Data Sheet
March 1993

AT&T Microelectronics

AT&T DSP3210 Digital Signal Processor
The Multimedia Solution

Features and Benefits

Feature	Benefit
Microprocessor bus compatibility: ■ 32-bit, byte-addressable address space ■ Retry, relinquish/retry, and bus error support ■ Page mode DRAM support ■ Direct support for both 680x0 and 80x86 signaling	Designed for efficient bus master designs allowing the DSP3210 to easily be incorporated into µP-based systems. The 32-bit, byte-addressable space enables the DSP3210 and a µP to share common address space and pointer values as well.
AT&T *VCOS*™ operating system: ■ Real-time, multitasking operating system ■ Uses host rather than local memory ■ True parallel processing ■ Complete task management	Open development environment: ■ Dramatically lower system costs ■ Full utilization of both µP and DSP3210 ■ Simplifies both algorithm and application development
Full 32-bit floating-point arithmetic C-like assembly language Single-cycle PC relative addressing	Ease of programming/higher performance.
All instructions are single-cycle (four memory accesses per instruction cycle)	Higher performance.
Access to DSP32C programs	Access to the largest existing 32-bit DSP SW base.
*Logic Automation** model	Faster, more efficient system development.

Introduction

The DSP3210 brings the power of floating-point signal processing to personal computers and workstations, opening a wide range of multimedia applications. The DSP3210 has been engineered with a single focus: to enable advanced multimedia applications in personal computers and workstations. Based on AT&T's DSP32C architecture, the DSP3210 has the unique ability to be integrated into personal computer and workstation system designs. Particular attention is paid to primary bus interfacing; the DSP3210 is compatible with both 80x86 and 680x0 microprocessor signaling. This allows designers to easily create low-cost systems by using the DSP3210 as a bus-master device. A full, bus-level *SmartModel** of the DSP3210 is offered by Logic Automation, Inc. for system simulation of designs incorporating the DSP3210.

Along with its optimizing C-compiler and assembly-language software tools, the DSP3210 is further supported with AT&T's *VCOS* operating system.

The *VCOS* operating system provides a powerful real-time, multitasking and multiprocessing environment which effectively manages multimedia applications across various computer platforms. By employing innovative task- and code-management techniques, *VCOS* operating system allows the DSP3210 to use existing system memory in PCs and workstations rather than expensive dedicated SRAM for DSP program and data storage.

Complete real-time debugging tools are included to speed both application and algorithm development. By separating the application and algorithm development phases of multimedia software creation, the *VCOS* operating system greatly simplifies and shortens development schedules.

* *Logic Automation* and *SmartModel* are registered trademarks of Logic Automation, Inc.

Table of Contents

Contents

Page

Introduction (continued)

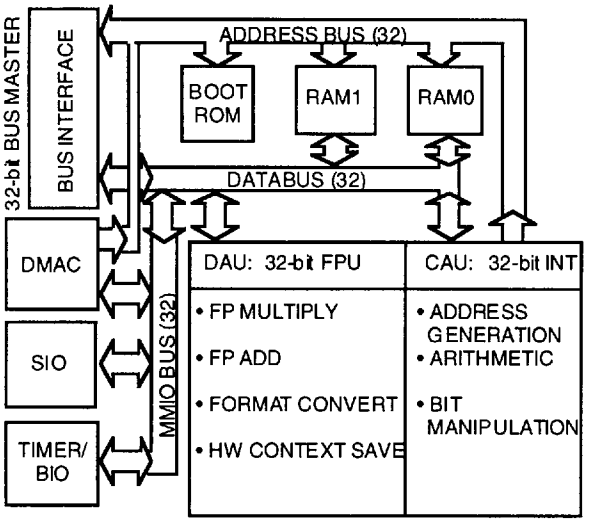


Figure 1. DSP3210 Block Diagram

The *VCOS* operating system includes its own multimedia library, complete with speech processing, speech recognition, graphics, music processing, and modem modules. The *VCOS* system is an open environment, so application developers may access third-party modules as well as AT&T's library.

Figure 1 shows the DSP3210 block diagram. In addition to a brief description of the DSP3210's application in multimedia environments, each functional unit, the instruction set, and the processor control features are described.

This data sheet is designed to be a companion document to the *AT&T DSP3210 Information Manual* (MN91-006OMOS). The brief descriptions of the DSP3210's architecture and instruction set are greatly expanded in the information manual. The data sheet is intended to give the latest, up-to-date, timing specifications and boot ROM firmware descriptions, while the information manual contains the detailed information needed to understand the DSP3210's operation.

Applications

The DSP3210 may be used in a variety of applications due to its raw floating-point power, low cost, low power dissipation, and interfacing flexibility. However, multimedia was the primary application considered when designing the DSP3210.

PC/Workstation Multimedia Applications

DSP3210 Motherboard Implementations

The DSP3210 is intended to be used in PC and workstation system architectures in which the DSP3210 is a parallel processor to a host processor. The DSP3210 maintains a 32-bit bus-master interface to system memory (see Figure 2).

The primary benefit of this system architecture is the DSP's ability to access program and data from system memory without host intervention. Furthermore, expensive local SRAM is replaced by the computer's existing system memory.

DSP3210 Add-In Cards

Existing computers with EISA, ISA, MCA, *NuBus**, SBus, and proprietary bus or CPU direct-slot capabilities can be easily retrofitted with low-cost DSP3210 boards to perform identical applications to motherboard-equipped PCs. In systems with direct CPU slots or 32-bit buses capable of bus mastering, these DSP3210-based cards require no DSP memory (memory is accessed via the bus or CPU direct slot).

Using the visible caching technique native to the *VCOS* operating system, boards installed in lower-bandwidth buses use inexpensive local 32-bit DRAM on the DSP3210 add-in card. In these environments, the DSP3210 uses the bus primarily to transfer tasks and I/O to and from the DSP3210 card.

* *NuBus* is a registered trademark of Massachusetts Institute of Technology.

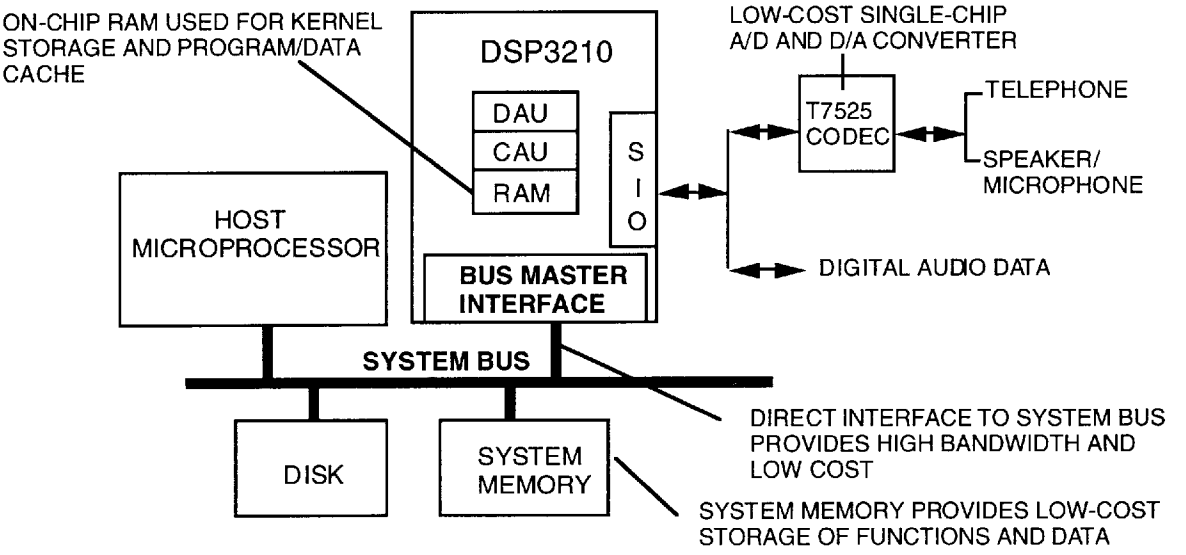


Figure 2. Typical DSP3210 PC/Workstation Motherboard Configuration

Applications (continued)

System Integration Under the *VCOS* Operating System

Since the DSP3210 supports both big- and little-endian byte ordering, sharing both data and pointer values with any host microprocessor is easily accomplished. This is especially useful in multimedia applications where intimate communications between the host microprocessor and DSP are necessary. For real-time signal processing under the *VCOS* operating system, on-chip SRAM is loaded with code and data from system memory before executing. Applications are broken into functions that are executed successively in this fashion. Nonreal-time background jobs may be either executed from system memory or cached into the DSP3210's internal memory.

Functional Description

Functional Units

The DSP3210 consists of seven functional units: control arithmetic unit (CAU), data arithmetic unit (DAU), on-chip memory (RAM0, RAM1, boot ROM), bus interface, serial I/O (SIO), DMA controller (DMAC), and timer/bit I/O (BIO) unit.

Control Arithmetic Unit (CAU)

The CAU is responsible for performing address calculations, branching control, and 16- or 32-bit integer arithmetic and logic operations. It is a RISC core consisting of a 32-bit arithmetic logic unit (ALU) that performs integer arithmetic and logical operations, a full 32-bit barrel shifter for efficient bit manipulation operations, a 32-bit program counter (PC), and twenty-two 32-bit general-purpose registers.

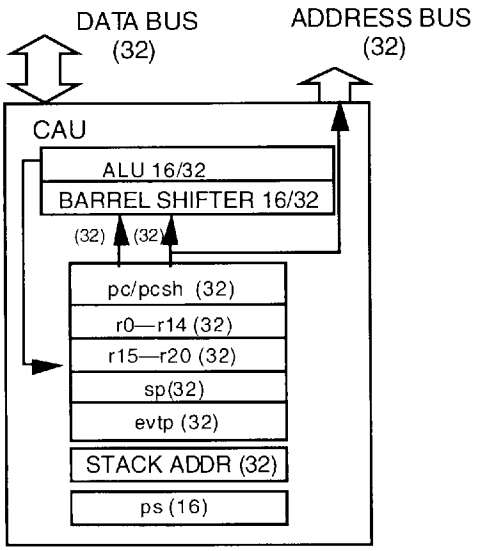


Figure 3. Control Arithmetic Unit (CAU)

The CAU performs two types of tasks: executing integer, data move, and control instructions (CA instructions), or generating addresses for the operands of floating-point instructions (DA instructions). CA instructions perform load/store, branching control, and 16- and 32-bit integer arithmetic and logical operations. DA instructions can have up to four memory accesses per instruction, and the CAU is responsible for generating these addresses using the postmodified, register-indirect addressing mode (one address is generated in each of the four states of an instruction cycle).

Data Arithmetic Unit (DAU)

The DAU is the primary execution unit for signal processing algorithms. This unit contains a 32-bit floating-point multiplier, a 40-bit floating-point adder, four 40-bit accumulators, and a control register (dauc). The multiplier and adder work in parallel to perform 16.7 million computations per second, yielding 33 MFLOP performance. The multiplier and adder each produce one floating-point result per instruction cycle. The DAU contains a four-stage pipeline (fetch, multiply, accumulate, write). Thus, in any instruction cycle, the DAU may be processing four different instructions, each in a different stage of execution.

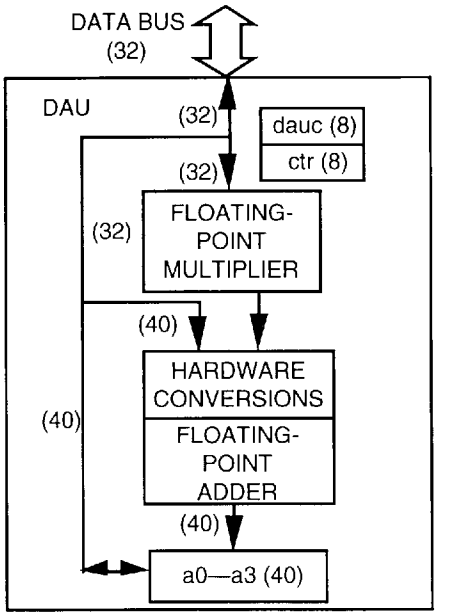


Figure 4. Data Arithmetic Unit (DAU)

The DAU supports two floating-point formats: single precision (32-bit) and extended single precision (40-bit). Extended single precision provides eight additional mantissa guard bits. Postnormalization logic transparently shifts binary points and adjusts exponents to prevent inaccurate rounding of bits when the floating-point numbers are added or multiplied. This eliminates concerns such as scaling and quantization error.

Functional Description (continued)

All normalization is done automatically, so the result in the accumulator is always fully normalized.

Single-instruction, data-type conversions are done in hardware in the DAU, thus reducing overhead required to do these conversions. The DAU performs data-type conversions between the DSP32 32-bit floating-point format and IEEE P754 standard 32-bit floating-point, 16- and 32-bit integer, 8-bit unsigned, μ-law, and A-law formats. The DAU also provides an instruction to convert a 32-bit floating-point operand to a 3-bit seed value used for reciprocal approximation in division operations.

On-Chip Memory

The DSP3210 provides on-chip memory for instructions and data. Instructions and data can arbitrarily reside in any location in both on- and off-chip memory. The DSP3210 provides two 1K x 32 RAMs and a 256 x 32 boot ROM. The boot ROM is preprogrammed to enable the DSP3210 to boot directly from external memory, such as slow, inexpensive ROM or EPROM.

Bus Interface

The external address bus of the DSP3210 is 32-bits wide and fully byte-addressable, allowing the DSP3210 to directly address 4 Gbytes of memory or memory-mapped hardware. External memory is partitioned into two logical address spaces, A and B. Each partition contains 2 Gbytes of address space.

The number of wait-states for external memory partitions A and B are independently configurable via the pcw register. Configured waits of 0, 1, 2, or 3 or more wait-states are programmable; this simplifies the interface to fast external memory. Unlike most digital signal processors (which employ full-cycle wait-states), the DSP3210 offers much greater flexibility. This flexibility is achieved by offering 1/4 cycle wait-states. Each wait-state is 1/4 of an instruction cycle, allowing greater granularity in determining optimal speed/cost memory trade-offs. When waits are externally controlled, the DSP adds wait-states until the memory acknowledges the transaction via the SRDYN pin.

The bus interface supports retry, relinquish/retry, and bus error exception handling. All signaling provided to the external system is configurable on reset to simplify the interface to a variety of microprocessor system buses.

Sharing of the external memory interface is performed via a complete request/acknowledge protocol. System throughput is greatly enhanced by the DSP3210's ability to execute from internal memory while the DSP3210 does not have ownership of the bus.

The DSP3210 will continue to execute from internal memory until accesses to the external memory are needed. At that point, the DSP asserts the BRN signal and waits for BGN. The bus arbiter acknowledges the bus request by asserting the DSP3210's bus grant pin (BGN). The DSP3210, in turn, acknowledges the grant by asserting the bus grant acknowledge (BGACKN) and driving the external memory interface pins. When the BGN is negated, any ongoing external memory transaction is completed before the DSP relinquishes the bus by placing the external memory interface bus in the high-impedance state and negating BGACKN.

Serial I/O (SIO)

The SIO unit provides serial communications and synchronization with external devices. The SIO signals support a direct interface to a time-division multiplexed (TDM) line, a zero-chip interface to codecs, and direct DSP-to-DSP transfers for multiprocessor applications. The SIO performs serial-to-parallel conversion of input data, and parallel-to-serial conversion of output data at a maximum rate of 25 Mbits/s. It is composed of a serial input port, a serial output port, and on-chip clock generators. Both ports are double buffered so that back-to-back transfers are possible. The SIO is configurable via the ioc register. The input buffer (IBUF), the output buffer (OBUF), and the ioc register are accessible as memory-mapped input/output (MMIO) registers in the instruction set. The ibuf and obuf are also accessible as I/O registers in the instruction set.

The data sizes of the serial input and output can be configured independently. Input data lengths of 8-, 16, and 32-bits and output data lengths of 8-, 16-, 24-, and 32-bits can be selected. The input and output data can be independently selected to be most significant bit first or least significant bit first.

SIO transfers can be made under program, interrupt, or DMA control. A program can test the input or output buffer status flags using conditional branch instructions. By configuring the exception mask register (emr), interrupt requests may be generated by the input and output buffer status flags. In DMA mode, transfers occur between IBUF, OBUF, and memory without program intervention.

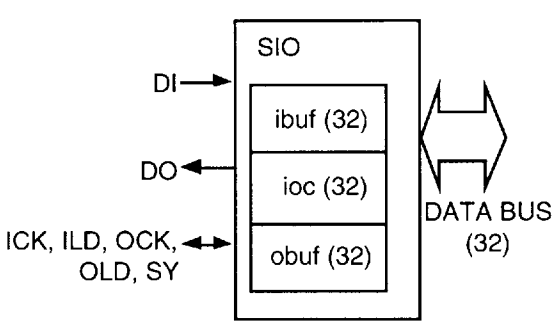


Figure 5. Serial IO (SIO)

Functional Description (continued)

DMA Controller (DMAC)

The DMA controller contains two DMA channels that are used in conjunction with the serial I/O: one for input DMA and one for output DMA. By configuring the input DMA channel, data being shifted into the serial input port can be buffered in memory without processor intervention. By configuring the output DMA channel, a buffer of data in memory can be supplied to the serial output, as necessary, without processor intervention.

The registers used to configure the DMA controller are directly accessible in the DSP3210's instruction set. By configuring the exception mask register (emr), interrupt requests can be generated when the memory buffer has been filled or emptied based on the size of the buffer requested.

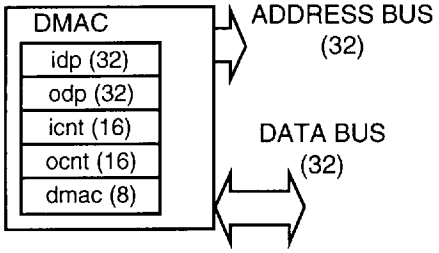


Figure 6. DMA Controller (DMAC)

Timer/Bit I/O

The timer is a programmable 32-bit interval timer/counter used for interval timing, rate generation, event counting, or waveform generation. The input to the timer can be derived from the DSP3210 clock, or it may come from an external source. The output of the timer can generate a maskable interrupt, or it may be selected as an output of the chip to drive external hardware. It can be configured to count down to zero once or to count continuously by automatically reloading the counter with its initial value when it reaches zero. The count value may be read or changed at any time during operation. The registers associated with the timer are accessible as MMIO registers in the instruction set. By configuring emr, interrupt requests may be generated when the count reaches zero.

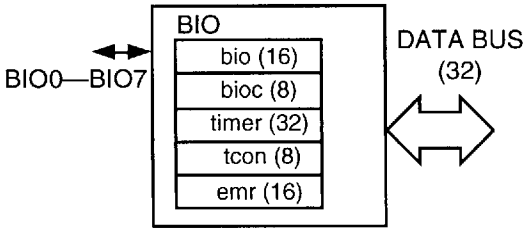


Figure 7. Timer/Bit I/O (BIO)

The BIO is a general-purpose 8-bit input/output port. It includes features that make it suitable for board-level status signal generation and control signal testing by the DSP3210. The BIO interface consists of eight I/O lines, any of which can be independently configured as an input or an output. Outputs can be written with either 1 or 0, toggled, or left unchanged. Inputs can be directly read and loaded into a CAU register and then tested, or the inputs can be read and directly written to memory. The registers associated with the BIO are accessible as registers in the DSP3210 instruction set.

DSP3210 Instruction Set

The assembly language of the DSP3210 frees programmers from tedious memorization of assembly language mnemonics. DSP3210 instructions are patterned after the C programming language and are entered in a natural equation syntax. In addition to being easier to learn, the resulting code is far more readable than mnemonic-based assembly languages, making code maintenance much easier.

C-like assembly language → Easy to learn/excellent readability

Below is an example assembly language instruction:

```
*r1++ = a0 = a1 + *r2++ * *r3++
```

The execution of this instruction simply follows the conventions of the high-level C programming language:

"Multiply the two floating-point values stored in the memory locations pointed to by registers r2 and r3. Add the result to the contents of accumulator a1, store the result in accumulator a0, and write the result to the 32-bit memory location pointed to by register r1. Postincrement pointer registers r1, r2, and r3."

Processor Control Features

The DSP3210 supports advanced control features that simplify system design and improve software performance. This section describes serial I/O direct-memory access (DMA), error and interrupt exceptions, and the powerdown mode.

Serial I/O DMA

External devices can access the on-chip RAM in the DSP3210, as well as external memory, using direct-memory access (DMA). DMA transfers occur between either internal or external memory and the serial I/O ports without processor intervention.

Functional Description (continued)

Exception Processing

Normal instruction processing can be altered by the introduction of interrupt routines or error handling routines. Exception processing is the set of activities performed by the processor in preparing to execute a handler routine or in returning to the program that took the exception. Error exception and interrupt exceptions cause different activities to be performed. In particular, error exceptions abort the current instruction and cannot resume processing. Interrupt exceptions shadow the current state of the processor before taking the interrupt exception; therefore, when the interrupt routine is complete, the program can be reinstated and continued.

The error and interrupt exceptions are prioritized, and some error sources, in addition to all interrupt sources, are individually maskable via the emr. A relocatable vector table controls program flow based on the source of the interrupt exception. In response to a given exception, the DSP branches to the corresponding address in the exception vector table which contains pairs of 32-bit words.

The DSP3210 provides a zero-overhead context save of the entire DAU (floating-point unit) for interrupt processing. Interrupt latency is only three instruction cycles for interrupt entry and one instruction for interrupt exit. Before servicing the interrupt, the DSP3210 automatically saves the state of the machine that is invisible to the programmer, as well as the DAU accumulators a0—a3 (including guard bits), all DAU flag status information, and the dauc register. Internal states that are visible to the programmer are saved and restored by the interrupt service routine. To return to the interrupted program, the interrupt service routine restores the user-visible state of the DSP3210 (that was saved) and then executes the ireturn instruction. This single-cycle interrupt return automatically restores the entire DAU state saved during interrupt entry. Quick interrupt entry/exit/context save is critical to real-time multimedia tasks.

Low-Power, Powerdown Mode

To address the needs of portable computer platforms, the DSP3210 is equipped with a powerdown mode that lowers power consumption to below 300 mW (from the typical power dissipation of 750 mW). This mode is implemented with a wait-for-interrupt instruction that stops internal execution in the DAU and CAU sections of the DSP3210. External memory and internal memory operations execute to completion and then wait. To ensure maximum system functionality, the DSP3210 bus arbitration logic remains active in this mode, and all peripheral units (serial I/O, DMA controller, timer, and BIO) remain active to perform I/O events. The interrupt handler is also active to sense interrupt requests. The DSP3210 exits the powerdown mode when either an unmasked interrupt is requested or an error exception occurs.

Boot ROM Code

F12 Silicon Revision

The boot ROM code in F12 silicon revision DSP3210 devices offers the user three different processor start-up options plus a self-test routine. The processor self-test routine performs a limited functional test of both on-chip arithmetic execution units (DAU and CAU). This routine is user-callable as described in the Detailed Description of Self-test Routine section.

Additionally, the 2048-word count limitation in the F11 EPROM boot routine has been removed. See the EPROM boot section . The F12 boot ROM code is upward compatible with F11 boot ROM code; subsequently, changes in design software or hardware are not needed when migrating to the F12 silicon revision.

Start-Up Options

Since the DSP3210 is intended for a variety of environments, three different start-up options are provided for maximum flexibility. Although the DSP3210 contains an on-chip boot ROM, the first and simplest start-up option doesn't use this ROM. This start-up option works as follows:

- The DSP3210 begins execution at external memory location 0 immediately following reset.

The other two boot options are controlled by the on-chip boot ROM of the DSP3210. These two routines work as follows:

- The DSP3210 loads a 32-bit address from external location 0 and begins executing code at the location indicated by the 32-bit address. This technique, useful in systems where the DSP3210 shares memory with another processor, is known as starting address redirection (SAR).
- The DSP3210 loads and executes instructions from off-chip, byte-wide memory. This technique is frequently used in embedded systems and is known as the EPROM boot routine.

The three start-up options are controlled by bits 10 and 13 of the **pcw** (processor control word) register. In turn, these **pcw** bits are initialized based on the powerup state of bits 7 and 4, respectively, of the DSP3210's bit I/O (BIO) port. Thus, the selection of the start-up routine is based on the powerup status of two signal pins. Details of the three options are given in Table 1, followed by a detailed description of each option. Also, the memory map for DSP3210 is shown in Figure 8

Functional Description (continued)

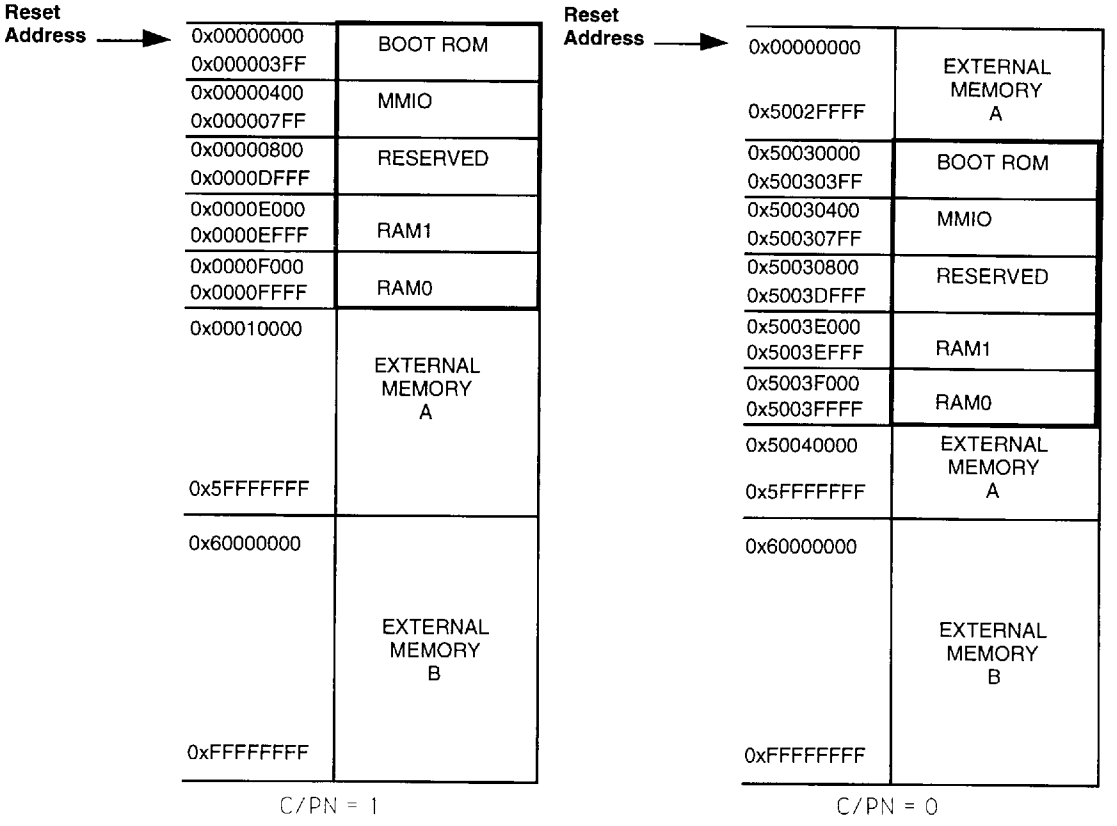


Figure 8. Memory Map

Table 1. Operation of Boot ROM Routines

Option	pcw[10]	pcw[13]	Operations
1	0	X (don't care)	Begin execution at external memory location 0 (external memory A) following reset.
2 (SAR)	1	0	1. Enable interrupt 1. 2. Enter wait-for-interrupt mode. 3. When interrupt 1 is acknowledged, change memory map to processor mode. 4. Read location 0 (ext memory) for branch vector, and write back a zero to location 0. 5. Disable interrupt 1. 6. Prior to return from interrupt, load **pcsh** with branch vector so that the DSP3210 starts executing from the branch vector. 7. Return from interrupt (begin execution at branch vector).
3 (EPROM)	1	1*	1. Read 4 bytes, organized as the least significant byte of the first four word addresses, from external memory starting at 0x60000000 into a 32-bit pointer (this value is known as Address). 2. Read the next 4 bytes, organized as the least significant byte of the next four word addresses, from external memory starting at 0x60000010 into a 32-bit integer (define this as the Count). 3. If Count is zero and Address is nonzero, branch to Address. 4. If Count is nonzero, read Count number of bytes from external memory starting at 0x60000020 and write them to Address to form words and then branch to Address.

* When **pcw**[13] is initialized to 1 via the bio[4] pin, future modification of the **pcw** is **not** disabled. Modification of the **pcw** is disabled only after **pcw**[13] is written with a 1 by a DSP3210 instruction.

Functional Description (continued)

Detailed Description of Start-up Boot Routines

Following reset, program execution always begins with the instruction at physical address 0. Physical address 0 will be either in boot ROM or external memory, depending on the state of pcw bit 10 (the C/PN bit). This bit selects one of two possible address maps, a computer mode address map or a processor mode address map, thus determining the address of the boot ROM. If C/PN is initialized to a 1 (computer mode), the boot ROM is mapped to begin at physical location 0, and the execution begins from the first location in the boot ROM. If C/PN is initialized to a 0 (processor mode), external memory A is mapped to physical location 0, execution begins there, and the boot ROM is ignored.

Processor Mode Boot

When C/PN = 0, the boot ROM is mapped to address 50030000. The boot ROM code is not executed since execution begins at address 0, which now lies within external memory A. Therefore, the DSP3210 executes code starting from location 0 of its off-chip memory.

Starting Address Redirection (SAR) Boot

When C/PN = 1, execution begins from the boot ROM, which is mapped to address 0. The boot routine tests **pcw** bit 13. If bit 13 = 0, the starting address redirection routine is selected. This is typically used in shared memory systems where a host microprocessor loads the DSP3210 instructions into a shared address space before the DSP3210 begins execution. The microprocessor then passes the starting address to the DSP3210 and signals the DSP3210 to begin execution.

The starting address redirection (SAR) routine begins in computer mode (with the boot ROM beginning at location 0) and ends with the DSP3210 configured in processor mode. The dynamic switching from computer mode to processor mode allows the DSP3210 to execute a start-up routine from boot ROM address 0 and still communicate with a host processor through memory location 0. At the beginning of the SAR routine, the DSP3210 sets r22 (also known as exception vector table pointer (**evtp**)) to point to memory location 0x5003000, which will be the address of the first location in boot ROM when the DSP3210 is reconfigured into processor mode (the first 32 word locations in the boot ROM are configured as an exception table). Next, the **emr** register is set to enable external interrupt 1. The DSP3210 then clears **pcw**[10] without changing the other bits in **pcw**. This places the DSP3210 into processor mode which maps external memory to begin at physical address 0.

Finally, the DSP3210 goes into wait-for-interrupt mode with its memory map set for processor mode and the **evtp** pointing to the boot ROM (which now begins at location 0x5003000). Note that the DSP3210 changes its memory map **while entering** the interrupt level. Following the memory map change, the DSP3210 attempts to prefetch an instruction at what was the boot ROM address boot_error from **external** memory.

This will appear as address 0xFC on the external address bus. (Recall that the boot ROM is initially mapped to begin at location 0, but external memory A is dynamically mapped to location 0 while the DSP3210 enters the interrupt routine.) Even though this fetch is discarded (the contents of external memory at this location are unimportant), **the external system must complete this read transaction** with the DSP3210, or the DSP3210 will hang while waiting to complete this read.

When the system host decides to wake up the DSP3210 (hopefully after it has loaded executable DSP3210 instructions to memory), it places a vector to the starting address of the DSP3210 program at physical location 0. The system host then pulses IREQ1 pin (external interrupt 1 request) on the DSP3210. The DSP3210 acknowledges the interrupt via the IACK1 pin and vectors to the external interrupt 1 vector in boot ROM. The DSP3210 loads the pc shadow register (**pcsh**) from location 0 (now in external memory) to cause the DSP3210 to branch to the address supplied by the host microprocessor when returning from the interrupt routine. It then clears the **emr** register to prevent any further interrupts.

The DSP3210 then performs a return from interrupt instruction (ireturn) which will excute the *r0 = r0 instruction from the instruction shadow register (which was prefetched while the latent instruction pcw = r1 was executed following the waiti instruction). This instruction clears location 0 which signals the host processor that the boot is complete, and the DSP3210 is now executing instructions at the vector location supplied by the host.

Note that if the system is configured for *Intel*-style signaling, the **pcw** register field M/IN (bit 9) **must** be configured to zero by the DSP3210 program before the DSP3210 attempts to perform byte or 16-bit read and/or write operations. If the system is also configured for little-endian byte ordering (typically, *Intel*-style systems are also little-endian), the **pcw** register field B/LN (bit 8) should also be set to 0 prior to any byte or 16-bit read and/or write operations (see Chapter 6 of the *DSP3210 Information Manual* for more details on the operation of the DSP3210 external bus interface).

EPROM Boot

When C/PN = 1 and the boot routine test of the **pcw** shows bit 13 = 1, the boot ROM executes a program that boots from an external 8-bit wide memory device, such as an EPROM, and begins execution. This routine is typically used in embedded applications. The EPROM must be 8 bits wide and is mapped to begin at external memory address 0x60000000.

Functional Description (continued)

Figure 9 shows the organization of data in the byte-wide EPROM.

The DSP3210 first performs 32-bit reads to eight consecutive 32-bit word locations in the external EPROM, but it only recognizes the data stored in bits 0—7. The least significant byte of the first 4 words makes up the address where the remaining contents of EPROM will be stored and where the DSP3210 will branch to upon completing the boot procedure (Address). During the boot procedure, each word read from external memory contains only 1 byte of information; therefore, four word reads are required to enter 32 bits of information from external memory.

Note the ordering of the bytes which make up each 32-bit word read from external memory; they are read into the DSP3210 with the first byte read occupying the most significant byte of the address word.

The least significant bytes of the next four words form a 32-bit word that contains the number of bytes which are to be copied (Count) from the remaining locations into the memory beginning at Address. Again, note that the 4 bytes of Count are read from external memory, one byte at a time, with first byte read occupying the most significant byte of the Count word. If Count is 0 and Address is nonzero, the DSP3210 jumps directly to Address and begins execution. If the count is nonzero, the DSP3210 proceeds to load Count number of bytes from EPROM into DSP3210 memory space. Since all DSP3210 executable instructions are 32-bit words, big-endian byte ordering requires that they be loaded most significant byte first from the boot memory to create valid instructions in the DSP3210's physical memory (see Figure 9). Upon completion of the copy operation, the DSP3210 branches to Address. Note that the memory map remains in the computer mode.

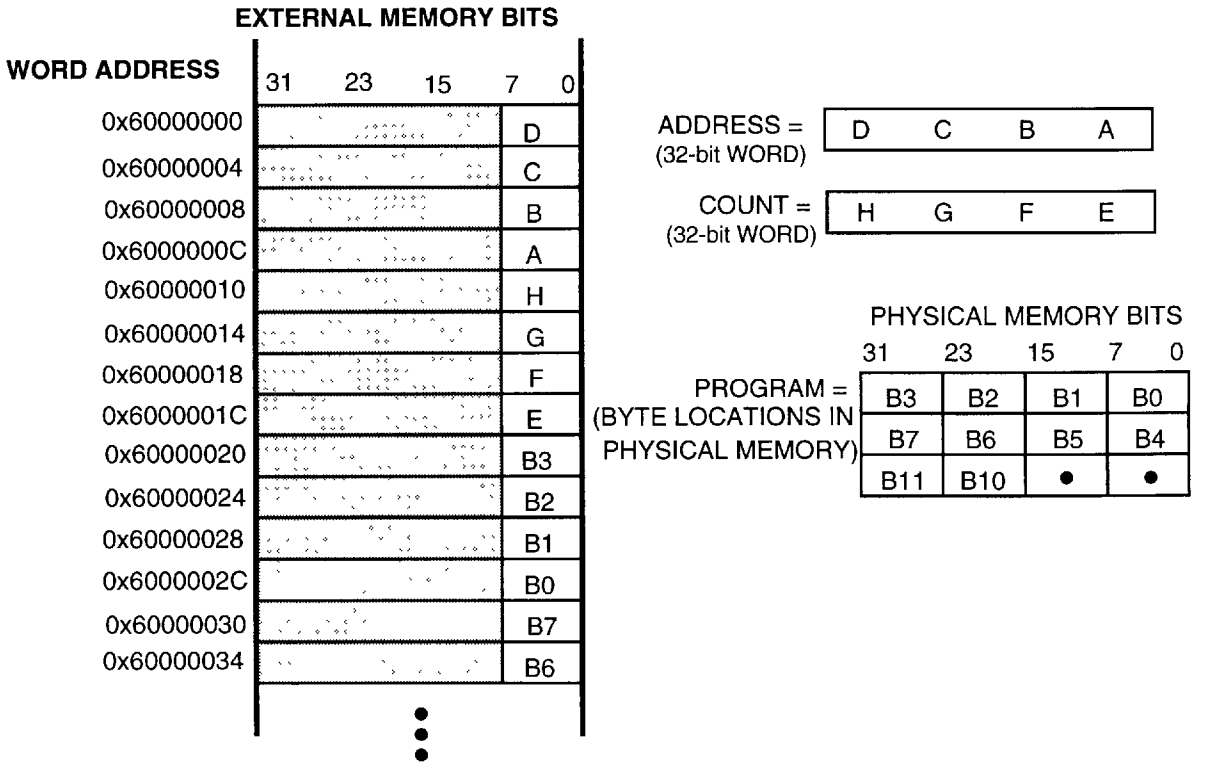


Figure 9. EPROM Data Organization (DSP3210 Begins Execution in Big-Endian Mode)

Functional Description (continued)

Special Note for *Intel*-Style Signaling Implementations (Alternate Loader Routines)

If the external memory of the DSP3210 system is configured in *Intel**- style signaling (with either big- or little-endian byte ordering), transfers directly from EPROM to **external** memory using the boot ROM procedure will not be correctly executed (since, after reset, the DSP3210 begins executing in the big-endian, *Motorola*†-style signaling mode, and the write operation to external memory is a byte operation in the boot ROM program). To correctly load programs and/or data, an alternate loader routine must first be loaded from EPROM to the DSP3210's **internal** RAM (which is unaffected by the connections to the external memory interface). This alternate loader routine will be loaded into memory as shown in Figure 9.

This program, when executed, should first correctly configure the **pcw** register (bits 8 and 9) and then perform the data move operation from external EPROM to external RAM.

If the system is configured for big-endian byte ordering and *Intel*-style signaling, the read/write routine in the boot ROM listing (following the boot_begin label) can be used in the alternate loader routine. (However, *Intel*-style signaling typically uses little-endian byte ordering). If the system is little-endian, the data from the EPROM to be loaded into memory should be ordered B0, B1, B2, B3, B4, B5, . . . to be loaded into physical memory as shown in Figure 9.

Detailed Description of Self-Test Routine

A user-callable self-test routine is included in F12 silicon revision devices. This routine performs a limited functional test of both on-chip arithmetic execution units (DAU and CAU). During execution, the self-test routine attempts to exercise as many different instruction types as possible.

The self-test routine is located in the boot ROM at location boot_rom + 0x58. Recall that the address of the boot ROM within the DSP3210 address map is relocatable, depending upon the C/PN bit in the pcw register. If in microcomputer mode (C/PN = 1), boot_rom is located at address 0x00000000; if in microprocessor mode (C/PN = 0), boot_rom is located at address 0x50030000.

* *Intel* is a trademark of Intel Corporation.
† *Motorola* is a registered trademark of Motorola, Inc.

The user invokes the self-test by calling the routine from the user program, with r1 used as the return address. During self-test execution, all user-visible registers are overwritten (except r1), therefore previous register contents are not saved. The self-test routine returns a value of zero in CAU register r2 if the test was successful. Otherwise, the routine returns a value in r2 equal to one, indicating failure. The following code sample shows typical usage of the DSP3210 self-test routine:

```
/* Define location of self-test routine using*/
/*microprocessor mode (BROM_ADDR should */
/*be defined equal to 0x00000000 if using */
/*microcomputer mode)*/

#define BROM_ADDR 0x50030000
#define SELFTEST   0x58

                    r10 = (long) BROM_ADDR
                    call r10 + SELFTEST (r1)
                    nop
                    r2–r0
                    if (ne) goto failed /* test r2 contents*/
passed:             nop
```

The self-test routine starts by automatically determining its location within the DSP3210 address map, as well as the location of on-chip RAM.
It then initializes r2 to a nonzero value and proceeds to copy the contents of boot ROM to RAM.
Registers a0 and r17 are cleared for use as checksum registers. Both the DAU and CAU execution units are simultaneously used to compute the arithmetic sum (16-bit values) of the first 256 words in RAM (the length of boot ROM). The total should add to zero; the last value stored in boot ROM (the CHECKSUM value) ensures this. The computed DAU and CAU checksum values are compared at the end of the routine.

Similarly, the CAU is used to perform a 32-bit addition of the entire RAM. The result, which should also equal zero, is tested at the end of the routine. The self-test routine also performs dummy reads of the ioc, timer, dauc, and idp registers to exercise their respective peripheral buses.

Lastly, if the computed checksums yield a zero arithmetic sum (indicating the device passed), a value of zero is stored in r2, and program flow branches back to the user via the return address stored in register r1.

Functional Description (continued)

Listing 1. Boot ROM Code

```
#define CHECKSUM      0x789B8099  /* This needs to be adjusted every time the boot ROM is modified. */
                                  /* The last location in boot ROM contains the 2's complement of this value */

#define RAMWORDS      0x7FF       /* size of ram */
#define ROMWORDS      0x0FF       /* size of rom */

.rsect  ".brom", TEXT
boot_rom:

        /* 3210 Exception Vector Table */

        if(true) pcgoto boot_begin      /* On Reset, goto boot_begin. */
        r1 = (short) pcw                /* Latent instruction- store pcw in r1 to later test for BRC (bit 13) field. */
        if(true) pcgoto boot_error      /* Bus Error */
        nop
        if(true) pcgoto boot_error      /* Illegal Opcode */
        nop
        if(true) pcgoto boot_error      /* Reserved */
        nop
        if(true) pcgoto boot_error      /* Address Error */
        nop
        if(true) pcgoto boot_error      /* DAU Overflow/Underflow */
        nop
        if(true) pcgoto boot_error      /* IEEE NAN (Not A  Number) floating-point exception*/
        nop
        if(true) pcgoto boot_error      /* Reserved */
        nop
        if(true) pcgoto boot_error      /* External Interrupt 0 */
        nop
        if(true) pcgoto boot_error      /* Timer */
        nop
        if(true) pcgoto boot_error      /* Reserved */
        nop
        if(true) pcgoto boot_test       /* Boot ROM Self-Test Routine */
        nop
        if(true) pcgoto boot_error      /* Reserved Boot ROM Routine */
        nop
        if(true) pcgoto boot_error      /* Reserved Boot ROM Routine */
        nop
        if(true) pcgoto boot_error      /* Reserved Boot ROM Routine */
        nop
        pcsh = (long) *r0               /* External Interrupt 1 */
        emr = (short) r0                /* Clear to prevent further interrupts */
        ireturn                         /* return to new value loaded into pcshadow register. */
        nop

boot_begin:
        r2 = r0 <<l 0x6000      /* Preload r2 with EPROM address (only used if EPROM boot is selected) */
        (short) r1 & 0x2000     /* Test bit 13 of r1 to test if BRC in pcw is true (pcw was stored in r1) */
        if(eq) pcgoto system_boot       /* check BRC bit in pcw register. */
        r3 = (ushort24) 0xE000  /* Load r3 with address to internal RAM for scratch memory */
        do 7 {                  /* Read "Address" and "Count" values from EPROM — 8 bytes total */
                r4 = *r2++      /* 32-bit word is read into r4 */
                nop             /* Register load from memory latency */
                *r3++ = (byte) r4       /* LSB of r4 stored.  r3 incremented by 1 byte.  Both "Address" */
        }
        r5 = *0xE004            /* fetch number of bytes to be read*/
        r1 = *0xE000            /* fetch  "Address" */
        if(eq) pcgoto boot_zero /* If # bytes = 0,  goto boot_zero */
```

Functional Description (continued)

```
        r3 = r1
        r5 = r5 – 2                 /* Initialize loop counter to # bytes – 2 */
boot_loop:
        r4 = *r2++                  /* Read from external memory (EPROM) to r4  */
        if(r5-- >= 0) pcgoto boot_loop
        *r3++ = (byte) r4           /* Store bytes to memory (Recall that these are big-endian writes) */
        goto r1 + 0                 /* Branch to "Address" */
        nop

boot_zero:
        if(eq) pcgoto boot_test     /* check start address. If equals zero, execute boot_test routine. */
        nop
        goto r1 + 0                 /* ....otherwise, branch to "Address". */
        nop

system_boot:
        r22 = r0 <<| 0x5003         /* Load r22 (evtp) with "future" address of boot ROM routine (exception table) */
        r2 = (ushort24) 0x8000      /* Set bit 15 true to enable interrupt pin 1 */
        emr = (short) r2            /* Enable interrupt pin 1 */
        r1 = r1 & (~0x0400)         /* Set bit 10 true to enable C/PN field in pcw */
        waiti                       /* Branch to interrupt */
        pcw = (short) r1            /* Change to Processor mode (Boot ROM "moves" to 0X50030000) */
                                    /* pcw = r1 is executed prior to entering ISR */
        *r0 = r0                    /* *r0 = r0 is loaded into instruction shadow register prior to entering */
                                    /* interrupt service routine.  This instruction clears location 0 after ireturn */

boot_error:
        ior10 = (short) r0          /* signal error- Both IACK signals are asserted. */
        if(true) pcgoto .           /* infinte loop */
        nop

boot_test:
        r6 = pc – (boot_test – boot_rom + 8)      /* starting address of boot rom */
        r8 = r6 + 0x3ff                 /* add size of boot rom to obtain ending address of boot rom. */
        r7 = (ushort24) 0xe000          /* offset address of RAM */
        r3 = r6 + r7                    /* starting address of RAM */
        *r3 = (long) r0
        a0 = *r3                        /* initialize a0 with zero */
        r2 = (short) 1                  /* initialize outcode to be bad */
        r4 = r6 << r0                   /* starting address of boot rom */
        do RAMWORDS {
                a1 = – a2 – (*r3++ = *r4++) * *r4       /* transfer contents of boot rom to ram */
                r4 = r4 & r8                    /* truncate r4 to limit pointer to length of ROM */
        }
        r17 = r0 # r0                   /* clear integer check sum */
        r10 = r6 ^ r7                   /* starting address of RAM */

boot_cau_dau:
        do ROMWORDS {
                a3 = float16(*r10++)    /* add up 16 bit values in DAU  */
                a2 = float16(*r10--)    /* and CAU, compare them at end */
                r18 = (short) *r10++
                r19 = (short) *r10++
                a0 = a0 + a3
                a0 = a0 + a2            /* for DAU compare */
                r17 = r17 + r18
                r17 = r17 + r19         /* for CAU compare */
        }
        r5  = r0 – r0               /* clear RAM check sum register */
        r11 = r0 | r0               /* clear RAM bar checksum */
        r19 = –1                    /* all ones for AND mask */
```

Functional Description (continued)

```
        r10 = r10 –1024          /* reset to beginning of RAM */

boot_ck_sum:
        do RAMWORDS {
                r9 = (long) *r10 /* fetch next word in RAM */
                r5 = r5 + r11    /* add in result of previous read(r11 should =0) to ram checksum register */
                r18 = r19 &~ r9  /* complement value that was in RAM */
                *r10 = r18       /* place complemented value back in RAM */
                r6 = *r10++
                r5 = r9 + r5     /* form checksum of first read; merely adds all values in RAM*/
                r11 = r6 &~ r18  /* see if complement value came back(then r11=0) */
        }
        r11 = r10 – 8192         /* reset to beginning of RAM */

boot_ending:
        *r11 = a1 = int32(a0)    /* store result of DAU additions in beginning location of RAM*/
        r15 = *0x040c            /* dummy ioc read to exercise UPB/SIO */
        r14 = *0x0414            /* dummy timer read to exercise UPB/TSC */
        r15 = (byte) ior14       /* dummy dauc read to exercise IOR */
        r12 = (long) *r11        /* store result of DAU additions in r12 */
        r16 = *0x0420            /* dummy idp read to exercise UPB/DMAC */
        r12 = r12 - r17          /* compare CAU and DAU sum; should be equal. */
        r12 = r12 + r5           /* add RAM check sum (should be zero) */
        goto r1                  /* return to "Address" */
        if(eq) r2 = r2 –1        /* set r2 = 0 if ok */

        /*fill pattern follows*/

        long    -CHECKSUM                /* last word in ROM */
```

Pin Information

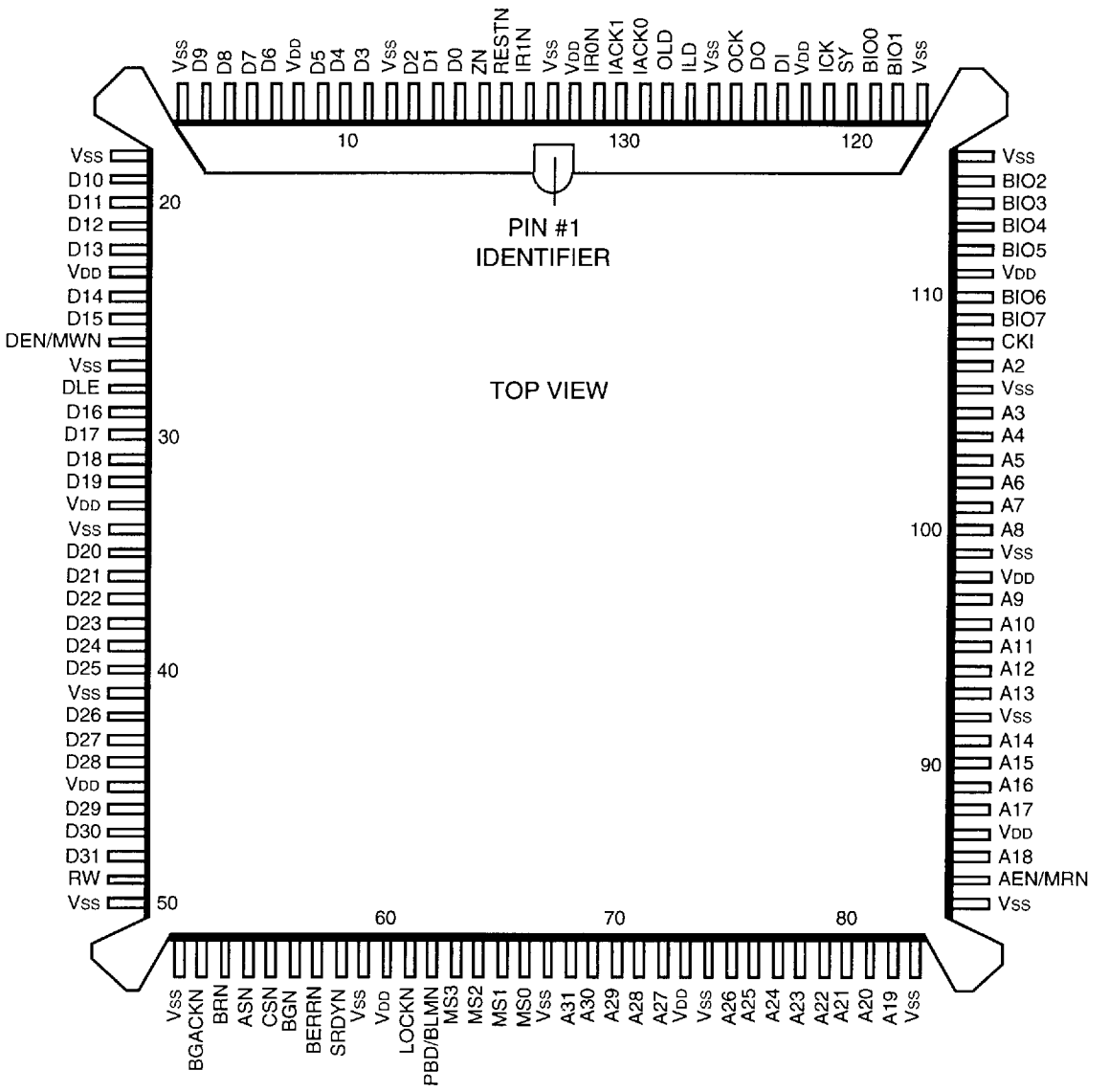


Figure 10. 132-Pin PQFP Pin Diagram

Pin Information (continued)

Table 2. Pin Descriptions

Please refer to the *AT&T DSP3210 Information Manual* for complete functional descriptions of each pin.

Pin	Symbol	Type*	Name
1	VSS	P	Ground
2	IR1N	I	Interrupt Request 1
3	RESTN	OD	Reset
4	ZN	I	3-State
5	D0	I/O(3)	Data Bus — Bit 0
6	D1	I/O(3)	Data Bus — Bit 1
7	D2	I/O(3)	Data Bus — Bit 2
8	VSS	P	Ground
9	D3	I/O(3)	Data Bus — Bit 3
10	D4	I/O(3)	Data Bus — Bit 4
11	D5	I/O(3)	Data Bus — Bit 5
12	VDD	P	Supply
13	D6	I/O(3)	Data Bus — Bit 6
14	D7	I/O(3)	Data Bus — Bit 7
15	D8	I/O(3)	Data Bus — Bit 8
16	D9	I/O(3)	Data Bus — Bit 9
17	VSS	P	Ground
18	VSS	P	Ground
19	D10	I/O(3)	Data Bus — Bit 10
20	D11	I/O(3)	Data Bus — Bit 11
21	D12	I/O(3)	Data Bus — Bit 12
22	D13	I/O(3)	Data Bus — Bit 13
23	VDD	P	Supply
24	D14	I/O(3)	Data Bus — Bit 14
25	D15	I/O(3)	Data Bus — Bit 15
26	DEN/MWN	I/O(3)	Data Enable/Write Strobe
27	VSS	P	Ground
28	DLE	I	Data Latch Enable
29	D16	I/O(3)	Data Bus — Bit 16
30	D17	I/O(3)	Data Bus — Bit 17
31	D18	I/O(3)	Data Bus — Bit 18
32	D19	I/O(3)	Data Bus — Bit 19
33	VDD	P	Supply
34	VSS	P	Ground
35	D20	I/O(3)	Data Bus — Bit 20
36	D21	I/O(3)	Data Bus — Bit 21
37	D22	I/O(3)	Data Bus — Bit 22
38	D23	I/O(3)	Data Bus — Bit 23
39	D24	I/O(3)	Data Bus — Bit 24
40	D25	I/O(3)	Data Bus — Bit 25
41	VSS	P	Ground
42	D26	I/O(3)	Data Bus — Bit 26
43	D27	I/O(3)	Data Bus — Bit 27
44	D28	I/O(3)	Data Bus — Bit 28
45	VDD	P	Supply
46	D29	I/O(3)	Data Bus — Bit 29
47	D30	I/O(3)	Data Bus — Bit 30
48	D31	I/O(3)	Data Bus — Bit 31

* I = input; O = output; P = power; OD = open drain; NC = no connection; (3) = 3-state.

Pin Information (continued)

Table 2. Pin Descriptions (continued)

Please refer to the *AT&T DSP3210 Information Manual* for complete functional descriptions of each pin.

Pin	Symbol	Type*	Name
49	RW	O(3)	Read/Write
50	VSS	P	Ground
51	VSS	P	Ground
52	BGACKN	O(3)	Bus Grant Acknowledge
53	BRN	O(3)	Bus Request
54	ASN	O(3)	Address Strobe
55	CSN	O(3)	Cycle Start
56	BGN	I	Bus Grant
57	BERRN	I	Bus Error
58	SRDYN	I	Synchronous Ready
59	VSS	P	Ground
60	VDD	P	Supply
61	LOCKN	O(3)	Bus Lock
62	PBD/BLMN	O(3)	Page Break Detect/Block Move
63	MS3	O(3)	Memory Select 3
64	MS2	O(3)	Memory Select 2
65	MS1	O(3)	Memory Select 1
66	MS0	O(3)	Memory Select 0
67	VSS	P	Ground
68	A31	O(3)	Address Bus — Bit 31
69	A30	O(3)	Address Bus — Bit 30
70	A29	O(3)	Address Bus — Bit 29
71	A28	O(3)	Address Bus — Bit 28
72	A27	O(3)	Address Bus — Bit 27
73	VDD	P	Supply
74	VSS	P	Ground
75	A26	O(3)	Address Bus — Bit 26
76	A25	O(3)	Address Bus — Bit 25
77	A24	O(3)	Address Bus — Bit 24
78	A23	O(3)	Address Bus — Bit 23
79	A22	O(3)	Address Bus — Bit 22
80	A21	O(3)	Address Bus — Bit 21
81	A20	O(3)	Address Bus — Bit 20
82	A19	O(3)	Address Bus — Bit 19
83	VSS	P	Ground
84	VSS	P	Ground
85	AEN/MRN	I/O(3)	Address Enable/Read Strobe
86	A18	O(3)	Address Bus — Bit 18
87	VDD	P	Supply
88	A17	O(3)	Address Bus — Bit 17
89	A16	O(3)	Address Bus — Bit 16
90	A15	O(3)	Address Bus — Bit 15
91	A14	O(3)	Address Bus — Bit 14
92	VSS	P	Ground
93	A13	O(3)	Address Bus — Bit 13
94	A12	O(3)	Address Bus — Bit 12
95	A11	O(3)	Address Bus — Bit 11
96	A10	O(3)	Address Bus — Bit 10

* I = input; O = output; P = power; OD = open drain; NC = no connection; (3) = 3-state.

Pin Information (continued)

Table 2. Pin Descriptions (continued)

Please refer to the *AT&T DSP3210 Information Manual* for complete functional descriptions of each pin.

Pin	Symbol	Type*	Name
97	A9	O(3)	**Address Bus — Bit 9**
98	VDD	P	**Supply**
99	VSS	P	**Ground**
100	A8	O(3)	**Address Bus — Bit 8**
101	A7	O(3)	**Address Bus — Bit 7**
102	A6	O(3)	**Address Bus — Bit 6**
103	A5	O(3)	**Address Bus — Bit 5**
104	A4	O(3)	**Address Bus — Bit 4**
105	A3	O(3)	**Address Bus — Bit 3**
106	VSS	P	**Ground**
107	A2	O(3)	**Address Bus — Bit 2**
108	CKI	I	**Clock In**
109	BIO7	I/O(3)	**Bit I/O — Bit 7**
110	BIO6	I/O(3)	**Bit I/O — Bit 6**
111	VDD	P	**Supply**
112	BIO5	I/O(3)	**Bit I/O — Bit 5**
113	BIO4	I/O(3)	**Bit I/O — Bit 4**
114	BIO3	I/O(3)	**Bit I/O — Bit 3**
115	BIO2	I/O(3)	**Bit I/O — Bit 2**
116	VSS	P	**Ground**
117	VSS	P	**Ground**
118	BIO1	I/O(3)	**Bit I/O — Bit 1**
119	BIO0	I/O(3)	**Bit I/O — Bit 0**
120	SY	I/O(3)	**Sync**
121	ICK	I/O(3)	**Input Clock**
122	VDD	P	**Supply**
123	DI	I	**Serial Data In**
124	DO	O(3)	**Serial Data Out**
125	OCK	I/O(3)	**Output Clock**
126	VSS	P	**Ground**
127	ILD	I/O(3)	**Input Load**
128	OLD	I/O(3)	**Output Load**
129	IACK0	O(3)	**Interrupt Acknowledge 0**
130	IACK1	O(3)	**Interrupt Acknowledge 1**
131	IR0N	I	**Interrupt Request 0**
132	VDD	P	**Supply**

* I = input; O = output; P = power; OD = open drain; NC = no connection; (3) = 3-state.

Absolute Maximum Ratings

Stresses in excess of the Absolute Maximum Ratings can cause permanent damage to the device. These are absolute stress ratings only. Functional operation of the device is not implied at these or any other conditions in excess of those given in the operational sections of the data sheet. Exposure to Absolute Maximum Ratings for extended periods can adversely affect device reliability.

Parameter	Value
Voltage Range on Any Pin with Respect to Ground	–0.5 V to +6 V
Ambient Temperature Range	–40 °C to +120 °C
Storage Temperature Range	–65 °C to +150 °C

Warning: All CMOS devices are prone to latch-up if excessive current is injected to/from the substrate. To prevent latch-up at powerup, no input pin should be subjected to input voltages greater than V_{IL} or less than V_{SS} – 0.5 V before V_{DD} is applied. After powerup, input should not be greater than V_{DD} + 0.5 V or less than V_{SS} – 0.5 V.

Handling Precautions

All MOS devices must be handled with certain precautions to avoid damage due to the accumulation of static charge. **Although input protection circuitry has been incorporated into the devices to minimize the effect of this static buildup, proper precautions should be taken to avoid exposure to electrostatic discharge during handling and mounting.** AT&T employs a human-body model for ESD susceptibility testing. Since the failure voltage of electronic devices is dependent on the current and voltage and, hence, the resistance and capacitance, it is important that standard values be employed to establish a reference by which to compare test data. Values of 100 pF and 1500 Ω are the most common and are the values used in the AT&T human-body model test circuit. The breakdown voltage for the DSP3210 is greater than 2000 V, according to the human-body model. ESD data for the charged-device model is available on request.

Electrical Specifications

The parameters below are valid for the following conditions: T_A = 0 °C to 70 °C ($T_J \leq$ 125 °C); V_{DD} = 5 V ± 5%; V_{SS} = 0 V.

Parameter	Symbol	Min	Max	Unit
Input Voltage except CKI:				
Low	V_{IL}	—	0.8	V
High	V_{IH}	2.0	—	V
Input Voltage for CKI:				
Low	V_{ILC}	—	0.8	V
High	V_{IHC}	2.4	—	V
Output Voltage:				
(I_{OL} = 6 mA) Low (TTL)	V_{OL}	—	0.4	V
(I_{OL} = 50 µA) Low (CMOS)	V_{OL}	—	0.2	V
(I_{OH} = –2 mA) High (TTL)	V_{OH}	2.4	—	V
(I_{OH} = –50 µA) High (CMOS)	V_{OH}	V_{DD} – 0.2	—	V
Output Current:				
(V_{OL} = 0.4 V) Low	I_{OL}	—	6.0	mA
(V_{OH} = 2.4 V) High	I_{OH}	–2.0	—	mA
Input Leakage except ZN:				
(V_{IL} = 0 V) Low	I_{IL}	–5	—	µA
(V_{IH} = 5.25 V) High	I_{IH}	—	5	µA
Input Leakage for ZN:				
(V_{IL} = 0 V) Low	I_{IL}	–450	—	µA
(V_{IH} = 5.25 V) High	I_{IH}	—	5	µA
Output Hi-Z Current:				
($V_{applied}$ = 0.0 V) Low	I_{OZL}	–10	—	µA
($V_{applied}$ = 5.25 V) High	I_{OZH}	—	10	µA
Input, Output, I/O Capacitance	C_I	—	10	pF
55 MHz:				
Power Supply Current* (V_{DD} = 5.25 V, CKI pd = 18 ns)	I_{DD}	—	220	mA
Power Dissipation† (V_{DD} = 5.25 V, CKI pd = 18 ns)	PD	—	1.20	W
66 MHz:				
Power Supply Current* (V_{DD} = 5.25 V, CKI pd = 15 ns)	I_{DD}	—	285	mA
Power Dissipation† (V_{DD} = 5.25 V, CKI pd = 15 ns)	PD	—	1.50	W

* Current in the input buffers is highly dependent on the input voltage level. At full CMOS levels, essentially no dc current is drawn, but for levels near the threshold of 1.4 V, high and unstable levels of current may flow. There are 56 input buffers on the chip. If all inputs are connected to a dc voltage of 1.4 V, an additional current in the range of 100 mA can be drawn.

† The power dissipation listed is for internal power dissipation only. Total power dissipation can be calculated based on the system configuration by adding $C * V_{DD}^2 * f$ for each output, where C is the load capacitance, and f is the output frequency.

Timing Specifications

The characteristics listed are valid under the following conditions: T_A = 0 °C to 70 °C; V_{DD} = 5 V ± 5%; V_{SS} = 0 V.

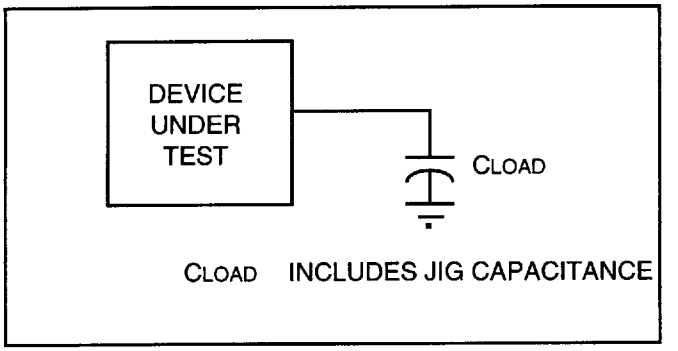


Figure 11. ac Testing Load Circuit

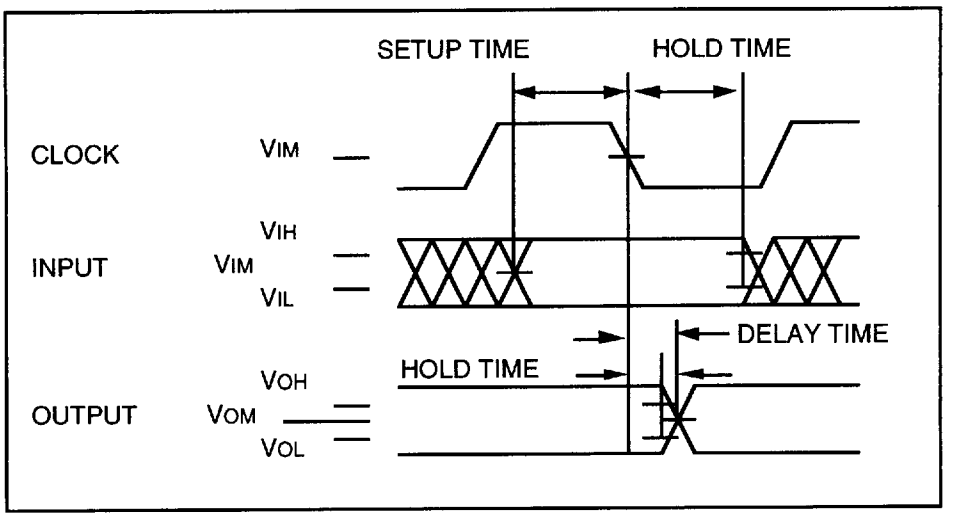


Figure 12. ac Testing Input/Output Waveform

Test conditions for inputs (unless otherwise specified):

- Rise and fall times of 4 ns or less.
- Timing reference level for setup times is V_{IM} = 1.5 V.
- Timing reference levels for hold times are V_{IH}, V_{IL}.

Test conditions for outputs (unless otherwise specified):

- C_{LOAD} = 50 pF unless otherwise stated.
- Timing reference level for delay times is V_{OM} = 1.5 V.
- Timing reference levels for hold times are V_{OH}, V_{OL}.

Timing Specifications (continued)

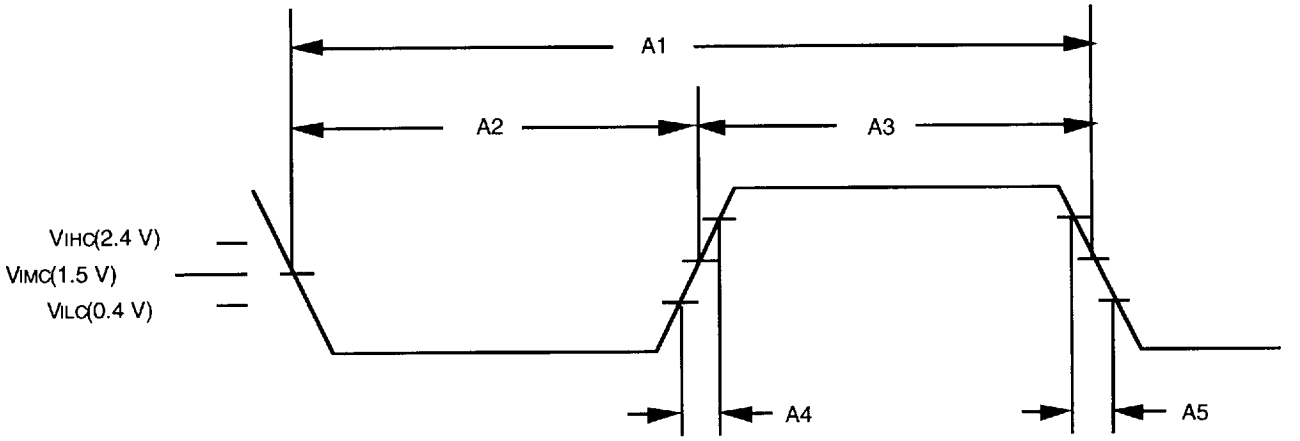


Figure 13. CKI Timing Specifications

Table 3. Timing Requirements for CKI (See Figure 13.)

			55 MHz		66 MHz		
Num*	Symbol	Parameter	Min	Max	Min	Max	Unit
A1	tCKILCKIL	Clock In Period†	18	—‡	15	—‡	ns
A2	tCKILCKIH	Clock In Low	8.1	—	6.75	—	ns
A3	tCKIHCKIL	Clock In High	8.1	—	6.75	—	ns
A4	tCKIRISE	Clock In Rise Time	—	4	—	4	ns
A5	tCKIFALL	Clock In Fall Time	—	4	—	4	ns

* This column in all timing requirements and/or characteristic tables refers to the internal alphanumeric symbol in the timing diagrams.
† tCKILCKIL is referred to as T for specifications that are a function of the period of CKI.
‡ This device is static. However, the maximum period guaranteed by test is 50 ns.

Table 4. Timing Requirements for Synchronous Bus Interface Inputs (See Figures 14 and 16.)

			55 MHz		66 MHz		
Num	Symbol	Parameter	Min	Max	Min	Max	Unit
B1	tSRNVCKIL	SRDYN Setup to CKI Low	5	—	3	—	ns
B2	tCKILSRNX	SRDYN Hold after CKI Low	3	—	2	—	ns
B3	tBENVCKIL	BERRN Setup to CKI Low	5	—	3	—	ns
B4	tCKILBRNX	BERRN Hold after CKI Low	3	—	2	—	ns
B5	tBGNVCKIL	BGN Setup to CKI Low	5	—	1.75	—	ns
B6	tCKILBGNX	BGN Hold after CKI Low	3	—	3	—	ns
B7	tDVCKIL	DATAin Setup to CKI Low	0	—	0	—	ns
B8	tCKILDX	DATAin Hold after CKI Low	6	—	6	—	ns

Timing Specifications (continued)

Table 5. Timing Characteristics for Synchronous Bus Interface Outputs (See Figures 14 and 16.)

Pin(s)	Max Load	Buffer Type	Delay Time		Hold Time	
			Num	Symbol*	Num	Symbol†
D0—D31	150 pF	A	B9	tCKILDV	B10	tCKILDX
A2—A31	150 pF	A	B11	tCKILAV	B12	tCKILAX
MS0—MS3	150 pF	A	B13	tCKILMSV	B14	tCKILMSX
CSN	50 pF	A	B15	tCKILCSNV	B16	tCKILCSNX
ASN	50 pF	C	B17	tCKIHASNL	—	—
	50 pF		B19	tCKILASNH	—	—
RW	50 pF	A	B21	tCKILRWV	B22	tCKILRWX
PBD/BLMN	50 pF	A	B23	tCKILPBV	B24	tCKILPBX
MRN	50 pF	B	B25	tCKIHMRNL	—	—
	50 pF		B27	tCKILMRNH	—	—
MWN	50 pF	B	B29	tCKIHMWNV	—	—
LOCKN	50 pF	A	B31	tCKILLONV	B32	tCKILLONX
BRN	50 pF	A	B33	tCKILBRNV	B34	tCKILBRNX
BGACKN	50 pF	B	B35	tCKILBGANL	B36	tCKILBGANHX
	50 pF	B	B37	tCKIHBGANH	B38	tCKIHBGANLX

* Description for all Delay Time Symbols in this table is CKI (Low, High) to Signal (Valid, High, Low).
† Description for all Hold Time Symbols in this table is Signal Value Held after CKI (Low, High).

Table 6. Timing Characteristics for Synchronous Delay/Hold Times (See Figures 14 and 16.)

		55 MHz						66 MHz						
Synchronous Delay Times (Odd Num)		**Min**			**Max**			**Min**			**Max**			**Unit**
Buffer Type		**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	
VOM	50 pF	—	—	—	15	14	13	—	—	—	13	12	12	ns
	100 pF	—	NA*	NA	18	NA	NA	—	NA	NA	16	NA	NA	ns
	150 pF	—	NA	NA	21	NA	NA	—	NA	NA	19	NA	NA	ns
VOL	50 pF	—	—	—	17	16	15	—	—	—	15	14	14	ns
	100 pF	—	NA	NA	22	NA	NA	—	NA	NA	20	NA	NA	ns
	150 pF	—	NA	NA	27	NA	NA	—	NA	NA	25	NA	NA	ns
VOH	50 pF	—	—	—	17	16	15	—	—	—	15	14	14	ns
	100 pF	—	NA	NA	20	NA	NA	—	NA	NA	18	NA	NA	ns
	150 pF	—	NA	NA	23	NA	NA	—	NA	NA	21	NA	NA	ns
Synchronous Hold Times (Even Num)		**Min**			**Max**			**Min**			**Max**			**Unit**
Buffer Type		**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	
VOL	50, 100, 150 (pF)	3	2	2	—	—	—	3	2	2	—	—	—	ns
VOH	50, 100, 150 (pF)	3	3	3	—	—	—	3	3	3	—	—	—	ns

* NA = not applicable.

Timing Specifications (continued)

Table 7. Timing Relationships for Synchronous Bus Interface Operation (55 MHz) (See Figure 14.)

Note: **All** signals referred to in this table are equally loaded with 50 pF. Signal timing relationships with respect to DATAin supercede B7 and B8 timing requirements.

Num	Symbol	Parameter	Min	Max	Unit
B39	tCKILDE	Data Bus Low Z after CKI Low	0	—	ns
B40	tCKILDZ	Data Bus Hi Z after CKI Low	—	12	ns
B41	tAVDINV	Address† Valid to DATAin Valid	—	2 * T – 11 + N * T	ns
B42	tAVASNL	Address Valid to Address Strobe Low	0.5 * T – 2‡	0.5 * T + 3	ns
B43	tASNLASNH	Address Strobe Width	1.5 * T – 5 + N * T	—	ns
B44	tASNHAX	Address Hold after Address Strobe High	–1	—	ns
B45	tASNLDINV	Address Strobe Low to DATAin Valid	—	1.5 * T – 11 + N * T	ns
B46	tDINVASNH	DATAin Setup to Address Strobe High	8	—	ns
B47	tASNHDX	DATAin Hold after Address Strobe High	0	—	ns
B48	tASNHDZ	Data Bus Hi Z after Address Strobe High	–3.0	2	ns
B49	tAVMRNL	Address Valid to Read Strobe Low	0.5 * T – 2	—	ns
B50	tMRNLMRNH	Read Strobe Width	1.5 * T – 4 + N * T	—	ns
B51	tMRNHMRNL	Read Strobe High to Read Strobe Low	0.5 * T – 1	—	ns
B52	tMRNLDINV	Read Strobe Low to DATAin Valid	—	1.5 * T – 11 + N * T	ns
B53	tDINVMGNH	DATAin Setup to Read Strobe High	9	—	ns
B54	tMRNHDX	DATAin Hold after Read Strobe High	–1	—	ns
B55	tMRNHDE	Read Strobe High to Data Bus Low Z	T – 4	—	ns
B56	tRWVDE	Read-Write Asserted to Data Bus Low Z	T – 4	—	ns
B57	tAVMWNL	Address Valid to Write Strobe Low	0.5 * T – 2	—	ns
B58	tMWNLMWNH	Write Strobe Width	T + M * T – 1§	—	ns
B59	tMWNHAX	Address Hold after Write Strobe High	0.5 * T – 5	—	ns
B60	tDVMWNH	Data Bus Valid to Write Strobe High: 0—1 Configured Wait-states 2—3 Configured Wait-states	 0.5 * T – 2 0.5 * T – 2 + (M – 1) * T	 — —	 ns ns
B61	tMWNHDZ	Data Bus Hi Z after Write Strobe High	0.5 * T – 6	0.5 * T + (N – M) * T + 0	ns
B77	tASNLDV	Address Strobe Low to Data Bus Valid: 0 Wait-states 1—3 Wait-states	 — —	 0.5 * T + 2 1.5 * T + 2	 ns ns

† Address refers to A2—A31 and MS0—MS3.
‡ T = tCKILCKIL; N = number of wait-states (programmed plus external).
§ M = number of wait-states configured in pcw.

Timing Specifications (continued)

Table 8. Timing Relationships for Synchronous Bus Interface Operation (66 MHz) (See Figure 14.)

Note: **All** signals referred to in this table are equally loaded with 50 pF. Signal timing relationships with respect to DATAin supercede B7 and B8 timing requirements.

Num	Symbol	Parameter	Min	Max	Unit
B39	tCKILDE	Data Bus Low Z after CKI Low	0	—	ns
B40	tCKILDZ	Data Bus Hi Z after CKI Low	—	10	ns
B41	tAVDINV	Address† Valid to DATAin Valid	—	2 * T – 10 + N * T	ns
B42	tAVASNL	Address Valid to Address Strobe Low	0.5 * T – 2‡	0.5 * T + 2	ns
B43	tASNLASNH	Address Strobe Width	1.5 * T – 4 + N * T	—	ns
B44	tASNHAX	Address Hold after Address Strobe High	–1	—	ns
B45	tASNLDINV	Address Strobe Low to DATAin Valid	—	1.5 * T – 10 + N * T	ns
B46	tDINVASNH	DATAin Setup to Address Strobe High	7	—	ns
B47	tASNHDX	DATAin Hold after Address Strobe High	0	—	ns
B48	tASNHDZ	Data Bus Hi Z after Address Strobe High	–3.0	1	ns
B49	tAVMRNL	Address Valid to Read Strobe Low	0.5 * T – 2	—	ns
B50	tMRNLMRNH	Read Strobe Width	1.5 * T – 4 + N * T	—	ns
B51	tMRNHMRNL	Read Strobe High to Read Strobe Low	0.5 * T	—	ns
B52	tMRNLDINV	Read Strobe Low to DATAin Valid	—	1.5 * T – 10 + N * T	ns
B53	tDINVMGNH	DATAin Setup to Read Strobe High	8	—	ns
B54	tMRNHDX	DATAin Hold after Read Strobe High	–1	—	ns
B55	tMRNHDE	Read Strobe High to Data Bus Low Z	T – 4	—	ns
B56	tRWVDE	Read-Write Asserted to Data Bus Low Z	T – 4	—	ns
B57	tAVMWNL	Address Valid to Write Strobe Low	0.5 * T – 2	—	ns
B58	tMWNLMWNH	Write Strobe Width	T + M * T – 1§	—	ns
B59	tMWNHAX	Address Hold after Write Strobe High	0.5 * T – 4	—	ns
B60	tDVMWNH	Data Bus Valid to Write Strobe High: 0—1 Configured Wait-states 2—3 Configured Wait-states	 0.5 * T – 2 0.5 * T – 2 + (M – 1) * T	 — —	 ns ns
B61	tMWNHDZ	Data Bus Hi Z after Write Strobe High	0.5 * T – 6	0.5 * T + (N – M) * T	ns
B77	tASNLDV	Address Strobe Low to Data Bus Valid: 0 Wait-states 1—3 Wait-states	 — —	 0.5 * T + 2 1.5 * T + 2	 ns ns

† Address refers to A2—A31 and MS0—MS3.
‡ T = tCKILCKIL; N = number of wait-states (programmed plus external).
§ M = number of wait-states configured in pcw.

Timing Specifications (continued)

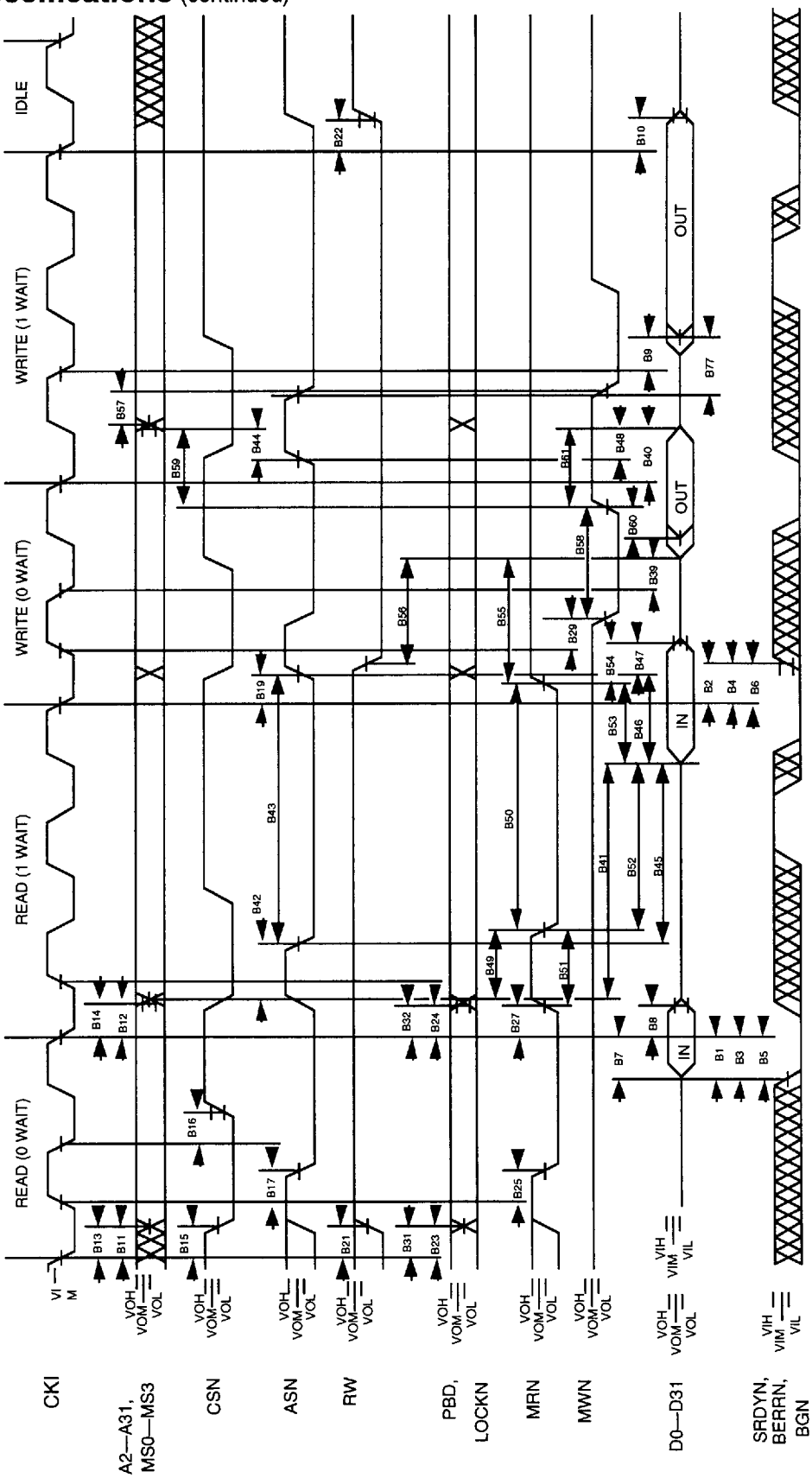


Figure 14. Synchronous Bus Interface Timing

Timing Specifications (continued)

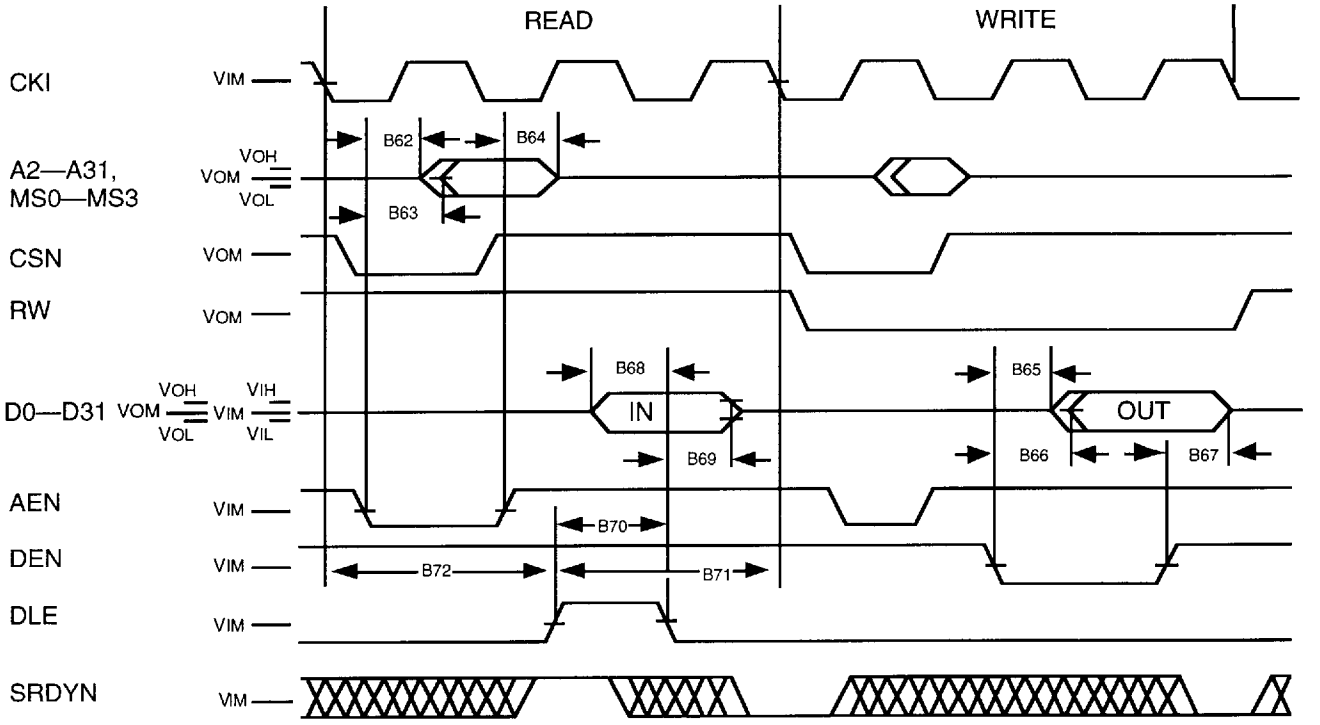


Figure 15. Asynchronous Bus Interface Timing

Table 9. Timing Relationships for Asynchronous Bus Interface Operation (See Figure 15.)

			55 MHz		66 MHz		
Num	**Symbol**	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Unit**
B62	tAENLAE	Address* Enable Low to Address Low Z	0	—	0	—	ns
B63	tAENLAV:	Address Enable Low to Address Valid					
	50 pF		—	12	—	10	ns
	100 pF		—	16	—	14	ns
	150 pF		—	19	—	18	ns
B64	tAENHAZ	Address Enable High to Address High Z	—	7	—	5	ns
B65	tDENLDE	Data Enable Low to Data Bus Low Z	0	—	0	—	ns
B66	tDENLDV:	Data Enable Low to Data Bus Valid					
	50 pF		—	12	—	10	ns
	100 pF		—	16	—	14	ns
	150 pF		—	19	—	18	ns
B67	tDENHDZ	Data Enable High to Data Bus High Z	—	7	—	5	ns
B68	tDVDLEL	DATAin Setup to Data Latch Enable Low	3	—	3	—	ns
B69	tDLELDXL	DATAin Hold after Data Latch Enable Low	3	—	3	—	ns
B70	tDLEHDLEL	Data Latch Enable Width	8	—	8	—	ns
B71	tDLEHCKIL	Data Latch Enable High to CKI Low when Asserted during State Ending Read Transaction (Bus interface state machine is in state E, and SRDYN is asserted.)	3	—	3	—	ns
B72	tCKILDLEH	CKI Low to Data Latch Enable High after State Ending Read Transaction	5	—	4	—	ns

* Address refers to A2—A31 and MS0—MS3.

Timing Specifications (continued)

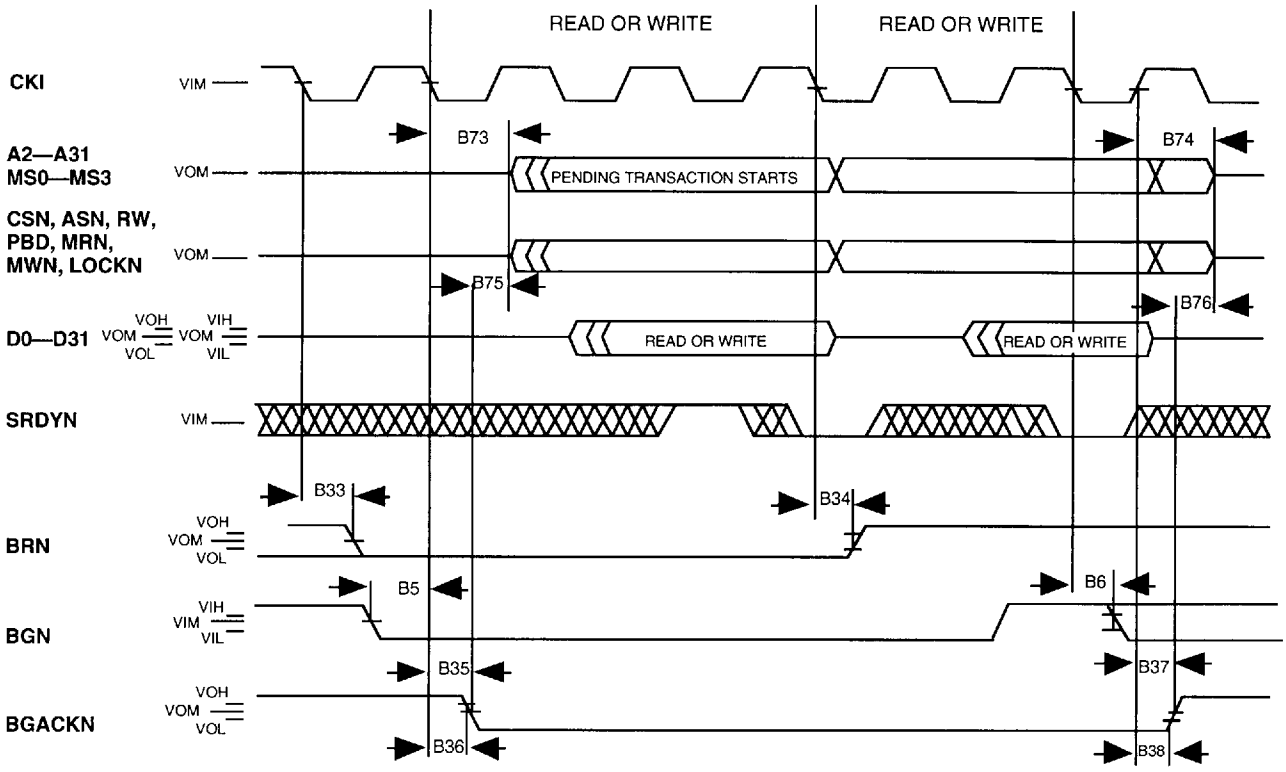


Figure 16. Bus Arbitration Timing

Table 10. Timing Characteristics for Bus Arbitration (See Figure 16.)

			55 MHz		66 MHz		
Num	**Symbol**	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Unit**
B73	tCKILBISE	CKI Low to Bus Interface Signals* Low Z	2	—	2	—	ns
B74	tCKIHDZ	CKI High to Bus Interface Signals High Z	2	12	2	10	ns
B75	tBGANLBISE	BGACKN Low to Bus Interface Signals Low Z	–4	—	–4	—	ns
B76	tBGANHBISZ	BGACKN High to Bus Interface Signals High Z	–4	2	–4	0	ns

* Bus interface signals are defined to be A2—A31, MS0—MS3, CSN, ASN, RW, PBD, MRN, MWN, and LOCKN.

Timing Specifications (continued)

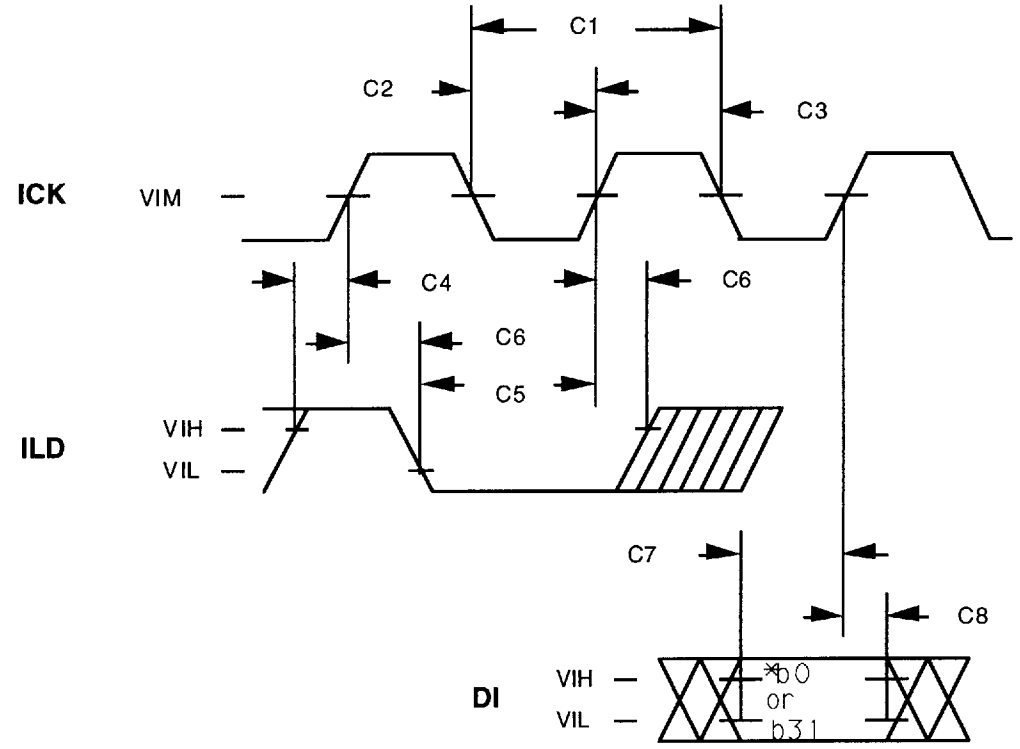


Figure 17a. Serial Input Timing (IIC = 0)

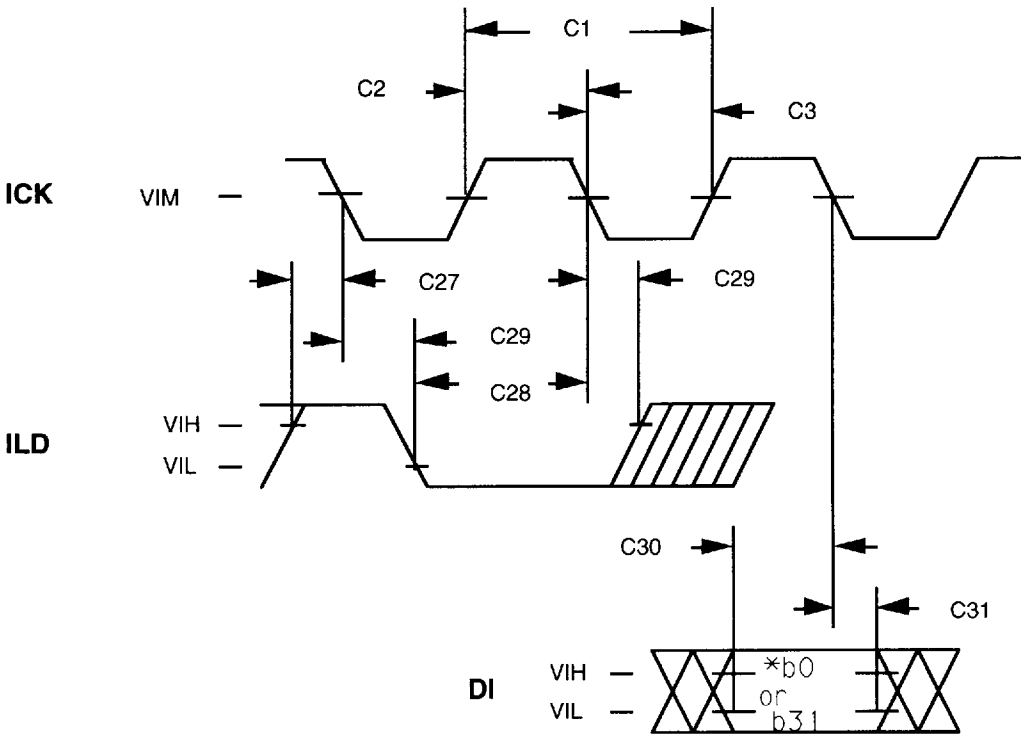


Figure 17b. Serial Input Timing (IIC = 1)

* b0 if LSB first mode, b31 if MSB first mode is used.

Timing Specifications (continued)

Table 11. Timing Requirements for Serial Inputs (See Figure 17a and 17b.)

			55 MHz	66 MHz		
Num	**Symbol**	**Parameter**	**Min**	**Min**	**Max**	**Unit**
C1	tICKLICKL	Serial Input Clock Period	40	40	—*	ns
C2	tICKLICKH	Serial Input Clock Low Time	18	18	—	ns
C3	tICKHICKL	Serial Input Clock High Time	18	18	—	ns
C4	tILDHICKH	Input Load High Setup Time	8	6	—	ns
C5	tILDLICKH	Input Load Low Setup Time	8	6	—	ns
C6	tICKHILDX	Input Load Hold Time	0	0	—	ns
C7	tDIVICKH	Serial Input Data Setup	7	6	—	ns
C8	tICKHDIX	Serial Input Data Hold	1	1	—	ns
C27	tILDHICKL	Input Load High Setup Time	8	6	—	ns
C28	tILDLICKL	Input Load Low Setup Time	8	6	—	ns
C29	tICKLILDX	Input Load Hold Time	0	0	—	ns
C30	tDIVICKL	Serial Input Data Setup	7	6	—	ns
C31	tICKLDIX	Serial Input Data Hold	1	1	—	ns

* The DSP3210 is fully static. However, the maximum period guaranteed by test is 200 ns.

Timing Specifications (continued)

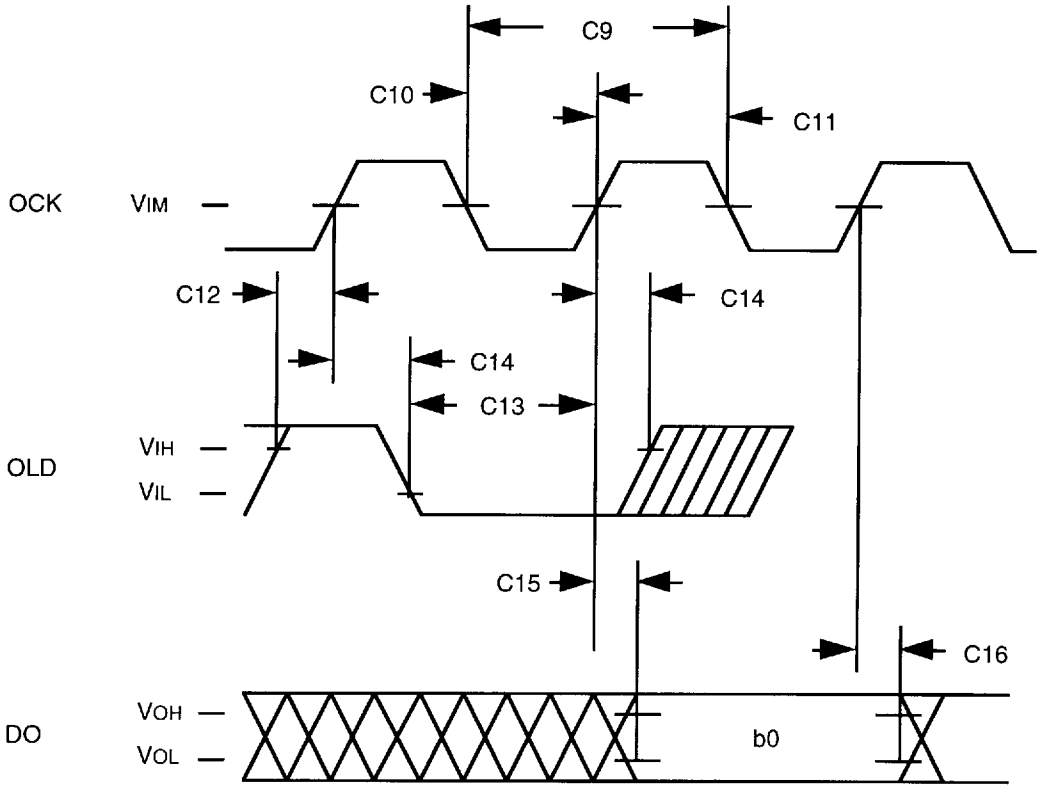


Figure 18. Serial Output Timing

Table 12. Timing Requirements for Serial Outputs (See Figure 18.)

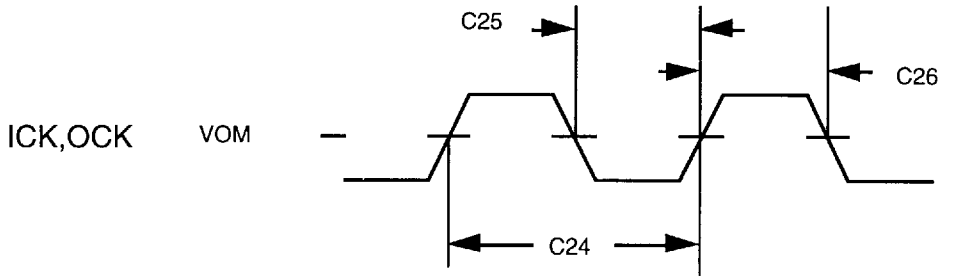
			55 MHz	66 MHz		
Num	**Symbol**	**Parameter**	**Min**	**Min**	**Max**	**Unit**
C9	tOCKLOCKL	Serial Output Clock Period	40	40	—*	ns
C10	tOCKLOCKH	Serial Output Clock Low Time	18	18	—	ns
C11	tOCKHOCKL	Serial Output Clock High Time	18	18	—	ns
C12	tOLDHOCKH	Output Load High Setup Time	8	6	—	ns
C13	tOLDLOCKH	Output Load Low Setup Time	8	6	—	ns
C14	tOCKHOLDX	Output Load Hold Time	0	0	—	ns

* This device is static. However, the maximum period guaranteed by test is 200 ns.

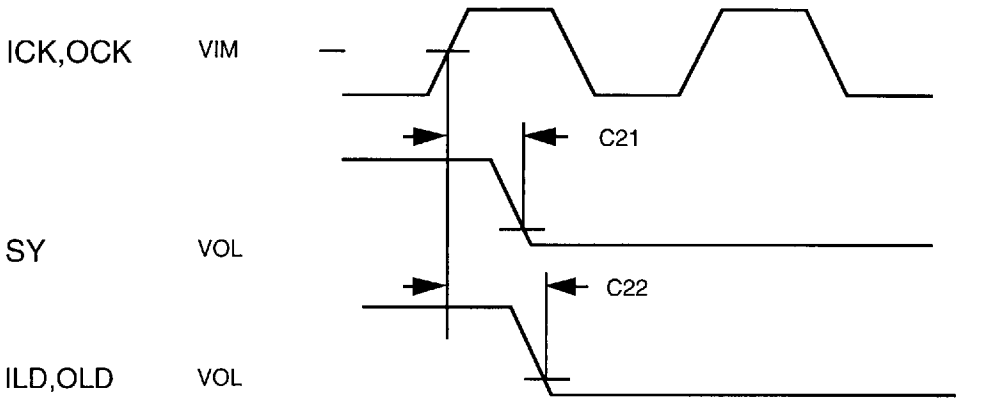
Table 13. Timing Characteristics for Serial Outputs (See Figure 18.)

			55 MHz		66 MHz		
Num	**Symbol**	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Unit**
C15	tOCKHDOV	Serial Output Data Delay	—	23	—	20	ns
C16	tOCKHDOX	Serial Output Data Hold	2	—	2	—	ns

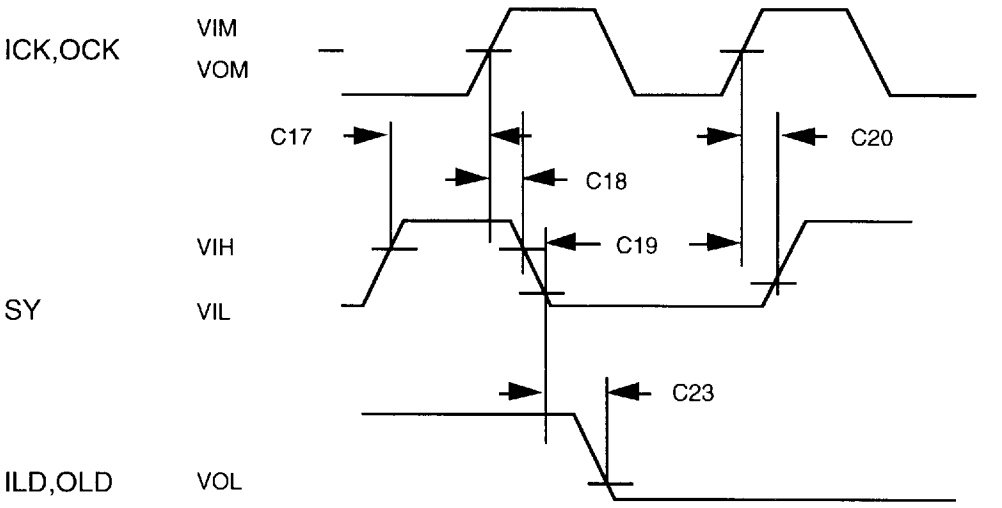
Timing Specifications (continued)



A. Internal ICK/OCK



B. Internal ILD/OLD/SY, External ICK/OCK



C. External SY, Internal ILD/OLD
External or Internal ICK/OCK

Figure 19. ac Serial Clock Generator Timing

Timing Specifications (continued)

Table 14. Timing Requirements for Serial Clock Generation (See Figure 19.)

			55 MHz	66 MHz		
Num	**Symbol**	**Parameter**	**Min**	**Min**	**Max**	**Unit**
C17	tSYHICKH* tSYHOCKH	SY High Setup	8	6	—	ns
C18	tICKHSYX tOCKHSYX	SY High Hold Time	0	0	—	ns
C19	tSYLICKH tSYLOCKH	SY Low Setup	8	6	—	ns
C20	tICKHSYX tOCKHSYX	SY Low Hold Time	0	0	—	ns

* ICK or OCK as selected by IOC[1].

Table 15. Timing Characteristics for Serial Clock Generation (See Figure 19.)

			55 MHz		66 MHz		
Num	**Symbol**	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Unit**
C21	tICKHSYL tOCKHSYL	Internal SY Delay	—	23	—	18	ns
C22	tICKHILDL tOCKHOLDL	Internal Load Delay	—	23	—	18	ns
C23	tSYLILDL tSYLOLDL	SY Low to Load Low Delay	—	23	—	18	ns
C24	tICKHICKH tOCKHOCKH	Internal SIO Clock Period	ICN * 4 * T – 10†	ICN * 4 * T + 10	ICN * 4 * T – 10*	ICN * 4 * T + 10	ns
C25	tICKLICKH tOCKLOCKH	Internal SIO Clock Low Time	ICN *2 * T – 10	ICN * 2 * T + 10	ICN * 2 * T – 10	ICN * 2 * T + 10	ns
C26	tICKHICKL tICKHICKL	Internal SIO Clock High Time	ICN * 2 * T – 10	ICN * 2 * T + 10	ICN * 2 * T – 10	ICN * 2 * T + 10	ns

† ICN refers to the value of the four bit field IOC[23—20]. When ICN = 0, the period of the clock is infinity. (The clock is disabled from counting.)

Timing Specifications (continued)

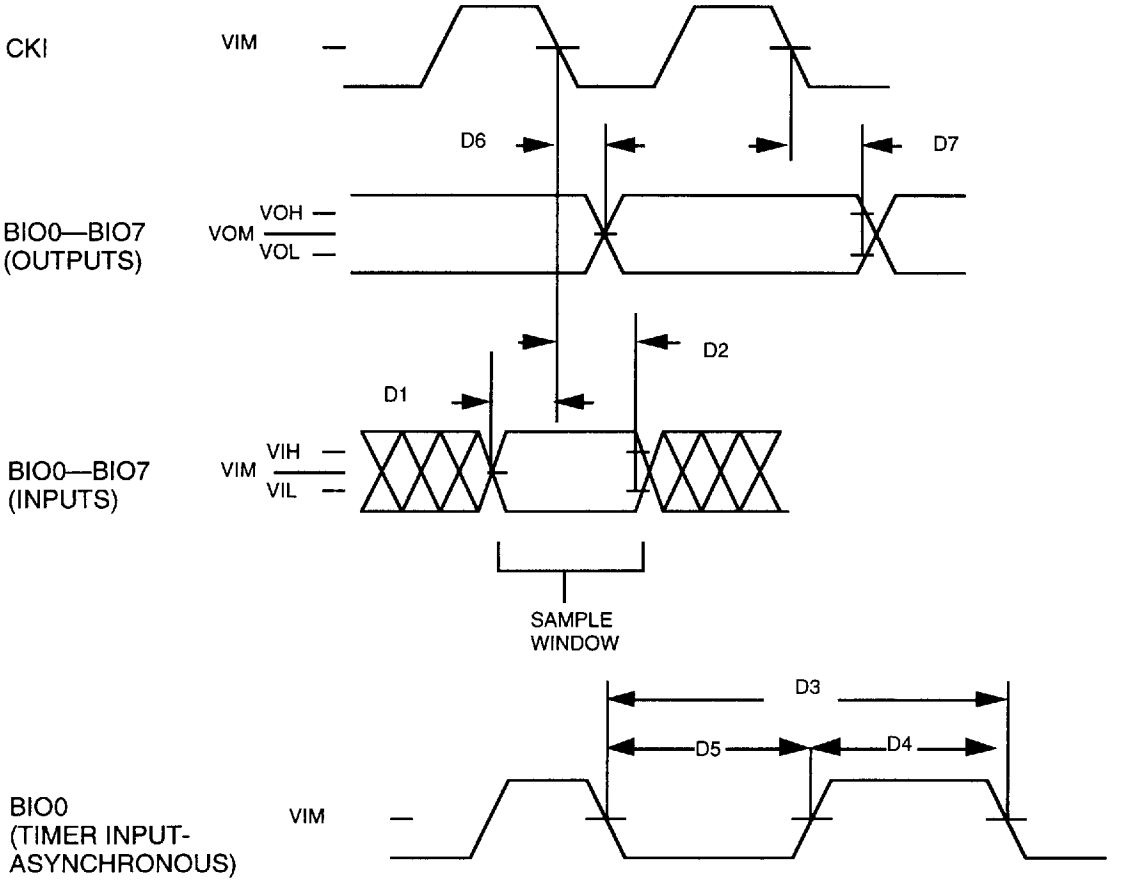


Figure 20. Bit I/O Timing

Table 16. Timing Requirements for Bit I/O (See Figure 20.)

			66 MHz and 55 MHz		
Num	**Symbol**	**Parameter**	**Min**	**Max**	**Unit**
D1	tBIOVCKIL	BIO Setup Time	3	—	ns
D2	tCKILBIOX	BIO Hold Time	3	—	ns
D3	tBIO0HBIO0H	Timer Clock Period	2T + 10	—	ns
D4	tBIO0HBIO0L	Timer Clock High Time	T + 5	—	ns
D5	tBIO0LBIO0H	Timer Clock Low Time	T + 5	—	ns

Table 17. Timing Characteristics for Bit I/O (See Figure 20.)

			55 MHz		66 MHz		
Num	**Symbol**	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Unit**
D6	tCKILBIOV	BIO Delay	—	15	—	14	ns
D7	tCKILBIOX	BIO Hold	3	—	3	—	ns

Timing Specifications (continued)

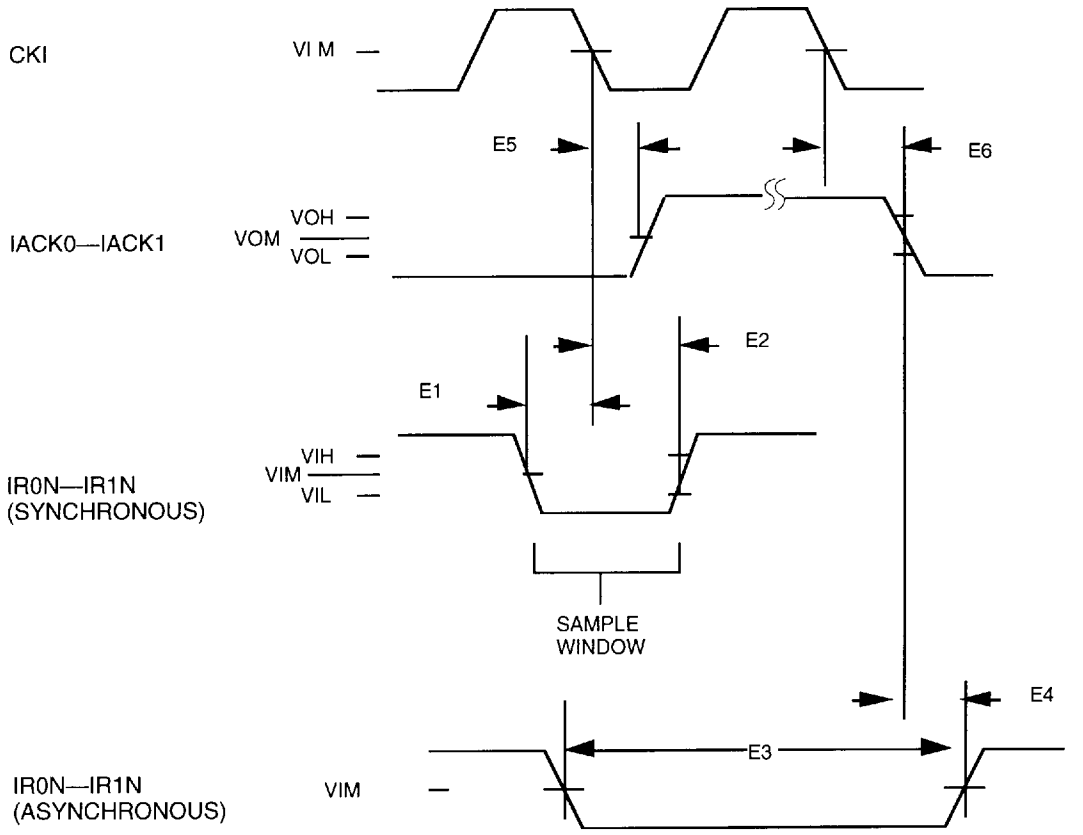


Figure 21. Interrupt Timing

Table 18. Timing Requirements for Interrupts (See Figure 21.)

			55 MHz and 66 MHz		
Num	**Symbol**	**Parameter**	**Min**	**Max**	**Unit**
E1	tIRLCKIL	Interrupt Asserted Setup Time (synchronous)	4	—	ns
E2	tCKILIRX	Interrupt Asserted Hold Time (synchronous)	2	—	ns
E3	tIRLIRH	Interrupt Asserted Pulse Width (asynchronous)	T + 5	—	ns
E4	tIACKLIRH	Interrupt Acknowledge to Interrupt Negation **	—	2T	ns

Table 19. Timing Characteristics for Interrupts (See Figure 21.)

			55 MHz		66 MHz		
Num	**Symbol**	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Unit**
E5	tCKILIACKH	Interrupt Acknowledge Delay	—	15	—	14	ns
E6	tCKILIACKX	Interrupt Acknowledge Hold *	3	—	3	—	ns

* The interrupt acknowledge signal will remain asserted until the ireturn instruction is executed.
** The interrupt request signal must be de-asserted within 2 CKI periods of the interrupt acknowledgement signal's de-assertion, in order to prevent a subsequent extraneous interrupt.

Timing Specifications (continued)

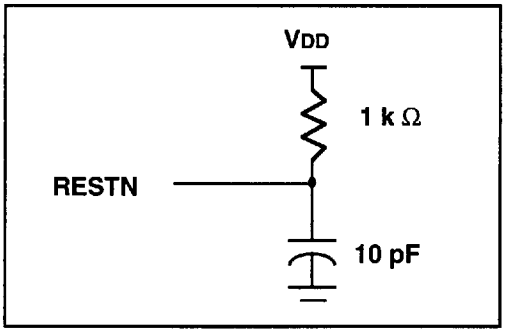


Figure 22. RESTN Test Load Circuit

Table 20. Timing Requirements for Reset (See Figures 22 to 24.)

Num	Symbol	Parameter	Min	Max	Unit
F1	tBIOVRESTH	BIO Setup to RESTN High* (55 MHz)	8	—	ns
F1	tBIOVRESTH	BIO Setup to RESTN High* (66 MHz)	6	—	ns
F2	tRESTHBIOX	BIO Hold after RESTN High	—	0	ns
F3	tRESTHZNH	ZN High after RESTN High	—	8T†	ns
F4	tRESTLRESTH	RESTN In Pulse Width	8T + 10	—	ns
F5	tRESTLRESTH	RESTN In Pulse Width	8T + 10	—	ns

* Only BIO4—BIO7 are latched into the device when RESTN makes a zero-to-one transition.
† ZN may be asserted at any time. This maximum specification permits correct DSP3210 operation in the run mode.

Table 21. Timing Characteristics for Reset and ZN (See Figures 22 to 24.)

				55 MHz	66 MHz	
Num	Symbol	Parameter	Min	Max	Max	Unit
F6	tRESTPU	Power-on RESTN Output Pulse Width	7T – 10	7T + 60	7T + 60	ns
F7	tCKILRESTV	Power-on RESTN Output Delay	—	60	60	ns
F8	tRESTLBIOZ	Power-on RESTN Low to BIO High Z (inputs)	—	9	9	ns
F9	tRESTLBISZ	Power-on RESTN Low to Bus Interface Signals High Z*	—	9	9	ns
F10	tRESTHBISE	RESTN High to Bus Interface Signals Low Z	—	12T + 18	12T + 15	ns
F11	tRESTLSOBZ	Power-on RESTN Low to Output and Biput Signals High Z†	—	9	9	ns
F12	tRESTLSOBE	Power-on RESTN Low to Output and Biput Signals Low Z	—	8T + 12	8T + 12	ns
F13	tZNLSOBZ	ZN Low to Output and Biput Signals High Z	—	12	9	ns
F14	tZNHSOBE	ZN High to Output and Biput Signals Low Z	—	12	9	ns
F15	tRESTILBIOZ	RESTN In Low to BIO High Z (Inputs)	—	12	9	ns
F16	tRESTILBISZ	RESTN In Low to Bus Interface Signals High Z	—	4T + 18	4T + 15	ns

* Bus interface signals are defined to be A2—A31, MS0—MS3, CSN, ASN, RW, PBD, MRN, MWN, and LOCKN.
† Outputs and biputs are defined to be BRN, BGACKN, BIO0—BIO7, ICK, ILD, DO, OCK, OLD, SY, and IACK0—IACK1.

Timing Specifications (continued)

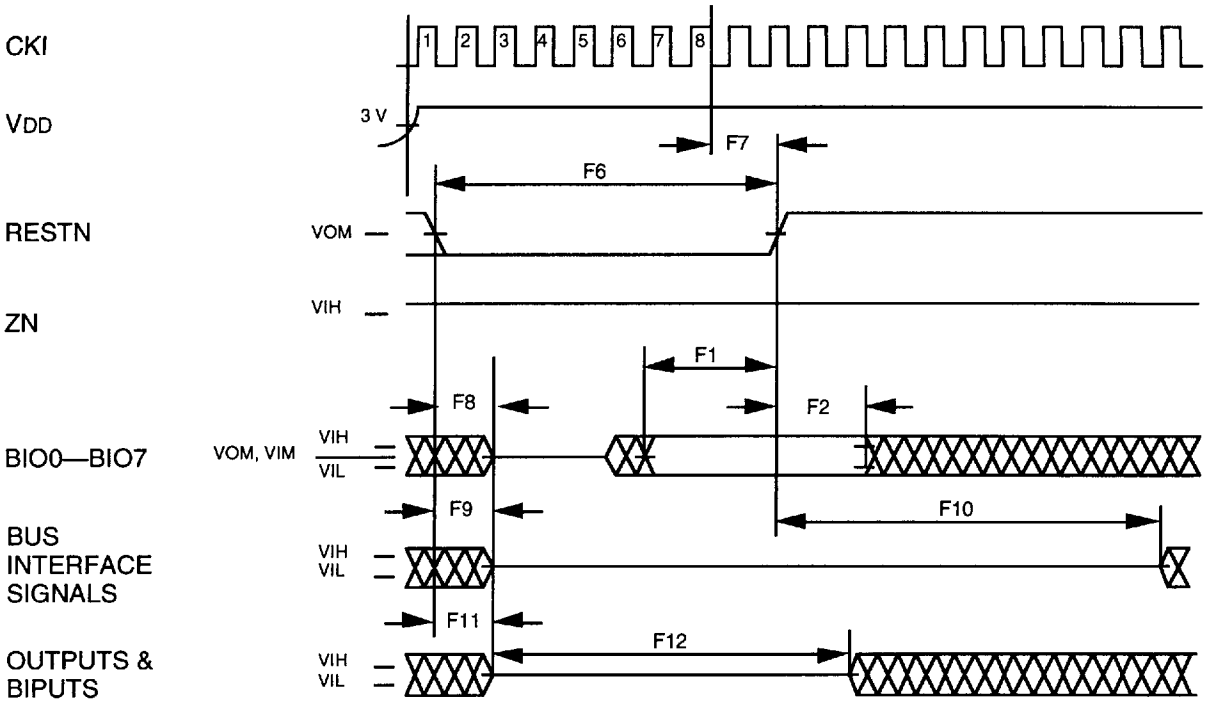


Figure 23. Power-On Reset Timing

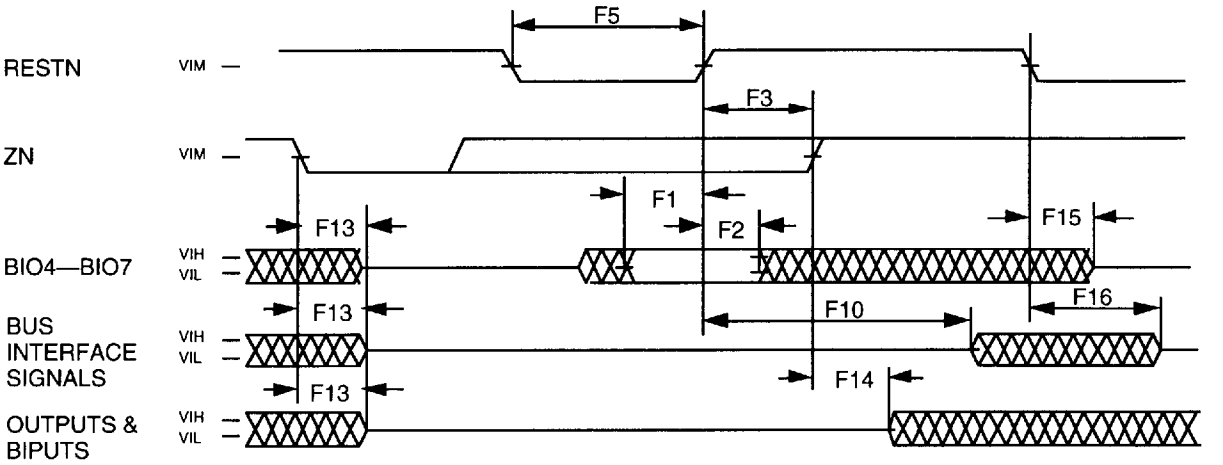


Figure 24. Externally Controlled Reset and ZN Timing

Outline Diagram

132-Pin PQFP

Dimensions are in inches and (millimeters).