcast (reserved)
L	L	L	L	MR	IPINTR/Interprocessor interrupt

*SR is a single responder and MR is more than one responder.

BI Parity (BIP)—A bidirectional signal that indicates the parity of the BID<31:00> and BII<3:0> information. It is asserted to generate odd parity if the sum of asserted bits in these two fields is an even number.

BI No Arbitration ($\overline{\text{BINOARB}}$)—A bidirectional signal that is asserted to inhibit using the BID<31:00> line information when nodes are arbitrating for control of the VAXBI bus. It is also used during the CBIC self-test program to prevent other nodes from starting transactions until all nodes are ready to participate.

BI Busy ($\overline{\text{BIBUSY}}$)—A bidirectional signal that is asserted to indicate that a transaction is in progress.

BI Confirmation ($\overline{\text{BICNF<2:0>}}$)—These bidirectional lines contain the response to command and data cycles.

BI ac Low ($\overline{\text{BIACLO}}$)—This input indicates that the ac line voltage of a critical bus component is below a specified minimum level.

$\overline{\text{BI}}$ dc Low ($\overline{\text{BIDCLO}}$)—This input indicates that the dc voltages are not within their specified limits.

BI Broke ($\overline{\text{BIBROKE}}$)—This input drives the $\overline{\text{BIBAD}}$ line of the VAXBI bus to inform the systems on the VAXBI bus that a self-test failure of a node has occurred. It is also used to determine when the status LED indicators of a node will be lighted. An open-drain buffer circuit is required when connecting this signal to the VAXBI bus.

BI Timing ($\overline{\text{BITIME}}$)—This input is a 20-MHz square-wave signal that is generated by an external differential ECL receiver at each node. This input and the $\overline{\text{BIPHASE}}$ input are used by the CBIC to generate all the required VAXBI bus synchronous timing signals. An open-drain buffer circuit is required when connecting this signal to the VAXBI bus.

BI Phase ($\overline{\text{BIPHASE}}$)—A 5-MHz square-wave input that is generated by an external differential ECL receiver device at each node. It is used with the $\overline{\text{BITIME}}$ input to generate all required VAXBI bus synchronous timing signals. An open-drain buffer circuit is required to connect this signal to the VAXBI bus.

• Standard VAXBI Node Registers

The CBIC contains standard node registers that are defined in the *VAXBI System Reference Manual* and listed in Table 5. The CBIC register functions that are different from those defined in the *VAXBI System Reference Manual* are described.

Table 5 • DC514 Standard VAXBI Node Registers

Register	Mnemonic	Address*
Device	DTYPE	bb+0
VAXBI Control and Status	VAXBICSR	bb+4
Bus Error	BER	bb+8
Error Interrupt Control	EINTRCSR	bb+C
Interrupt Destination	INTRDES	bb+10
Interprocessor Interrupt Mask	IPINTRMSK	bb+14
Force-bit IPINTR/STOP Destination	FIPSDES	bb+18
Interprocessor Interrupt Source	IPINTRSIC	bb+1C
Starting Address	SADR	bb+20
Ending Address	EADR	bb+24
BCI Control and Status	BCICSR	bb+28
Write Status	WSTAT	bb+2C
Force-bit IPINTR/STOP command	FIPSDES	bb+30

Register	Mnemonic	Address*
User Interface Interrupt Control	UINTRCSR	bb+40
Bus Error Mask	BEMR	bb+48
General Purpose 0	GPR0	bb+F0
General Purpose 1	GPR1	bb+F4
General Purpose 2	GPR2	bb+F8
General Purpose 3	GPR3	bb+FC

*bb is the base address of the first location of the nodespace. The Bus Error Mask Register (BEMR) is implemented in the CBIC but not defined in the *VAXBI System Reference Manual.*

Bus Error Register—The User Parity Enable (UPEN) bit 03 of the CBIC register is not writable and read as a zero.

Bus Error Register—The User Parity Enable (UPEN) bit 03 of the CBIC register is not writable and read as a zero.

Bus Error Mask Register—Contains a bit-for-bit correspondence with the Bus Error Register (BER). Setting a bit in this register inhibits the assertion of the $\overline{\text{IIBSTAT}}$ output when the corresponding bit in the BER is set thereby disabling the interrupt request.

VAXBI Control and Status Register—The Broke bit 12 determines the state of the $\overline{\text{BIBROKE}}$ output of the CBIC. It is a read/write (R/W) bit.

User Interface Interrupt Control Register—The External Vector (EXVECTOR) bit 15 of the CBIC register is not writable and is read as zero. Internal vectors are provided in response to IDENT transactions only.

BCI Control and Status Register—The BIIC CSR Space Enable (BICSREN) bit 07 of the CBIC register is not writable and is read as a zero. Accesses to the BIIC CSR space are processed internally to the CBIC.

Data Buffer File Registers

The Data Buffer File (DBF) contains additional registers to the standard VAXBI Registers. These are the master-port registers, slave-port registers, control and status registers, and valid-bit-clear-on-read register. The hexadecimal address assignments and read/write capabilities of each register are shown. Refer to the *VAXBI System Reference Manual* for registers referred to but not described in this document.

Master-port Registers

Master-port registers are used for II bus-initiated transfers to the VAXBI bus. The CBIC contains high-speed DMA master ports optimized for block data transfers and a map master port. Both the DMA master ports and the map master port have a command/address register with autoincrement capability and page-cross detection. A local processor can perform longword accesses to the VAXBI bus in the middle of a block DMA transaction without storing the state of the previous transaction.

DMA Master-port registers—The DMA master port consists of a Port A octaword buffer and a Port B octaword buffer. It also contains an address register, command register, a next-page-frame register, and eight valid-bit registers as shown in Figure 3.

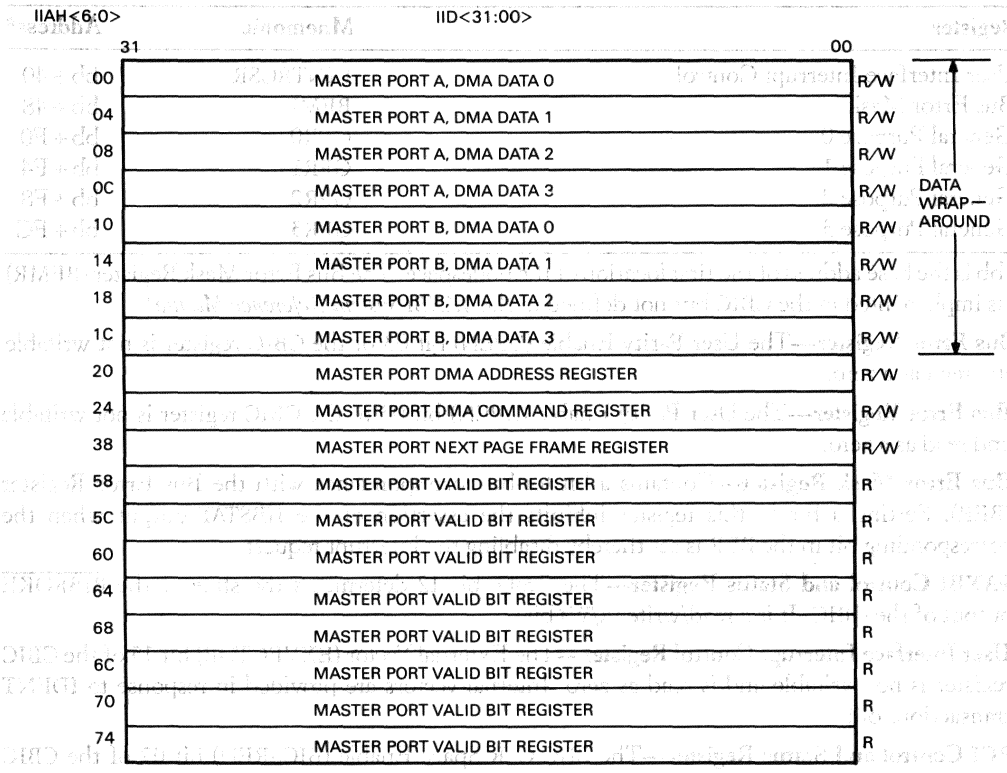


Figure 3 • DC514 Master-port DMA Registers

The Master-port DMA data registers store eight contiguous longwords in read/write memory. The registers are designated Master-port A registers (Data 0—Data 3) and Master-port B registers (Data 0—Data 3). The longwords are organized into the octaword data buffers. The CBIC supports all possible address alignments to these buffers by using any four sequential bytes of the two octawords referred to as a transaction buffer. One transaction buffer may be accessed by an II bus transaction while the other is accessed by the CBIC master control device to generate a VAXBI bus transaction. If an overflow occurs when reading or writing from either octaword, it is automatically directed to the first bytes of the other octaword. For example, the fourth longword of an unnaturally aligned octaword transaction will extend into the first three bytes of the remaining octaword.

The Master-port DMA Address register contains the address for the command/address cycle of a VAXBI bus master-port transaction during a DMA operation. If the $\overline{\text{IIINCENA}}$ input is asserted during the transaction request, the lower 9-bits of the address are incremented by 16. For the next master-port DMA transaction, this register contains the address of the next sequential octaword in VAXBI bus memory. When executing block DMA transfers, the increment feature eliminates the need of the II bus master to reload the DMA address before requesting the next VAXBI bus transaction.

The Master-port DMA Command register contains the VAXBI bus command for the command/address cycle of a VAXBI bus master-port transaction during a DMA operation. The 4-bit command is written into bits 19:16 of this register as shown Figure 4.

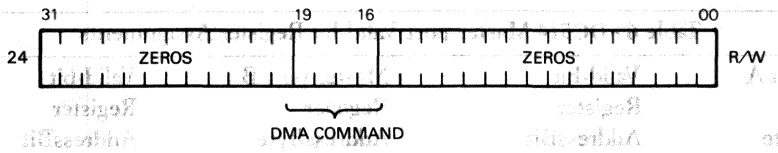


Figure 4 • DC514 Master-port DMA Command Register Format

During II bus write operations to this register, only the bits asserted on data lines IID<19:16> are written. The CBIC checks for correct parity only on these lines so that the value on IIP2 is a calculation for the four command bits. The parity on the remaining IIP3 and IIP<1:0> lines are not significant.

During II bus read operations of this register, bits 31:20 and 15:00 are read as zeros. The CBIC generates correct parity on lines IIP<3:0> for the entire longword.

The Master-port Next Page-frame Register (NPFR) holds the map for the next page. It is preloaded by the II bus master with the highest address of the next physical page to be accessed during a DMA block move operation after the DMA address register has been incremented beyond a page boundary. When the DMA address register reaches a page boundary, bits 31:09 of the NPFR are transferred to the Master-port DMA address register bits 31:09. This feature can increase the data throughput during block DMA transactions. Instead of halting while waiting for a new map, VAXBI bus transactions can continue for up to a page while the II bus master fetches and loads the next map, via transactions through the master-port map. Bits 08:00 of this register are not transferred.

During II bus write operations to this register, the information on data lines IID<08:00> is not written. The CBIC checks for correct parity on lines IID<31:09> and the parity is indicated on lines IIP<3:1>. The IIP0 line value is not significant.

During II bus read operations, bits 08:00 of this register are read as zeros. The CBIC generates correct parity for the entire longword on lines IIP<3:0>.

One master-port valid-bit register is assigned to each of the eight master-port data registers in the DBF. Figure 5 shows the master-port valid-bit register formats. Bits 19 through 16 contain the valid-bit information related to each byte of data.

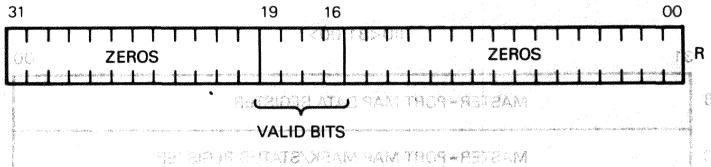


Figure 5 • DC514 Master-port Valid-bit Register Format

Valid bits are set when data is written into byte locations in the data register if the byte mask input IIBM<3:0> that corresponds to the data location indicates that the data is valid. Table 6 lists the byte locations and their corresponding valid bits.

Table 6 • DC514 Master-port Valid-bit Register Assignments

Master-port A Register AddressByte		Valid-bit Register AddressBit		Master-port B Register AddressByte		Valid-bit Register AddressBit	
00	07:00	58	16	10	07:00	68	16
	15:08		17		15:08		17
	23:16		18		23:16		18
	31:24		19		31:24		19
04	07:00	5C	16	14	07:00	6C	16
	15:08		17		15:08		17
	23:16		18		23:16		18
	23:16		18		31:24		19
08	07:00	60	16	18	07:00	70	16
	5:08		17		15:08		17
	23:16		18		23:16		18
	31:24		19		31:24	19	
0C	07:00	64	16	1C	07:00	74	16
	15:08		17		15:08		17
	23:16		18		23:16		18
	31:24		19		31:24		19

The valid bit is used as the data mask on the $\overline{BII<3:0>}$ lines during the data cycle of a VAXBI bus UWMCI or WMCI write transaction. The valid bits are accessible to the II bus as read-only locations in bits 19:16 of addresses 58 to 74 (hexadecimal). A valid bit is cleared when its corresponding byte is accessed by the CBIC master control device in supplying data for a VAXBI bus octaword write transaction. All valid bits are cleared by a II bus read transaction from location 7C and following the self-test of the CBIC.

Map Master-port Registers—The map master port contains a longword data register, a mask/status register, an address register, and a command register as shown in Figure 6.

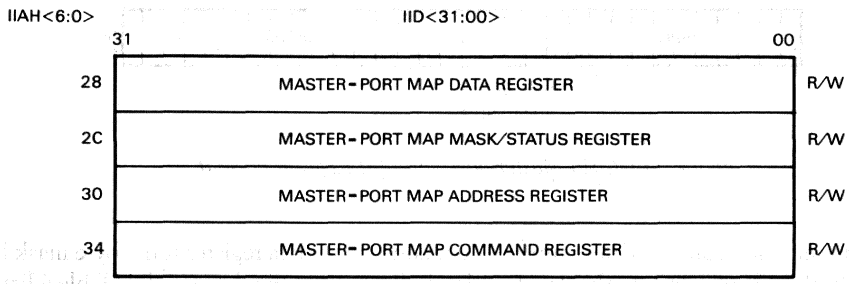


Figure 6 • DC514 Master-port Map Registers

The Master-port Map Data register stores a longword of data during read or write master-port map transactions.

The Master-port Map Mask/Status register contains mask information during map write transactions and status information during map read transactions. Figure 7 shows the register format. During VAXBI bus UWMCI and WMCI transactions to the master port, bits 19:16 of this register are preloaded by the II bus master with the 4-bit mask associated with the data in the master-port DMA data register. During II bus write operations to this register, only the information on data lines IID<19:16> is written. The CBIC checks for correct parity on data IID<19:16> and the mask bit parity is indicated on line IIP2. The parity on lines IIP<3,1:0> is not significant.

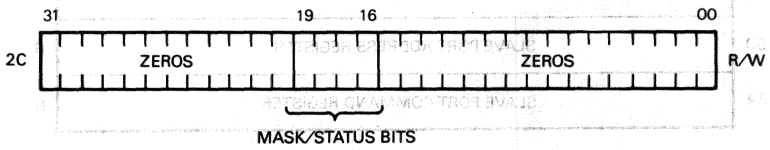


Figure 7 • DC514 Master-port Map Mask/Status Register Format

Following a VAXBI bus map read transaction to the master port, bits <19:16> contain the 4-bit read status code for the data in the master-port DMA data register. During II bus read operation of this register, bits 31:20 and 15:00 are read as zeros. The parity for the entire longword is indicated by the CBIC on lines IIP<3:0>.

The Master-port Map Address register provides the address for the command/address cycle of a VAXBI bus master-port transaction during a map port operation. If the $\overline{\text{IIINCENA}}$ input is asserted during the transaction request, bits 08:00 of the address are incremented by a count of 4. For the next master-port map transaction, this register contain the address of the next sequential longword in VAXBI bus memory. This feature eliminates the need of the II bus master to reload the map address before requesting the next transaction.

The Master-port Map Command register provides the VAXBI bus command for the command/address cycle of a VAXBI bus master-port transaction during a map operation. The register format is shown in Figure 8. Bits 19:16 of this register contain the command information.

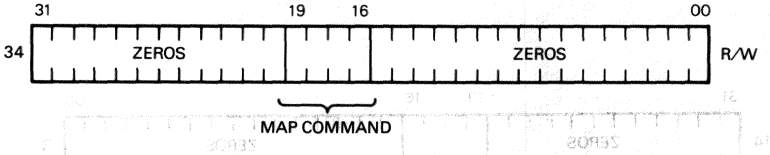


Figure 8 • DC514 Master-port Map Command Register Format

During II bus write operations to this register, only the information on data lines IID<19:16> is written. The CBIC checks this information for correct parity and the parity of the four command bits is indicated on line IIP2. The parity on the IIP3 and IIP<1:0> lines are not significant.

During II bus read operations of this register, bits 31:20 and 15:00 are read as zeros. The parity for the longword is generated by the CBIC and indicated on the IIP<3:0> lines.

Slave-port Registers

The Slave-port registers, shown in Figure 9, are used to respond to slave-port interface transactions.

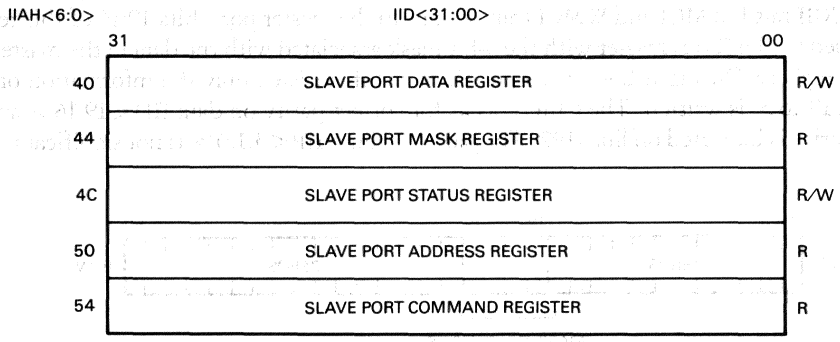


Figure 9 • DC514 Slave-port Registers

The Slave-port Data register stores one longword of data during read or write slave-port transactions. The VAXBI bus transaction in process are normally extended. When this register is accessed from the II bus, the VAXBI bus transactions are terminated. During slave read transactions, the extension of the transaction provides time for the II bus master to read the slave-port command and address registers, to access local memory for the required read data, and to write the data into the slave-port data register. During slave-port write transactions, extending the current transaction prevents the execution of subsequent slave transactions that would result in the overwriting of data in the slave-port registers before being read by the II bus master. This register should be accessed by the II bus master as soon as possible to prevent an excessive extension of the VAXBI bus transactions.

The Slave-port Mask register, shown in Figure 10, contains the mask bits associated with the data in the Slave-port Data register. After receiving a UWMCI and WMCI VAXBI bus write transactions to the the slave port, bits 19:16 of this register contain a 4-bit write mask code associated with the write data in the slave-port data register. During II bus read operations of this register, bits 31:20 and 15:00 are read as zeros. The CBIC generates correct parity for the entire longword on lines IIP<3:0>.

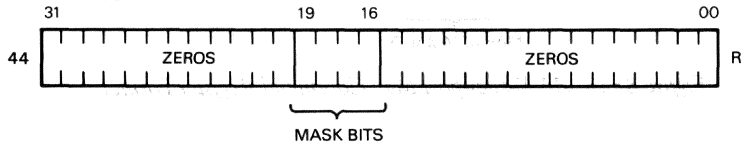


Figure 10 • DC514 Slave-port Mask Register Format

The Slave-port Status register contains the status information in the Slave-port data register during VAXBI bus slave port read transactions. The status information is preloaded by the II bus master in bits 19:16 as shown in Figure 11. During II bus write operations to this register, only the information on data lines IID<19:16> is written. The CBIC checks for correct parity only on these lines and indicates the parity on line IIP2. Parity bits IIP3 and IIP<3:0> are not significant. A read response code must be written to this register by the II bus master if the read response code changes from the previous slave read transaction.

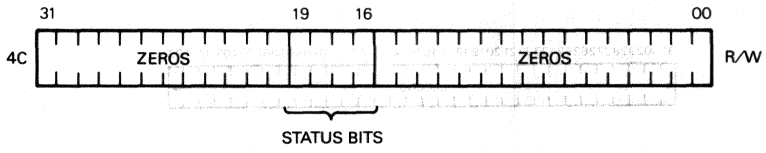


Figure 11 • DC514 Slave-port Status Register Format

The Slave-port Address register stores the address from the command/address cycle of a VAXBI bus slave-port transaction.

The Slave-port Command register stores the command for the command/address cycle of a VAXBI bus slave-port transaction. The command is stored in bits 19:16 as shown in Figure 12. During II bus read operations to this register, bits 31:20 and 15:00 are read as zeros. The CBIC generates correct parity for the entire longword on lines IIP<3:0>.

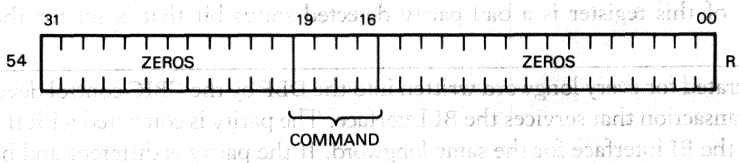


Figure 12 • DC514 Slave-port Command Register

Event and CBIC Control and Status Registers

The CBIC contains an Event Status Register, an Event Status Mask Register and a Control and Status Register shown in Figure 13.

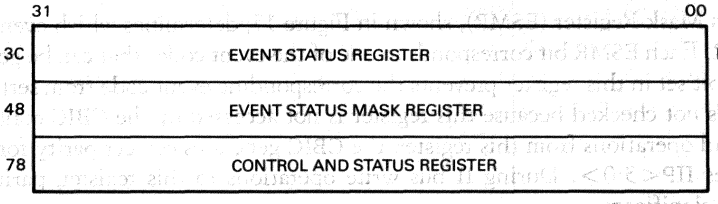
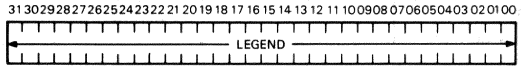


Figure 13 • DC514 Event and CBIC Control and Status Registers

The Event Status Register (ESR) stores the first unmasked event code to be generated since the register was previous read. When the event code is received, the $\overline{\text{IIESTAT}}$ output is asserted. If the first generated event code has a corresponding mask bit set in the Event Status Mask register (ESMR), then the bit in the ESR will not be set and $\overline{\text{IIESTAT}}$ will remain deasserted. The next event code to be generated that does not have a corresponding bit set in the ESMR will cause the appropriate bit in the ESR to be set and will assert the $\overline{\text{IIESTAT}}$ output. One bit in the ESR is assigned to each event code that can be generated on the event code lines $\overline{\text{IIEV}}<4:0>$. Figure 14 shows the register format.



LEGEND

00	MASTER TRANSMIT CHECK ERROR	16	EXTERNAL VECTOR SELECTED, LEVEL 7
01	BAD PARITY RECEIVED IN MASTER TRANSACTION	17	EXTERNAL VECTOR SELECTED, LEVEL 6
02	RETRY TIMEOUT	18	EXTERNAL VECTOR SELECTED, LEVEL 5
03	ILLEGAL CNF RECEIVED BY MASTER PORT IN DATA CYCLE	19	EXTERNAL VECTOR SELECTED, LEVEL 4
04	BAD PARITY RECEIVED	20	IDENT ARB LOST
05	NO ACK CNF RECEIVED FOR MASTER PORT COMMAND	21	ACK CNF RECEIVED FOR ERROR VECTOR
06	ILLEGAL CNF RECEIVED FOR MASTER PORT COMMAND	22	NO ACK OR ILLEGAL CNF RECEIVED FOR FORCE-BIT IPINTR/STOP COMMAND
07	READ DATA SUBSTITUTE OR RESERVED STATUS CODE RECEIVED	23	NO ACK OR ILLEGAL CNF RECEIVED FOR INTR COMMAND
08	ACK CNF RECEIVED FOR NON-ERROR VECTOR, LEVEL 4	24	ADVANCED RETRY CNF RECEIVED
09	ACK CNF RECEIVED FOR NON-ERROR VECTOR, LEVEL 5	25	INTERNAL REGISTER WRITTEN
10	ACK CNF RECEIVED FOR NON-ERROR VECTOR, LEVEL 6	26	RETRY CNF RECEIVED FOR MASTER PORT COMMAND
11	ACK CNF RECEIVED FOR NON-ERROR VECTOR, LEVEL 7	27	SELF-TEST PASSED
12	BUS BSY ERROR	28	BUS TIMEOUT
13	ILLEGAL CNF RECEIVED FOR SLAVE DATA	29	ACK RECEIVED FOR SLAVE READ DATA
14	BAD PARITY RECEIVED DURING SLAVE TRANSACTION	30	MASTER PORT TRANSACTION COMPLETE
15	STALL TIMEOUT ON SLAVE TRANSACTION	31	BAD PARITY DETECTED STATUS

Figure 14 • DC514 Event Status Register Format

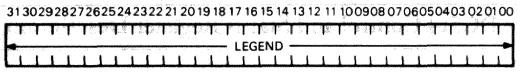
Reading the ESR clears the register for the next unmasked event code and deasserts the $\overline{\text{IIESTAT}}$ output. Bit 31 of this register is a bad parity detected status bit that is set for the following conditions:

- Parity is generated for every longword written into the DBF by the CBIC control device during a VAXBI bus transaction that services the BI Interface. The parity is compared with the parity bit generated by the BI interface for the same longword. If the parity is different and bit 31 in the ESMR is not set, then bit 31 in the ESR will be set and the $\overline{\text{IIESTAT}}$ output will be asserted.

- Parity is generated for every byte read from the DBF by the CBIC control device during VAXBI bus transactions that service the BI interface. This parity is compared with the parity stored with the byte when it was written by the II bus master or slave. If the parity is different and bit 31 in the ESMR is not set, bit 31 in the ESR will be set and the $\overline{\text{IIESTAT}}$ output will be asserted.

During II bus read operations to this register, the CBIC generates the parity for the entire longword on lines IIP<3:0>.

The Event Status Mask Register (ESMR), shown in Figure 15, determines which event codes are stored by the ESR. Each ESMR bit corresponds to one of the event codes that can be generated on $\overline{\text{IIEV}}<4:0>$. A bit set in this register prevents the corresponding event code from setting a bit in the ESR. Parity is not checked because this register is not accessed by the CBIC control device. During II bus read operations from this register, the CBIC generates correct parity for the entire longword on lines IIP<3:0>. During II bus write operations to this register, parity on lines IIP<3:0> is not significant.



LEGEND

00	MASTER TRANSMIT CHECK ERROR	16	EXTERNAL VECTOR SELECTED, LEVEL 7
01	BAD PARITY RECEIVED IN MASTER TRANSACTION	17	EXTERNAL VECTOR SELECTED, LEVEL 6
02	RETRY TIMEOUT	18	EXTERNAL VECTOR SELECTED, LEVEL 5
03	ILLEGAL CNF RECEIVED BY MASTER PORT IN DATA CYCLE	19	EXTERNAL VECTOR SELECTED, LEVEL 4
04	BAD PARITY RECEIVED	20	IDENT ARB LOST
05	NO ACK CNF RECEIVED FOR MASTER PORT COMMAND	21	ACK CNF RECEIVED FOR ERROR VECTOR
06	ILLEGAL CNF RECEIVED FOR MASTER PORT COMMAND	22	NO ACK OR ILLEGAL CNF RECEIVED FOR FORCE-BIT IPINTR/STOP COMMAND
07	READ DATA SUBSTITUTE OR RESERVED STATUS CODE RECEIVED	23	NO ACK OR ILLEGAL CNF RECEIVED FOR INTR COMMAND
08	ACK CNF RECEIVED FOR NON-ERROR VECTOR, LEVEL 4	24	ADVANCED RETRY CNF RECEIVED
09	ACK CNF RECEIVED FOR NON-ERROR VECTOR, LEVEL 5	25	INTERNAL REGISTER WRITTEN
10	ACK CNF RECEIVED FOR NON-ERROR VECTOR, LEVEL 6	26	RETRY CNF RECEIVED FOR MASTER PORT COMMAND
11	ACK CNF RECEIVED FOR NON-ERROR VECTOR, LEVEL 7	27	SELF-TEST PASSED
12	BUS BSY ERROR	28	BUS TIMEOUT
13	ILLEGAL CNF RECEIVED FOR SLAVE DATA	29	ACK RECEIVED FOR SLAVE READ DATA
14	BAD PARITY RECEIVED DURING SLAVE TRANSACTION	30	MASTER PORT TRANSACTION COMPLETE
15	STALL TIMEOUT ON SLAVE TRANSACTION	31	NO EVENT

Figure 15 • DC514 Event Status Mask Register Format

The CBIC Control and Status Register (CBIC CSR) controls and monitors miscellaneous CBIC operations. During II bus write operations to this register, only the information on data lines IID<31:26> is written. Parity is not checked because the CBIC control device does not access this register. Parity on lines IIP<3:0> is not significant. During II bus read operations from this register, bits 25:00 are read as zeros. The CBIC generates parity for the longword on IIP<3:0>. The register format is shown in Figure 16. Table 7 describes the bit functions.

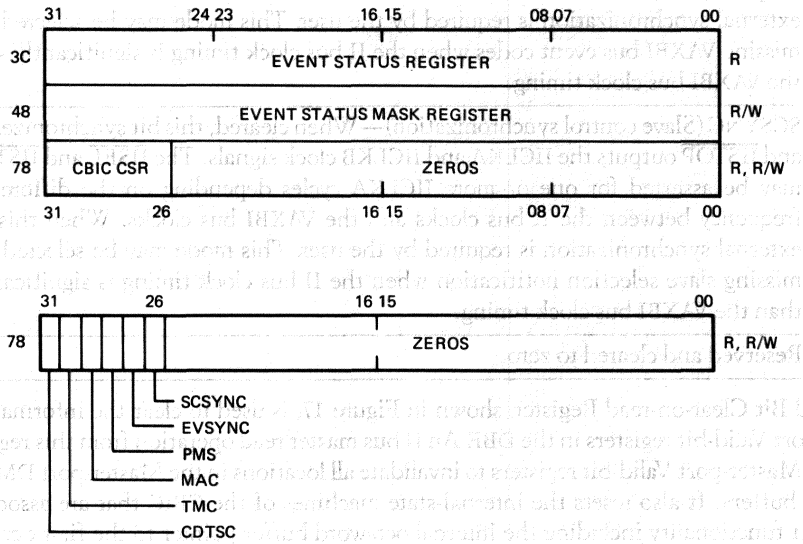


Figure 16 • DC514 Control and Status Register Format

Table 7 • DC514 Control and Status Register Description

Bit	Function
31	CDTSC (CBIC driver three-state control)—Not set during normal operation. This function is used during in-circuit and production testing. When this bit is set, the II bus drivers and the $\overline{\text{BIBROKE}}$ output become a high-impedance.
30	TMC (Turbo mode control)—Enables the VAXBI bus drivers to operate with 100 nanosecond bus cycles instead of the normal 200 nanosecond bus cycles.
29	MAC (Master abort control)—When set, a master abort condition is recognized by the BI interface. After recognition, this bit is cleared. Refer to the *VAXBI System Reference Manual* for detailed information.
28	PMS (Parity mode select)—Selects the source of the parity passed by the CBIC when moving data from the DBF to the VAXBI bus. When cleared, the user control parity mode is selected and parity errors from the II bus or the DBF are passed to the VAXBI bus. When set, internal parity mode is selected and the CBIC regenerates valid parity to be passed to the VAXBI bus. This bit does not affect operation of the bad parity detected bit in the ESR. Although valid parity is passed to the VAXBI bus in internal parity mode, the original parity errors are detected and recorded in the ESR when the bad parity detected bit is not masked by the ESMR.
27	EVSYNC (Event synchronization)—When cleared, this bit synchronizes the $\overline{\text{IIEV<4:0>}}$ outputs with the IICLKA and IICLKB clock signals. The $\overline{\text{IIEV<4:0>}}$ outputs may be asserted for one or more IICLKA cycles depending on the difference of the frequency between the II bus clocks and the VAXBI bus clocks. When this bit is set, external synchronization is required by the user. This mode may be selected to prevent missing VAXBI bus event codes when the II bus clock timing is significantly slower than the VAXBI bus clock timing.
26	SCSYNC (Slave control synchronization)—When cleared, this bit synchronizes the $\overline{\text{IISEL}}$ and $\overline{\text{IISTOP}}$ outputs the IICLKA and IICLKB clock signals. The $\overline{\text{IISEL}}$ and $\overline{\text{IISTOP}}$ signals may be asserted for one or more IICLKA cycles depending on the difference of the frequency between the II bus clocks and the VAXBI bus clocks. When this bit is set, external synchronization is required by the user. This mode may be selected to prevent missing slave selection notification when the II bus clock timing is significantly slower than the VAXBI bus clock timing.
25:00	Reserved and cleared to zero.

The Valid Bit Clear-on-read Register, shown in Figure 17, is used to clear the information in the Master-port Valid-bit registers in the DBF. An II bus master read operation from this register clears the eight Master-port Valid-bit registers to invalidate all locations in the Master-port DMA A and B octaword buffers. It also resets the internal-state machines of the CBIC that are associated with DMA port functionality including the internal octaword buffer pointer to the first octaword data buffer. After a block move operation has been terminated as a result of an error condition, this register is used to clear the valid bits before initiating another block move operation. It can also be used to clear the valid-bit registers following the successful completion of a block move transaction.

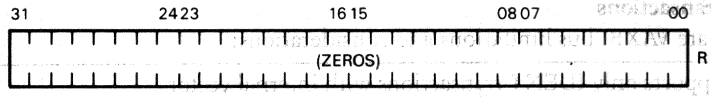


Figure 17 • DC514 Valid Bit Clear-on-read Register Format

During II bus read operations from this register, the CBIC generates correct parity for the entire longword on the IIP<3:0> lines.

• Functional Operation

This section provides functional information related to the operation of the CBIC. Refer to the *VAXBI Systems Reference Manual* for detailed information of the VAXBI bus and associated interfaces.

Master-port Functions

If a Retry confirmation code is received during a map or DMA read or write transaction from a VAXBI bus master, the CBIC will retry the transaction until it is successfully completed. If the retry counter in the CBIC times-out before the transaction has been completed, the CBIC will discontinue the retry attempt and will set the retry time-out (RTO) bit 20 in the Bus Error register. An error interrupt is then initiated if enabled and a RETRY time-out event code is generated if enabled.

If a master-port transaction is retried, no new II bus master-port transaction requests are honored until either the master-port transaction has been completed or a retry time-out occurs. If a DMA master-port transaction is retried, the data in the octaword buffer being used as the source for the transaction must not be changed.

A retry transaction will be aborted by a VAXBI bus Stop transaction that selects the retrying CBIC as a receiving slave or when the master abort bit 29 in the CBIC CSR is set.

Slave-port Function

During a VAXBI bus slave-port read transaction where the CBIC is the selected as a slave, the CBIC extends the transaction until the II bus master can read the slave-port address register in the DBF and write the correct information to the slave-port data and status registers. The slave-port data register should be the last register accessed.

The CBIC is limited to a maximum cycle extension of eight VAXBI bus stall cycles. The II bus slave must respond with the read data and must read the Slave-port Data register within 1.2 microseconds after the $\overline{\text{IISEL}}$ signal is asserted.

When two write transactions to slave-port registers are issued in proximity to each other, the data written by the first transaction may be overwritten by the data from the second transaction. To prevent this condition, the CBIC extends the data cycle of the first transaction by issuing stall cycles until the II bus slave can access these registers. The last register to be accessed by the II bus slave should be the slave-port data register.

VAXBI Bus Transactions

The following are VAXBI bus limitations and considerations:

- The CBIC supports only IDENT transactions with internal vectors.
- When a Stop transaction is received and the CBIC is a selected slave, the CBIC acknowledges but does not extend the transaction and performs the following actions:

1. Deasserts the $\overline{\text{BINOARB}}$ signal if a pending master state is aborted.
2. Sets the INIT bit 13 in the VAXBI Bus Control and Status Register (VAXBICSR)
3. Removes all current transaction requests at the II bus interface.
4. Ignores all subsequent VAXBI bus transaction requests at the II bus interface caused by asserting the $\overline{\text{IIRQ0}}$ line by the II bus master and sets the INIT bit 13 in the VAXBICSR. Subsequent loopback transaction requests presented at the II bus interface caused by the assertion of the $\overline{\text{IIRQ1}}$ by the II bus master are processed normally.
5. Resets the master and slave sequencers of the CBIC. As a result of this action, the master port interface that send a Stop transaction to its own slave-port interface will not receive a summary event code.
6. Clears all posted interrupt states. This clears the Sent and Force bits in the user's interface, the error interrupt control registers, the Retry state if it exists, the Retry counter, and the HEIE bit 7 and SEIE bit 6 in the VAXBICSR.

Clearing the STOPEN bit 13 in the BCICSR suppresses the generation of the $\overline{\text{IISEL}}$ and $\overline{\text{IISTOP}}$ outputs. The CBIC does not perform the initialization previously described.

Diagnostic Features

The CBIC contains diagnostic features to ensure reliable operation and to facilitate maintenance including self-test programs, parity generation, and a diagnostic mode.

Self-test and Initialization

The CBIC performs a self-test operation during the powerup sequence and as part of a node reset sequence. During either sequence the self-test begins after the $\overline{\text{IIDCLO}}$ input is deasserted. During the powerup self-test, the $\overline{\text{BINOARB}}$ signal is held asserted to prevent bus activity. During a node reset self-test, the $\overline{\text{BINOARB}}$ signal remains deasserted. In the cycle following successful completion of self-test, the CBIC transfers the self-test passed event code and sets the Self-Test Status (STS) bit 11 in the VAXBICSR. This bit is cleared during powerup.

The absence of a self-test passed event code indicates that the self-test has failed and the CBIC deasserts all VAXBI bus drivers by using a redundant driver disable signal. The duration of a successful self-test is approximately 4096 cycles (0.82 milliseconds). If the self-test is not completed in this time, a timer terminates the self-test after approximately 2.5 million cycles (500 milliseconds) and disables the VAXBI bus drivers.

The II bus master can determine the result of the self-test operation by waiting for the self-test passed event code to be received or by reading the STS bit 11 in the VAXBICSR. A loopback transaction is used to read the VAXBICSR because the STS bit is also used to enable the VAXBI bus drivers. If self-test fails, the STS bit is cleared, the VAXBI bus drivers remain disabled, and a VAXBI bus transaction from this node cannot be successfully completed. A loopback transaction that does not use the VAXBI bus data path can be performed provided that the self-test failure does not disable the loopback read transaction by the CBIC.

Parity Generation

The BIIC generates and checks odd parity. A parity bit is generated for each byte of data in the DBF including command, mask, status, or valid-bit field bytes. The parity bit remains associated with a data byte when the byte alignment changes for odd VAXBI bus addresses in the DMA master port. During II bus read operations, the parity bits associated with nonexistent bytes is supplied correctly. This includes the upper bytes during shifted read operation to non-wraparound registers and bytes of registers that are read as zero.

Parity is generated for every byte that is read from the DBF by the CBIC control logic as it services the BI interface during a VAXBI bus transaction. This parity is compared with the parity bit stored with that byte when it was written by the II bus master or slave to the DBF. If the parity is different and bit 00 in the ESMR is not set, the bad parity detected bit 31 in the ESR is set and the $\overline{\text{IIESTAT}}$ output is asserted.

The Parity Mode Select (PMS) bit 28 in the CBIC CSR determines which parity bit is passed when moving data from the DBF to the VAXBI interface. When set, internal parity mode is selected and the CBIC regenerates good parity. When cleared, user parity mode is selected, and parity errors from the II bus or from the DBF are passed to the VAXBI bus. Any write transactions to memory in progress when a parity error is detected will be aborted. The PMS bit should be set if parity is not implemented in the user's adapter. The CBIC would then generate good parity to be passed to the VAXBI bus regardless of the parity previously detected. The state of the PMS bit does not affect operation of the bad parity detected status bit 31 in the ESR. Previous parity errors are detected and recorded in the ESR when the bad parity detected bit is not masked by the ESMR.

For every longword written to the DBF by the CBIC when servicing the VAXBI interface, parity is generated and compared with the parity bit generated by the VAXBI interface for that longword. When the parity is different and bit 00 in the ESMR is not set, the bad parity detected bit (bit 31 in the ESR) is set and the $\overline{\text{IIESTAT}}$ signal is asserted.

Diagnostic Mode

The CBIC implements a subset of the diagnostic mode used in the BIIC. Refer to the *VAXBI System Reference Manual* for detailed BIIC information. This mode can be used to develop bus testers and other diagnostic equipment to facilitate the testing of the CBIC and provide more flexible access to the VAXBI bus.

The CBIC implements one of the two BIIC transparent modes. The II bus signals are reassigned for correspondence between the VAXBI bus signals and II bus signals as shown in Table 8.

Table 8 • DC514 VAXBI Bus and II Bus Signal Correspondence

II Bus Signal	State	VAXBI Bus Signal
IID<31:00>	inverted	$\overline{\text{BID<31:00>}}$
IIP<3:0>	inverted	$\overline{\text{BII<31:00>}}$
IIBM0	inverted	$\overline{\text{BIP}}$
$\overline{\text{IIEV0}}$	not inverted	$\overline{\text{BICNF0}}$
$\overline{\text{IIEV1}}$	not inverted	$\overline{\text{BICNF1}}$
$\overline{\text{IIEV2}}$	not inverted	$\overline{\text{BICNF2}}$
$\overline{\text{IIEV3}}$	not Inverted	$\overline{\text{BINOARB}}$
$\overline{\text{IIEV4}}$	not Inverted	$\overline{\text{BIBUSY}}$

In transparent mode, the user's interface transfers data on the IID, IIP, BM, and EV lines synchronously with the VAXBI bus clock signals. The CBIC asserts the data on the VAXBI bus. The diagnostic mode code must not be transferred on lines $\overline{IIRQ<1:0>}$ until the self-test has been completed. The diagnostic mode control signals $\overline{IIRQ<1:0>}$ may be transferred concurrently with the code when the transparent mode is selected for the CBIC.

Three-state Functions

The II bus drivers are three-state outputs to facilitate in-circuit and production tests. When bit 08 of the CBIC CSR is set, the II bus drivers, except the $\overline{IIACLO}$ and $\overline{IIDCLO}$ outputs, become a high impedance. This bit must not be set during normal operation.

Powerup Operation

During powerup operations, the CBIC asynchronously asserts the $\overline{IIDCLO}$ output after the the $\overline{BIDCLO}$ input is asserted and all VAXBI bus drivers are disabled. During the last cycle in which the $\overline{IIDCLO}$ output is asserted, the CBIC loads the device register with data from the IID<31:00> lines and loads the node ID field in the VAXBICSR with data from IIP<3:0>. The $\overline{IIDCLO}$ output is used to transfer this data. Internal pullup circuits will set the IID<31:00> lines to a high-impedance state at this time. This feature can minimize the number of signals to be driven by the user's interface during powerup operations. The output current characteristics of the pullup circuits should be verified to ensure that they are sufficient for the requirements.

The user's interface must transfer the node ID on lines IIP<3:0> while $\overline{IIDCLO}$ is asserted. If no other data is provided, the CBIC will load all ones into the device register which can then be loaded with data during node initialization by a normal write-type transaction.

The powerup sequence of user-designed nodes are required to conform to VAXBI bus architectural standards.

Transaction Timing Sequences

The transaction and control timing sequences of the CBIC are shown in the ac electrical characteristics.

Specifications

The mechanical, electrical, and environmental characteristics of the CBIC are described in the following paragraphs. The test conditions for the electrical values are as follows unless otherwise specified.

- Junction temperature (T_J): 0°C to 125°C
- Power supply voltage (V_{CC}): 4.75 V to 5.25 V

Mechanical Configuration

The physical dimensions of the DC514 133-pin Pin Grid Array (PGA) package are shown in the Appendix .

Absolute Maximum Ratings

Stresses greater than absolute maximum ratings may permanently damage the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

- Pin voltages: −1.5 V to 7.0 V
- Operating junction temperature (T_J): 0°C to 125°C
- Storage temperature (T_S): −55°C to 125°C
- Ambient temperature operating range (T_A): 0°C to 70°C
- Package dissipation: 2.5 W*

*Package dissipation is approximately 0.575 watts higher than the product of the maximum supply current and supply voltage because of the dissipation of the VAXBI bus drivers used to sink the external VAXBI bus pullup current.

dc Electrical Characteristics

Table 9 lists the dc electrical parameters for the input and output pins of the CBIC.

Table 9 • DC514 dc Input and Output Parameters

Symbol	Parameter	Requirements Min.	Requirements Max.	Unit	Test Conditions
II I_I	Input current	—	±20	µA	$0 < V_I < 5.25$ V $0 < V_{CC} < 5.25$ V
II I_{ID}	Input current	—	−0.25	mA	$V_I = 2.4$ V[1]
	$\overline{\text{IIADCLO}}$ asserted	−1.0	—	mA	$V_I = 0.5$ V[1]
II I_{OH}	High-level output current except $\overline{\text{IIADCLO}}$	−400	—	µA	V_{OUT} = II V_{OH}
	$\overline{\text{IIADCLO}}$ only	−5.4	—	mA	V_{OUT} = II V_{OH}
II I_{OL}	Low-level output current except $\overline{\text{IIADCLO}}$	4.0	—	mA	V_{OUT} = II V_{OL}
	$\overline{\text{IIADCLO}}$ only, power off	100	—	µA	V_{OUT} = II V_{OL} $V_{CC} = 0$ V
II V_{IL}	Low-level input voltage	−1.0	0.8	V	
II V_{IH}	High-level input voltage except $\overline{\text{BITIME}}$	2.0	—	V	
	$\overline{\text{BITIME}}$ only	2.4	—	V	
II V_{OH}	High-level output voltage	2.7	—	V	I_{OUT} = II I_{OH}
II V_{OL}	Low-level output voltage	—	0.5	V	I_{OUT} = II I_{OL}
II I_{OS}	Short-circuit output current	—	−150	mA	[2]

Symbol	Parameter	Requirements Min.	Max.	Unit	Test Conditions
II I_{ZO}	High-impedance leakage current	—	±20	µA	
II C_{IO}	Pin capacitance	—	10	pF	$0 < V_{IO} < V_{CC}$
BI I_I	Input current	-270	30	µA	$0 < V_{IO} < V_{CC}$
BI I_{OZ}	Leakage current	—	20	µA	$0 < V_{IO} < V_{CC}$
BI I_{OL}	Low-level output current	21	—	mA	$V_{OUT} = BI\ V_{OL}$
BI V_{OL}	Low-level output voltage	—	0.6	V	$I_{OUT} = BI\ I_{OL}$
BI V_{OH}	High-level output voltage	2.3	3.5	V	
BI V_{IH}	High-level input voltage	1.95	—	V	
BI V_{HHY}	High-level hysteresis voltage	1.45	—	V	[3]
BI V_{IL}	Low-level input voltage	-1.0	1.1	V	
BI V_{LHY}	Low-level hysteresis voltage	—	1.4	V	[3]
BI C_{IO}	Input/output pin capacitance	—	6.0	pF	$V_{IO} = 2.5$ V[4]
I_{CC}	Power supply current	—	300	mA	$V_{CC} = 5.25$ V

[1]While $\overline{\text{IIADCLO}}$ is asserted, IID<31:00> and IIP<3:0> are internally pulled up and can source a minimum of 250 µA at 2.4 V. The user's interface logic must sink a minimum of 1.0 mA at 0.5 V to drive these lines low while $\overline{\text{IIADCLO}}$ is asserted.

[2]Not more than one output should be short circuited at a time and the duration of the short should not exceed 1.0 second.

[3]For BI V_{HHY}, the CBIC does not detect a change in input state of the hysteresis voltage even if the input voltage drops to BI V_{HHY} following the application of BI V_{HHY}.
For BI V_{LHY}, the CBIC does not detect a change in input state even if the input voltage rises to BI V_{LHY} following the application of of BI V_{LHY}.

[4]The device under test must be poweredup during this test and $\overline{\text{BIDCLO}}$ should be asserted at all times, except when measuring C_{IO} for $\overline{\text{BIDCLO}}$.

ac Electrical Characteristics

The input and output signal timing sequences for the DC514 CBIC are shown in Figures 18 through 27. Table 10 lists the signal timing parameters.

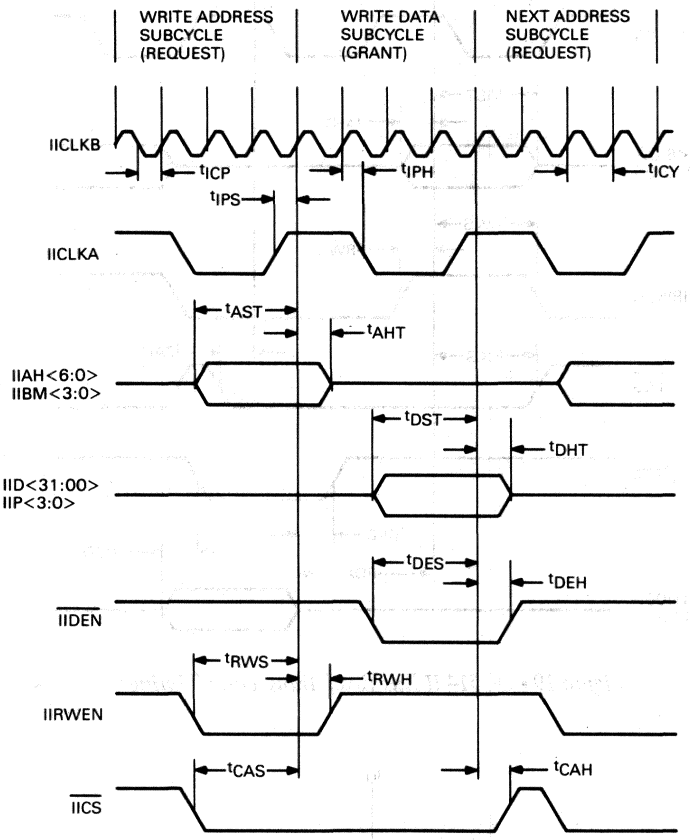


Figure 18 • DC514 II Bus Write Transaction Timing

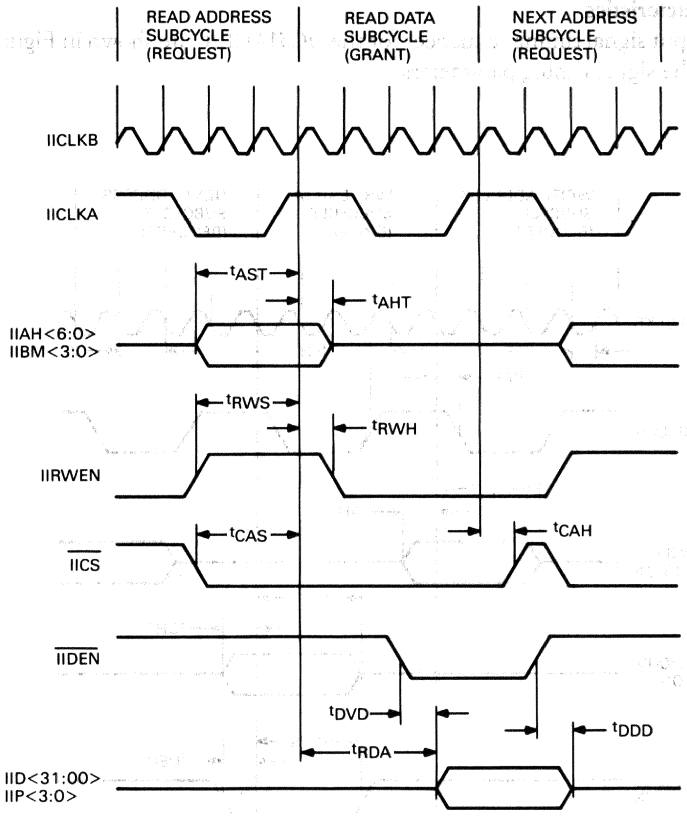


Figure 19 • DC514 II Bus Read Transaction Timing

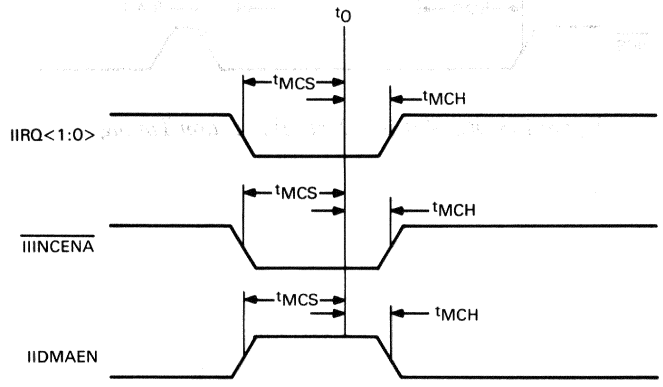


Figure 20 • DC514 Master-port Control Signal Timing

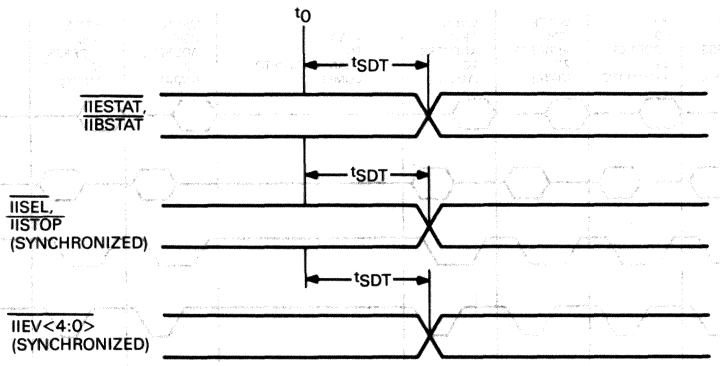


Figure 21 • DC514 Master- and Slave-port Status Signal Timing

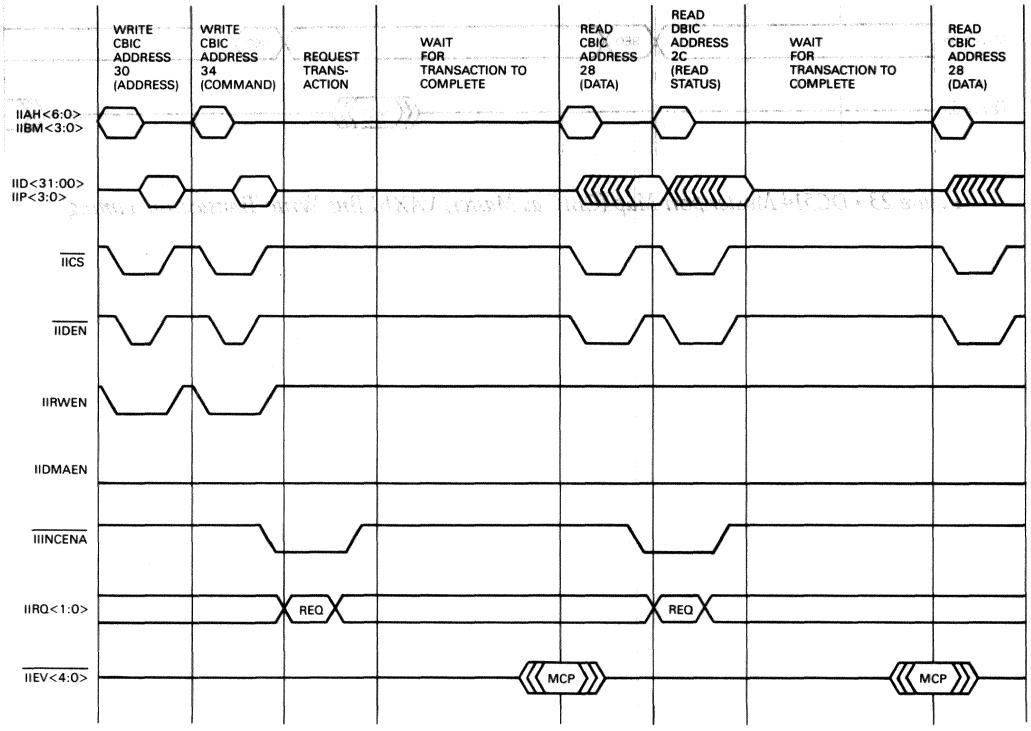


Figure 22 • DC514 Master-port Map (CBIC as Master) VAXBI Bus Read Read Transaction Timing

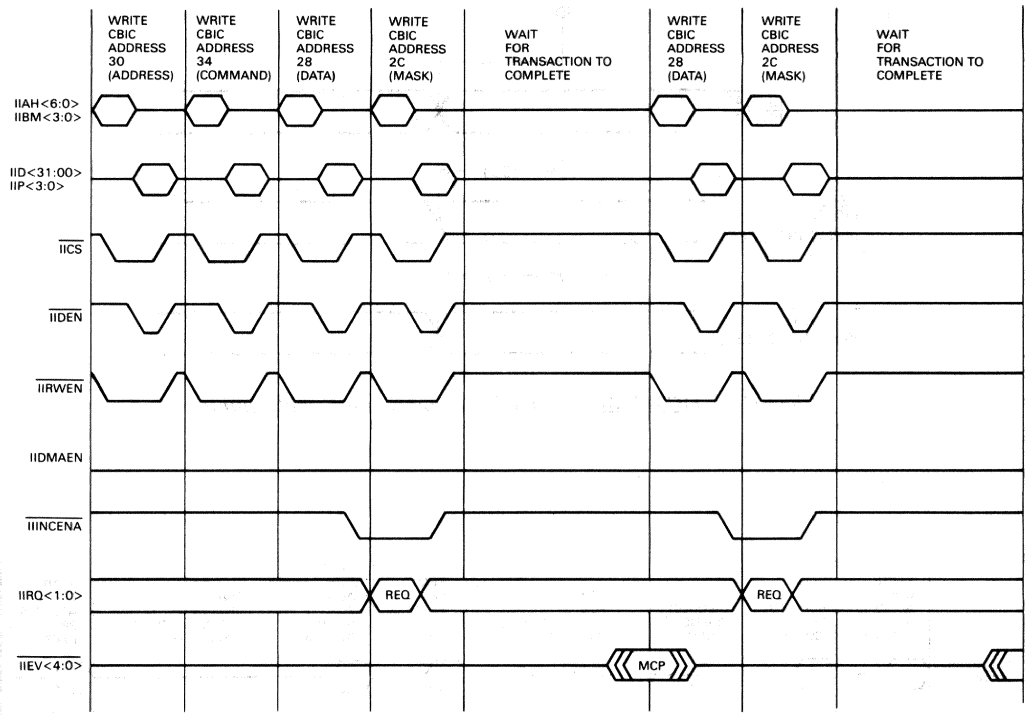


Figure 23 ▪ DC514 Master-port Map (CBIC as Master) VAXBI Bus Write Transaction Timing

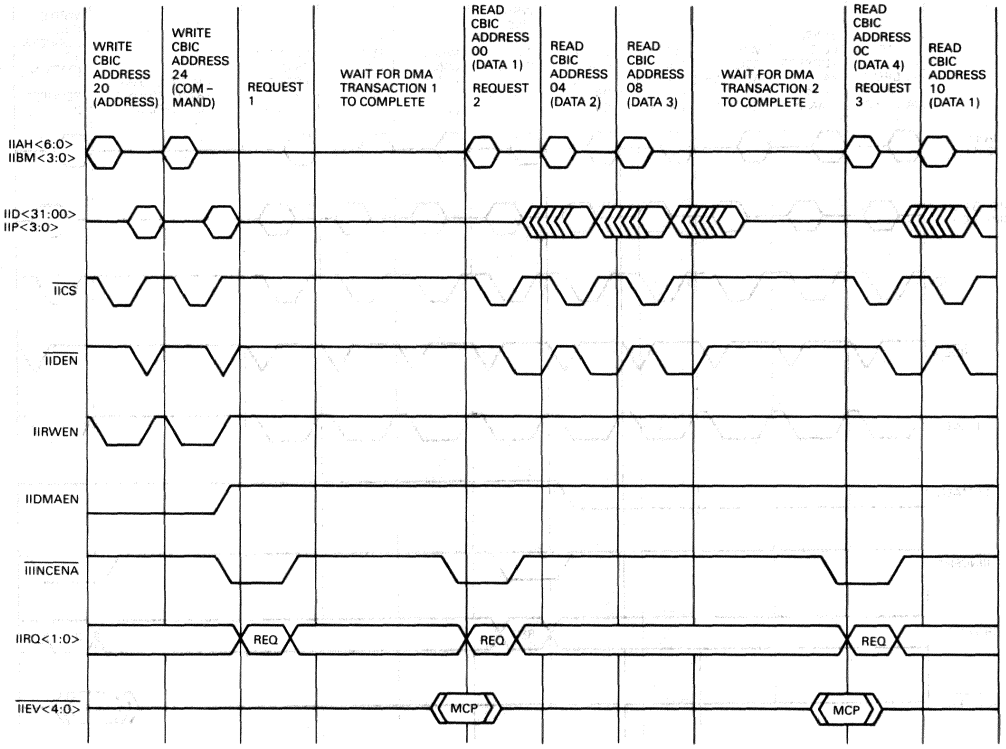


Figure 24 ▪ DC514 Master-port DMA (CBIC as Master) VAXBI Bus Read Transaction Timing

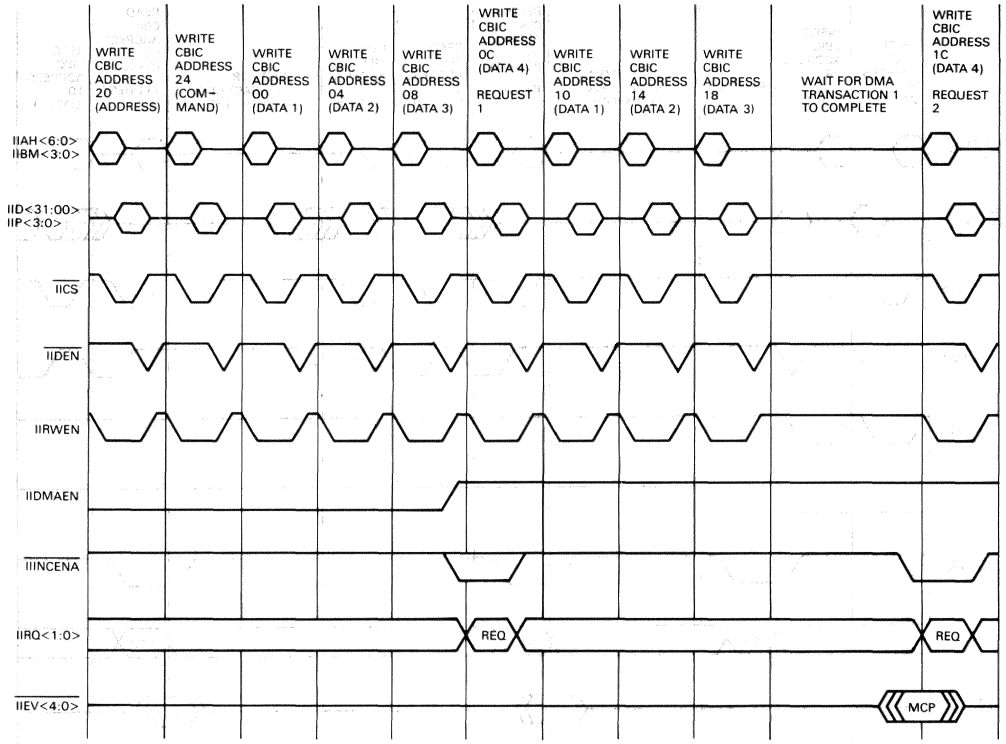


Figure 25 • DC514 Master-port DMA (CBIC as Master) VAXBI Bus Write Transaction Timing

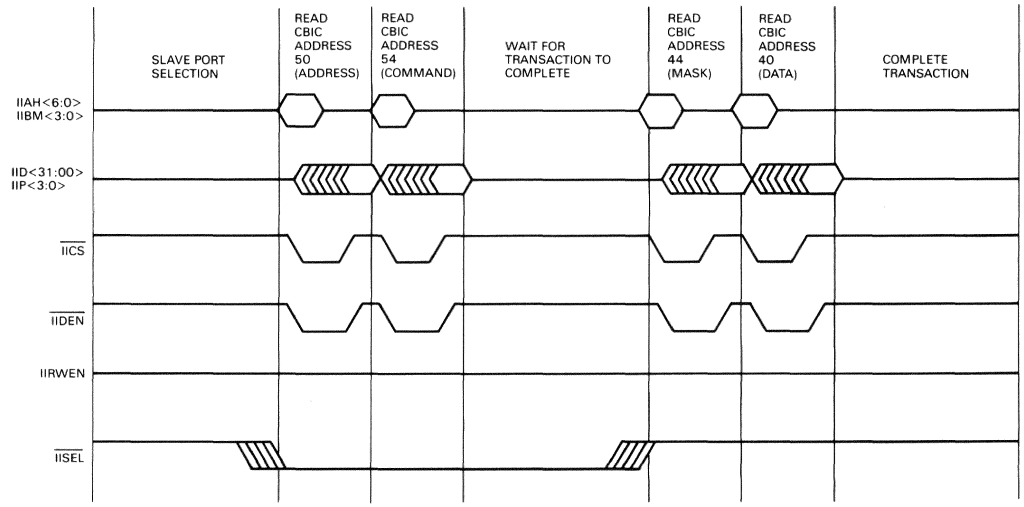


Figure 26 • DC514 Slave-port (CBIC as Slave) VAXBI Bus Read Transaction Timing

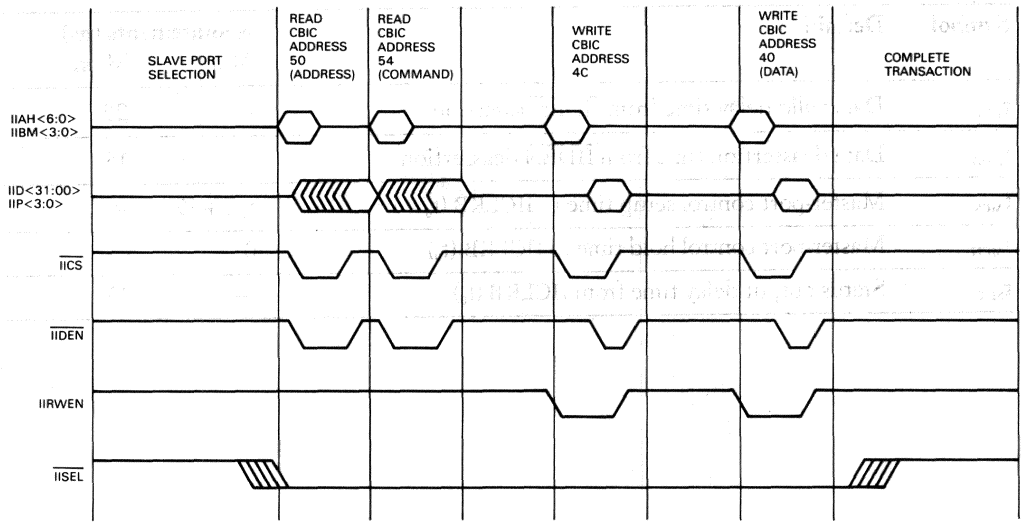


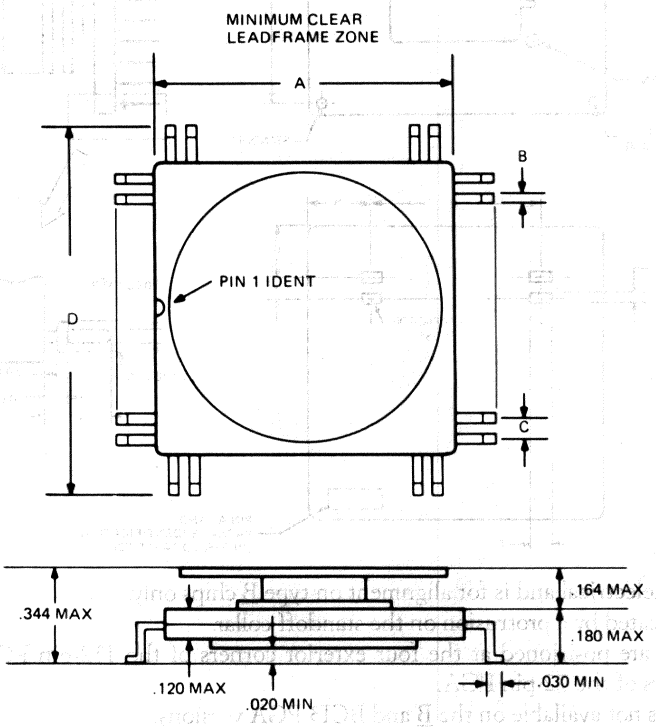
Figure 27 • DC514 Slave-port (CBIC as Slave) VAXBI Bus Write Transaction Timing

Table 10 • DC514 ac Timing Parameters

Symbol	Definition	Requirements (ns) Min	Max.
t_{ICY}	IICLKB clock period	24	DC
t_{ICP}	Clock pulse width high or low	8.0	—
t_{IPS}	IICLKB setup time to IICLKB (t_0)	5.0	—
t_{IPH}	IICLKA hold time from IICLKB (t_0)	5.0	—
t_{AST}	II bus address setup time to IICLKB (t_0)	0	—
t_{AHT}	II bus address hold time to IICLKB (t_0)	10	—
t_{DST}	II bus data setup time to IICLKB (t_0)	t_{ICY}	—
t_{DHT}	II bus data hold time to IICLKB (t_0)	0	—
t_{RWS}	IIRWEN setup time to IICLKB (t_0)	15	—
t_{RWH}	IIRWEN hold time to IICLKB (t_0)	10	—
t_{CAS}	$\overline{\text{IICS}}$ setup time from IICLKB (t_0)	15	—
t_{CAH}	$\overline{\text{IICS}}$ hold time from IICLKB (t_0)	10	—
t_{DES}	$\overline{\text{IIDEN}}$ setup time to IICLKB (t_0)	t_{ICY}	—
t_{DEH}	$\overline{\text{IIDEN}}$ hold time to IICLKB (t_0)	0	—
t_{RDA}	Read data access time from IICLKB (t_0)	—	$t_{ICP}+55$

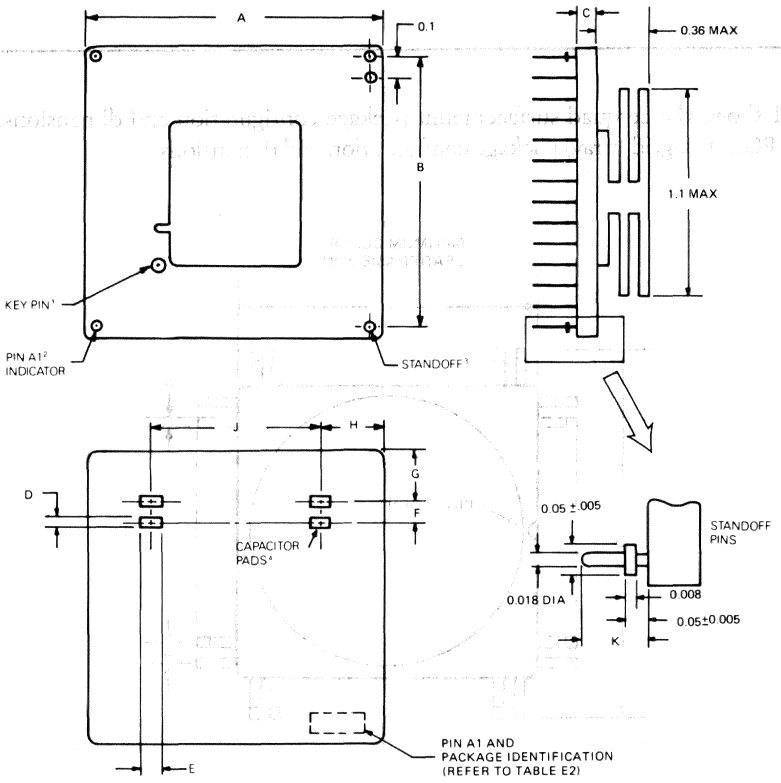
Symbol	Definition	Requirements (ns) Min	Max.
t_{DVD}	Data valid delay time from $\overline{IIDEN}$ assertion	—	25
t_{DDD}	Data deassertion time from $\overline{IIDEN}$ deassertion	—	15
t_{MCS}	Master-port control setup time to IICLKB (t_0)	$t_{ICY} + 15$	—
t_{MCH}	Master-port control hold time to IICLKB (t_0)	0	—
t_{SDT}	Status output delay time from IICLKB (t_0)	—	15

Figure A.1 shows the cerquad surfacemount package configuration and dimensions. Figure A.2 shows the PGA (pin-grid-array) package configuration and dimensions.



Number of Leads	Dimensions A	B	C	D
44	0.6	0.02	0.05	0.825
68	0.9	0.02	0.05	1.125
84	1.1	0.02	0.05	1.325
132	0.9	0.012	0.025	1.125
164	1.1	0.012	0.025	1.325

Figure A.1 ▪ Cerquad Surfacemount Package Configuration and Dimensions



[1]Key pin is nonelectrical and is for alignment on type B chips only.
[2]Pin A1 is indicated by a protrusion on the standoff collar.
[3]Standoff pins are positioned at the four exterior corners of the 132-pin PGA and at the four interior corners of the 72-pin PGA.
[4]Capacitor pads not available on the B and BCI3 PGA versions.

Type*	Pins	Dimensions									
		A	B	C	D	E	F	G	H	J	K
	72	1.17	1.0	0.1	0.05	0.16	0.1	0.36	0.145	0.88	0.17
B	132	1.4	1.3	0.1	N/A	N/A	N/A	N/A	N/A	N/A	0.18
M, IE, F	132	1.4	1.3	0.12	0.05	0.12	0.12	0.35	0.33	0.74	0.18
BCI3	132	1.4	1.3	0.12	N/A	N/A	N/A	N/A	N/A	N/A	0.18

*Package Identification:

Type B = VAXBI bus BCAI and BIIC chips
M = V-11 M chip
I/E = V-11 I/E chip
F = V-11 F chip
BCI3 = VAXBI bus BCI3 chip

Figure A.2 • PGA Package Configuration and Dimensions