Revisions between Versions 3.11 and 3.12

Appendix A, "EISA Bus Electrical Characteristics" is new in Version 3.12.

Version 3.12 also includes additional information in Appendix B, "Configuration File Extensions."

Also, please pay close attention to the changes in section 2.7 "Locked Cycles."

TABLE OF CONTENTS

LIST OF FIGURES

Foreword

Since its inception, the growth of the personal computer market has been driven by the emergence of a de-facto industry standard. The industry standard started with the original IBM PC system architecture and has evolved to the 80386/80486 architecture in use today.

The industry-standard architecture (ISA) provides enormous benefits to the PC user community. It is a stable platform for software and hardware development that gives customers the largest selection of products in the history of computing. ISA compatibility across a wide range of products enables users to adopt new technologies quickly and efficiently, while protecting their investment in expansion boards and software. Availability of a variety of ISA compatible products has freed PC users from a single-vendor, proprietary architecture and given them real freedom of choice to select the best computers, software and peripherals to meet their needs. Over the last seven years, ISA has evolved to a customer-controlled standard rather than a vendor-controlled standard.

Between 10 and 15 million personal computers based on the industry standard architecture are in use today. There are tens of thousands of software products and thousands of expansion boards and peripherals available for ISA compatible PCs. Hundreds of personal computer models are available from dozens of manufacturers that take advantage of the huge base of hardware and software. U.S. business has invested nearly $100 billion in ISA personal computers, software, expansion boards, peripherals and user training.

A steady progression of advances has resulted in performance and function enhancements to the industry standard, while maintaining full compatibility with PC hardware and software products. Microprocessors progressed from the 8088 and 8086 to the 80286, to the 80386, and then to the i486. DOS has evolved to support over a gigabyte of fixed disk storage space and expanded memory manager software has been developed to allow DOS applications access to expanded memory. MS-Windows, OS/2, UNIX, and XENIX and now provide multi-tasking capabilities on the 80286. Expansion bus I/O and memory addressing were increased with the addition of a 16-bit data bus and a 24-bit (16 megabyte) address bus. Each advance was carefully engineered for full compatibility with industry standard hardware and software.

Upon this firmly established foundation, the industry standard will continue to strengthen and evolve. The future will bring even faster 80386 microprocessors and eventually a compatible 80486 microprocessor. It will bring new, compatible versions of operating systems, including advanced versions of DOS and an 80386 version of OS/2.

The combination of the 386 architecture and advanced operating systems will stimulate the development of a new generation of PC applications traditionally associated with departmental computer systems: like advanced networking, communications gateways, database access by multiple users and transaction processing. These multi-user applications require the transfer of large volumes of data and will create the need to extend the ISA data and address bus to 32-bits.

This EISA specification is a joint effort by computer industry leaders to develop the 32-bit extension for industry standard computers. It defines a high-performance, open-architecture bus available to PC manufacturers, expansion board vendors, software developers and semiconductor suppliers without financial or technical constraints.

Notational Conventions

The following notational conventions are used throughout this specification.

Register Notation and Usage

The standard Intel naming conventions are used for the 80386 registers. AX, BX, CX, and DX are the names of the general registers when used as word-length (16-bit). AH, AL, BH, BL, CH, CL, DH, and DL are the names for the general registers when they are used as byte-length registers (8-bit). When addresses are handled, BX usually contains the offset. However, SI (source index) or BP (base pointer) may also be used with the ES (extra segment) register.

Bit Notation

Bit fields within a byte or word are shown as a range of decimal numbers separated by two dots and enclosed in angle brackets, as name <x:y>.

Signal Names

A bus is shown as the bus signal name followed by a range of decimal numbers separated by two dots and enclosed in angle brachets, for example, SA<19:0>.

A slot-specific signal is shown as the signal name followed by a lower case x, for example, AEXx.

Negative true logic is indicated by an asterisk (*) following the signal name, for example, START*.

Radix Notation

Hexadecimal numbers are indicated by a lower case "h" following the digits, for example, 100h.

Bytes, Words, Double Words

A byte is 8 bits. A word is 16 bits. A dword is 32 bits.

Units of Measure

The following units of measure are used throughout this specification.

A	amp	
cm	centimeter	10^{-2} meters
GB	gigabyte	2^{30} bytes
K	kilo-ohm	10^{3} ohms
KB	kilobyte	2^{10} bytes
KHz	kilohertz	10^{3} hertz
MB	megabyte	2^{20} bytes
MHz	megahertz	10^{6} hertz
m	meter	
us	microsecond	10^{-6} sec
mA	milliampere	10^{-3} amps
mm	millimeter	10^{-3} meters
ms	millisecond	10^{-3} sec
ns	nanosecond	10^{-9} sec
pF	picofarad	10^{-12} farads
s	second	
uA	microamps	10^{-6} amps
V	volt	
W	watt	

1. EISA Overview

The Extended Industry Standard Architecture (EISA) is a superset of the ISA 8- and 16-bit architecture. It extends the capabilities of that standard while maintaining compatibility with ISA expansion boards.

EISA introduces the following major advances:

- 32-bit memory addressing for CPU, Direct Memory Access (DMA) devices and bus masters
- 16- or 32-bit data transfers for CPU, DMA and bus master devices
- An efficient synchronous data transfer protocol that allows for normal single transfers as well as high-speed Burst transfers
- Automatic translation of bus cycles between EISA and ISA masters and slaves
- Support of intelligent bus master peripheral controllers
- Enhanced DMA arbitration and transfer rates
- 33 MB/s data transfer rate for bus masters and DMA devices
- Shareable interrupts, programmable for edge or level triggering
- Automatic configuration of system and expansion boards

1.1 Compatibility with ISA

EISA systems maintain full compatibility with the existing industry standard. EISA connectors are a superset of the 16-bit connectors on ISA system boards. ISA 8- and 16-bit expansion boards can be installed in EISA slots. All EISA performance and function enhancements are, similarly, superset features that maintain full compatibility with ISA expansion boards and software.

1.2 Memory Capacity

EISA systems support a 32-bit address path. The main CPU, bus masters and DMA devices can access the entire 80386 memory space. ISA memory cards can be used in the lower 16 megabytes without modification. EISA memory cards can add as much memory as needed for the application. The total memory supported is limited only by the packaging constraints of the individual product, rather than the system architecture.

1.3 Synchronous Data Transfer Protocol

The EISA bus achieves its speed and flexibility through the use of a synchronous transfer protocol. Bus masters and multiple processors can synchronize their bus cycles to a common clock to achieve maximum performance. The synchronous transfer protocol also provides the cycle control necessary to execute Burst cycles with up to 33 MB/s data transfer rate.

On the EISA synchronous bus, control signals, address lines and data bus use a bus clock generated by the system board as the reference for a transfer. Unlike many systems, however, the bus clock is not a fixed frequency. Since the system board is the source of most bus cycles, the system board adjusts the bus clock frequency and phase to achieve the maximum performance of the CPU and memory.

EISA provides a variety of cycle types to cover the range of speed and the complexity requirements for different applications. The standard transfer cycle requires 2 clock cycles, but CPUs are permitted to generate a 1.5 clock COMPRESSED cycle for slaves that request it. At the high end of the performance spectrum are Burst cycles which require 1 clock per transfer.

1.4 Enhanced DMA Functions

EISA systems provide a number of DMA enhancements, including: 32-bit addressability, 8-, 16-, and 32-bit data transfers and higher performance arbitration and data transfer cycles. EISA DMA provides ISA compatible modes, with ISA timing and function as the default.

DMA offers a lower cost alternative to an intelligent bus master. The EISA DMA functions are intended for I/O peripherals that do not require local intelligence on the peripheral interface.

1.4.1 32-bit Address Support for DMA Transfers

EISA 32-bit address support enables ISA, as well as EISA DMA devices to transfer data to any 32-bit memory address. The default DMA supports ISA compatible 24-bit address with no software or hardware modifications. DMA software can be modified to support the 32-bit memory space, without modifications to the DMA hardware.

1.4.2 8-, 16- or 32-bit Data Transfers from DMA Devices

Any DMA channel can be programmed for 8-, 16- or 32-bit data transfers. An 8-bit DMA device uses the low 8 bits of the data bus, a 16-bit device uses the low 16 bits, and a 32-bit device uses the full 32-bit data bus.

A 32-bit DMA device can perform up to 33 MB/s data transfers using Burst cycles.

Performance Gains for DMA Devices

EISA DMA devices can be programmed for high-performance data transfers using one of four DMA cycle types. The default cycle type, Compatible cycles, delivers a higher data transfer rate than ISA compatible computers. The improvement is the result of EISA's faster bus arbitration and requires no hardware or software modifications to ISA compatible DMA devices. Type "A" and Type "B" cycles are EISA modes that, with special programming, allow some ISA compatible DMA devices to achieve even higher performance. The Burst DMA (Type "C") cycle type is the highest performance DMA cycle and is only available to DMA devices designed specifically for Burst.

The following table indicates peak data transfer rates for each DMA cycle type and the DMA devices that are compatible with the cycle type.

DMA Cycle Types

DMA Cycle Type	Transfer Rate (MB/s)	Compatibility
Compatible		
8-bit	1.0	All ISA
16-bit	2.1	All ISA
Type "A"		
8-bit	1.4	Most ISA
16-bit	2.8	Most ISA
32-bit	5.6	EISA Only
Type "B"		
8-bit	2.1	Some ISA
16-bit	4.2	Some ISA
32-bit	8.3	EISA Only
Burst DMA (Type "C")		
8-bit	8.3	EISA Only
16-bit	16.7	EISA Only
32-bit	33.3	EISA Only

1.5 Bus Master Capabilities

EISA based computers support a bus master architecture for intelligent peripherals. The bus master architecture provides a high-speed data channel with data rates up to 33 MB/s using EISA Burst cycles. The bus master provides local intelligence by including a dedicated I/O processor and local memory. It can relieve the burden on the main CPU by performing sophisticated memory access functions, such as non-ordered scatter-gather data transfers. Examples of applications that might benefit from a bus master implementation include communication gateways, disk controllers, LAN interfaces, data acquisition systems, and certain classes of graphics controllers.

1.6 Data Size Translation

The EISA bus system provides a mechanism for EISA expansion boards to communicate with ISA compatible devices. The EISA bus master or slave generates EISA data and control signals, letting the system board copy the data to the appropriate byte lanes and translate the control signals as necessary.

The system board provides the automatic translation for 16-bit ISA bus masters, 8- or 16-bit memory and I/O slaves, and DMA devices. The system board also provides automatic translation for transactions between 16- and 32-bit EISA devices.

1.7 Bus Arbitration

EISA systems also provide a centralized arbitration scheme that allows efficient bus sharing among multiple EISA bus masters and DMA devices. The centralized arbitration supports preemption of an active bus master or DMA device and can reset a device that does not release the bus after preemption.

The EISA arbitration method grants the bus to DMA devices, DRAM refresh, bus masters and CPU functions on a fair, rotational basis. The rotational scheme provides a short latency for DMA devices to assure compatibility with ISA DMA devices. Bus masters and the CPU, which typically have buffering available, have longer, but deterministic latencies.

1.8 Edge/Level Triggered Interrupts

EISA systems provide level-triggered, shareable interrupts. Any EISA interrupt can be individually configured for level- or edge-triggered operation. Edge-triggered operation provides full compatibility with existing, interrupt-driven, ISA devices. Level-triggered operation facilitates the sharing of a single system interrupt by a number of devices. Level-triggered interrupts might be used, for example, to share a single interrupt between a number of serial ports.

1.9 Automatic System Configuration

EISA provides the capabilities for automatic configuration of system and expansion boards. EISA expansion board manufacturers include configuration files with expansion board products. The configuration files can be included with either new, fully programmable EISA boards or switch-configured ISA products. The configuration files are used at system configuration time to assign system resources (such as DMA channels, interrupt levels) and thus prevent conflicts between the installed expansion boards. For switch-configurable boards, the configuration files can be used to outline the proper assignment of resources and instruct the user about the proper selection of switch settings.

To accomplish the automatic system and expansion board configuration, EISA provides a method for accessing I/O port ranges that are slot specific. This means that a board using these ranges can be plugged into any slot in the system without the risk of I/O range conflicts. These I/O ranges can be used for expansion board initialization or for normal I/O port assignments that are guaranteed not to conflict with any other expansion board installed in the system.

EISA also includes a product identification mechanism for systems and expansion board products. The product identifier allows products to be identified during the configuration and initialization sequences for the system and expansion boards. EISA includes guidelines for selection of a product identifier. The identifier of each product is selected by the product manufacturer and does not need the approval of any other party in the industry.

1.10 EISA Feature/Benefit Summary

The following is a summary of the key features and benefits of the extended industry standard architecture.

Feature	Benefit
Full support of industry standard expansion boards	Preserves customer and industry investment. Provides maximum flexibility in product selection.
ISA expansion board size	63 square inches of board space for complex peripherals and ease of implementation.
Maximum +5 V power per slot of approximately 4.5 A	Ample power available for complex, intelligent peripherals.
Full-function 32-bit address and data buses	33 MB/s bus master and DMA data transfer rates for high-performance peripherals.
	Support for greater than 16 MB of memory.
	Programmable level- or edge-triggered interrupts
	Facilitates interrupt sharing by multiple devices.

Feature	Benefit
Enhanced DMA functions	
• Efficient arbitration cycles • Support of demand and block DMA transfers • Fast DMA cycle times • Support of 32-bit address and data size	Improved performance and memory addressing for ISA and EISA DMA devices. Improved efficiency of DMA data block transfers up to rates of 33 MB/s for 32-bit DMA transfers.
Bus master support	
• Support for multiple bus master peripherals • Efficient arbitration cycles • Automatic 32-, 16- or 8-bit data path translation • Support of 32-bit transfers • Support of fast Burst cycles	Provides high performance and local intelligence for sophisticated peripherals. Data transfer rate up to 33 MB/s for 32-bit bus master peripheral. Enhanced ease of configuration for new EISA boards and existing ISA expansion boards.
Automatic expansion board configuration	

2. EISA Bus Specification

2.1 Signal Descriptions

This section describes signals from each connector of the EISA bus.

2.1.1 Address and Data Bus Signal Group

This section describes the bus signals used for memory and I/O addressing and bus signals used for the transfer of data.

BE*<3:0> - (EISA Connector)

BE*<3:0> are the byte enable signals that identify the specific bytes addressed in a dword. BE*<3:0> are pipelined from one cycle to the next and must be latched by the addressed slave if required for the whole cycle. The timing of these signals varies depending on the cycle type. During normal cycles, they go valid before BALE goes active and remain valid as long as the LA<31:2> lines remain valid. During DMA or 16-bit ISA bus master cycles, they go valid at least 1/2 BCLK before the CMD* or ISA command signals go active.

It is permissible for a 32-bit bus master to drive both of the high bytes of the data bus on write cycles even if it only places valid data (as indicated by BE*<3:0> lines) on one of the high bytes.

SBHE* - (ISA Connector)

SBHE* (System Bus High Enable) indicates (when low) that expansion boards that support 16-bit data transfers should drive data on the high half of the D<15:0> data bus. On normal cycles, SBHE* becomes valid on the bus when BALE is asserted and remains valid until after the command (MRDC*, MWTC*, IORC*, IOWC* or CMD*) is negated. On DMA or 16-bit ISA bus master cycles, SBHE* is valid nominally one BCLK before the command signals and remains valid nominally one BCLK after the command signals go away.

AENx - (ISA Connector)

This slot-specific (the "x" refers to the slot number) signal, when negated (low), indicates that an I/O slave may respond to addresses and I/O commands on the bus. AENx is asserted (high) during DMA cycles to prevent I/O slaves from mis-interpreting DMA cycles as valid I/O cycles. The system board must negate AENx when START* is asserted for an I/O access, and AENx must remain negated until after CMD* is asserted. AENx is also used to disable I/O accesses to all other option slots during accesses to a particular slot's slot-specific I/O address range.

2.1.2 Data Transfer Control Signal Group

This section describes the signals used to control data transfer cycles on the 8-, 16- and 32-bit bus.

BCLK - (ISA Connector)

BCLK is provided to synchronize events with the main system clock. BCLK operates at a frequency between 8.333 MHz and ~~6~~ 4 MHz, with a normal duty cycle of 50 percent. BCLK is driven only by the system board. The BCLK period is sometimes extended for synchronization to the main CPU or other system board devices. For example, the COMPRESSED cycle type extends each BCLK period by holding BCLK low for half a cycle beyond the normal transition to high. The BCLK extension facilitates synchronization during the 1.5 BCLK COMPRESSED cycle. During bus master accesses, the system board extends BCLK only when required to synchronize with main memory. Events must be synchronized to BCLK edges without regard to frequency or duty cycle. BCLK is always synchronous with the trailing edge of START* and the leading edge of CMD*. BCLK may not be synchronous with the leading edge of START* or the trailing edge of CMD*.

MSBURST* - (EISA Connector)

Bus master asserts MSBURST* to indicate to the burstable slave that the bus master will execute burst cycles. MSBURST* may only be asserted after SLBURST* is sampled asserted and when no cycle translation is required. A bus master asserts MSBURST* with the LA<31:2> address lines for the second and all subsequent cycles of the burst and the slave samples MSBURST* on the BCLK rising edge. ~~An EISA CPU or bus master asserts MSBURST* to indicate to the slave (typically, main memory) that the CPU or bus master can provide Burst cycles. MSBURST* is asserted with the LA<31:2> address lines for the second and all subsequent cycles of the Burst and is sampled on the rising edge of BCLK by the slave.~~

SLBURST* - (EISA Connector)

A slave (typically, main memory) indicates its support of Burst cycles by asserting SLBURST*. The slave develops SLBURST* from the LA<31:10> address lines and M-IO and produces SLBURST* regardless of the state of MSBURST*. **A memory slave that asserts SLBURST* must sample memory write data on or before a rising BCLK edge with CMD* asserted (regardless of the state of MSBURST*).** SLBURST* is sampled on the rising edge of BCLK by the main CPU, DMA controller or bus master.

M-IO - (EISA Connector)

The main CPU or an EISA bus master asserts M-IO to indicate the type cycle in progress as a memory cycle (high) or I/O cycle (low). M-IO is pipelined from one cycle to the next and is latched by the addressed slave if needed for the whole cycle. M-IO should be included in all decodes by EISA slaves. M-IO must not be used in decoding the signals M16* or IO16*.

LOCK* - (EISA Connector)

The main CPU or a bus master may assert LOCK* to guarantee exclusive memory access during the time LOCK* is asserted. A bus master may also assert LOCK* to guarantee exclusive I/O access during the time LOCK* is asserted. Assertion of LOCK* allows ~~bit~~-test-and-set operations (as used for semaphores) to be executed as a unit, with the bus lock preventing multiple devices from simultaneously modifying the semaphore-~~bit~~.

EX32* - (EISA Connector)

A memory or I/O slave asserts EX32* to indicate that it supports 32-bit (dword) transfers. A two BCLK cycle is executed when a slave asserts EX32* during a memory access. The slave asserts EX32* after decoding a valid address on the LA<31:2> address lines and M-IO. EX32* should not be latched by the slave. Both 16- and 32-bit EISA bus masters must monitor EX32* at the trailing edge of START* to determine if the slave supports 32- (and 16-) bit EISA transfers (asserted), or if the system board is performing data size translation (negated). If data size translation is being done and the master is a 32-bit master, then the system board asserts EX32* to indicate completion of the translation.

EX16* - (EISA Connector)

An EISA memory or I/O slave asserts EX16* to indicate that it supports 16-bit (word) transfers. A 16-bit EISA bus master samples EX16* asserted to confirm a 16-bit EISA slave. An EISA cycle (two BCLK) is executed when a slave asserts EX16* during a memory access by the system board or a 16-bit EISA bus master. The slave asserts EX16* after decoding a valid address on the LA<31:2> address lines and M-IO. EX16* should not be latched by the slave. 16-bit EISA bus masters must monitor EX16* to determine if the slave supports 16-bit EISA transfers (asserted), or if the system board is performing data size translation (negated). If data size translation is being done (ISA cycles) and the master is a 16-bit master (indicated by the master asserting MASTER16*), then the system board asserts EX16* to indicate completion of the translation.

EXRDY - (EISA Connector)

EISA I/O and memory slaves negate EXRDY to request wait state timing (each wait state is one BCLK). The system board samples EXRDY on each falling edge of BCLK after it asserts CMD*. **(On cycles to ISA slaves EXRDY is sampled on each falling edge of BCLK after IORC*, IOWC*, MRDC* and MWTC* are asserted.)** The system board holds CMD* asserted during the entire period EXRDY is negated, and at least one half BCLK after sampling EXRDY asserted. EXRDY must be driven with an open-collector type buffer (a system board pull up resistor provides the asserting drive current). The EISA slave should negate EXRDY during START* or on the rising edge of BCLK at the end of START* if wait states are to be added. The slave *must* allow EXRDY to float high (asserted) synchronously with BCLK falling edge and must not hold EXRDY negated ~~asserted~~ longer than 2.5 μs. EXRDY should *never* be driven high.

START* - (EISA Connector)

The START* signal provides timing control at the start of a cycle. The CPU or bus master asserts START* after LA<31:2> and M-IO become valid and negates START* on a rising edge of BCLK after one BCLK cycle time. BE*<3:0> and W-R may not be valid at the leading edge of START*.

CMD* - (EISA Connector)

CMD* provides timing control within the cycle. The system board asserts CMD* on the rising edge of BCLK, simultaneously with negation of START*. The system board holds CMD* asserted until the end of the cycle. The end of the cycle normally is synchronized with the rising edge of BCLK, but in certain cases is asynchronous. A bus master does not drive CMD*.

W-R - (EISA Connector)

The status signal, W-R, identifies the cycle as a write (high) or read (low). W-R becomes valid after assertion of START* and before assertion of CMD*. W-R remains valid as long as address lines LA<31:2> are valid. W-R is driven from the same edge of BCLK that activates the START* signal.

BALE - (ISA Connector)

BALE (when high) indicates that a valid address is present on the LA<31:2> address lines. The LA<31:2> address lines or any decodes developed from them by ISA devices are latched (with transparent latches) on the trailing edge of BALE if the address is needed for the whole cycle. BALE is always high during a DMA or 16-bit ISA bus master operation. EISA devices should not use BALE to latch addresses; the trailing edge of START* or leading edge of CMD* should be used.

MRDC* - (ISA Connector)

The system board or ISA bus master asserts MRDC* to indicate that the addressed ISA memory slave should drive its data onto the memory bus. MRDC* is asserted for read accesses to memory, except when inhibited by assertion of EX32* or EX16* (an EISA device responded). During ISA Compatible DMA cycles, MRDC* is asserted for read accesses to memory addresses between 00000000h to 00FFFFFFh, regardless of the type of memory responding. A DMA device should not use MRDC* to decode its I/O address. MRDC* is also asserted for refresh cycles. MRDC* can be driven by an expansion board acting as an ISA 16-bit bus master.

MWTC* - (ISA Connector)

The system board or ISA bus master asserts MWTC* to indicate that the addressed ISA memory slave may latch data from the memory bus. MWTC* is asserted for write accesses to memory, except when inhibited by assertion of EX32* or EX16* (an EISA device responded). During Compatible DMA cycles, MWTC* is asserted for write accesses to memory addresses between 00000000h to 00FFFFFFh, regardless of the type of memory responding. A DMA device should not use MWTC* to decode its I/O address. MWTC* can be driven by an expansion board acting as an ISA 16-bit bus master.

SMWTC* - (ISA Connector)

The system board asserts SMWTC* to indicate that the addressed memory slave may latch data from the memory bus. SMWTC* is only asserted for ISA write accesses to memory addresses between 00000000h to 000FFFFFh. SMWTC* is derived from MWTC* and has similar timing.

SMRDC* - (ISA Connector)

The system board asserts SMRDC* to indicate that the addressed memory slave should drive its data onto the memory bus. SMRDC* is only asserted for ISA read accesses to memory addresses between 00000000h to 000FFFFFh or refresh cycles. SMRDC* is derived from MRDC* and has similar timing.

IOWC* - (ISA Connector)

The system board or ISA bus master asserts IOWC* to indicate that the addressed ISA I/O slave may latch data from the EISA bus. ~~A DMA device can latch data from the data bus when IOWC* is asserted.~~ **IOWC* is asserted for I/O write accesses, except when inhibited by assertion of EX32* or EX16* (an EISA device responded).** An ISA I/O slave latches data from the data bus when IOWC* is asserted and AENx is negated. The main CPU or bus master must drive valid data on the bus before asserting IOWC*. **A DMA device can latch data from the data bus when IOWC* is asserted.**

IORC* - (ISA Connector)

The system board or ISA bus master asserts IORC* to indicate that the addressed ISA I/O slave should drive its data onto the EISA bus. ~~A DMA device can drive data on the data bus after sampling IORC* asserted.~~ IORC* is asserted for I/O read accesses, except when inhibited by assertion of EX32* or EX16* (an EISA device responded). An ISA I/O slave drives data onto the bus while IORC* is asserted and AENx is negated (low). The device must hold the data valid until sampling IORC* negated. A DMA device can drive data on the data bus after sampling IORC* asserted.

CHRDY - (ISA Connector)

An ISA memory or I/O slave can negate CHRDY to lengthen a bus cycle from the default time. The slave negates CHRDY after decoding a valid address and sampling the command signal (MRDC*, MWTC*, SMRDC*, SMWTC*, IORC* or IOWC*) asserted. When the slave's access has completed, CHRDY should be allowed to float high (asserted). Bus cycles are lengthened by an integral number of BCLK cycles. The ISA command signals remain active at least one BCLK after the slave asserts CHRDY. CHRDY should be driven with an open collector type of driver, and should *never* be driven high. CHRDY may not be held low for more than ~~2.1~~ 2.5 μs. EISA slaves should never negate CHRDY.

NOWS* - (ISA Connector)

An ISA memory slave asserts NOWS* (No Wait State) after its address and a command have been decoded to indicate that the remaining clock cycles are not required. NOWS* must be asserted before the falling edge of BCLK to be recognized during ISA cycles. During EISA cycles, an addressed EISA slave may assert NOWS* before the main CPU negates START* to generate COMPRESSED cycles (1.5 BCLKs/cycle). A slave should not assert NOWS* and negate EXRDY or CHRDY during the same cycle.

M16* - (ISA Connector)

M16* signals the system that the addressed ISA memory is capable of transferring 16 bits of data at once. When M16* is asserted, during a memory read or write and is not superseded by EX32* or EX16*, the ISA compatible three BCLK memory cycle is run. M16* is decoded from LA<23:17>. M-IO is not included in the decode and M16* should not be latched by the ISA slave. Only ISA memory slaves need to generate M16*; the system board generates M16* from EX32* or EX16* for EISA memory slaves. M16* should only be driven with an open-collector type of driver.

IO16* - (ISA Connector)

A 16-bit ISA I/O slave asserts IO16* (after decoding a valid address on SA<15:1>) to indicate its 16-bit data size. The system board defaults to a three BCLK I/O cycle when it samples IO16* asserted by an ISA I/O slave (EX32* and EX16* negated). IO16* should only be driven with an open-collector type of driver.

The system board does not automatically assert IO16* when a 16-bit ISA bus master accesses an EISA I/O slave. EISA slaves that support 16-bit ISA bus masters must assert IO16* as well as EX32* (or EX16*) when addressed. **The 16- or 32-bit EISA I/O slave should decode the LA<11:2> address with AENx low (do not include M-IO), and latch the result in a transparent latch which is open when BALE is high. The output from the latch is used to control the assertion of IO16*.** ~~The EISA I/O slave asserts IO16* on decoding a valid address on LA<15:2>.~~ EISA I/O slaves that do not support 16-bit ISA bus masters need not assert IO16*.

2.1.3 Bus Arbitration Signal Group

This section describes signals used to arbitrate for bus control. These signals are a combination of new EISA signals and existing ISA signals.

MREQx* - (EISA Connector)

MREQx* is a slot-specific signal used by EISA bus masters to request bus access. The "x" refers to the slot number. Bus masters requiring use of the bus must assert MREQx* until the system board grants bus access by asserting MAKx*. The requesting device must hold MREQx* asserted until the system board asserts the appropriate MAKx* signal. The system board samples MREQx* on the rising edge of BCLK. If MREQx* is sampled asserted, the arbitration controller performs the arbitration and the system board asserts MAKx* when the bus becomes available. The bus master can begin driving the bus with address and other signals on the falling edge of BCLK when MAKx* is sampled asserted.

When a bus master completes a transfer, it can release the bus by negating MREQx* on the falling edge of BCLK. If no bus cycle is in progress when MREQx* is negated, the bus master must float LA<31:2>, BE*<3:0>, MSBURST*, LOCK*, D<31:0>, START*, M-IO, and W-R on or before the rising edge of BCLK after MREQx* is negated. If a cycle is in progress when MREQx* is negated, then the LA<31:2>, BE*<3:0>, MSBURST*, LOCK*, START*, M-IO, and W-R signals must be floated by the rising edge of BCLK at the end of the cycle. The data signals D<31:0> must be floated on (EXRDY termination) or before (EX32* or EX16* termination) the falling edge of BCLK after the end of the cycle. Cycle completion is indicated by the memory or I/O slave asserting EXRDY or the system board asserting EX16* or EX32* after completing bus conversions. A bus master must wait at least two BCLKs after releasing the bus before re-asserting its MREQx*. The trailing edge of MREQx* must meet the setup and hold time to the sampling point for proper system operation.

An EISA bus may provide EISA slots that do not support an EISA bus master (see section 4.7.3, BUSMASTER statement). The non-bus master EISA slot must hold MAKx* permanently negated and must not connect any signal to the MREQx* connector pin.

MAKx* - (EISA Connector)

MAKx* is a slot-specific signal that is asserted by the system board to grant bus access to an EISA bus master. The "x" refers to the slot number. MAKx* is asserted from the rising edge of BCLK and the bus master can begin driving LA<31:2>, BE*<3:0>, MSBURST*, START*, M-IO, and W-R on the next falling edge of BCLK. The system board negates MAKx* on the rising edge of BCLK after sampling MREQx* negated. The system board can also negate MAKx* to indicate to an active bus master that another device has requested the bus. The bus master must negate MREQx* to release the bus within 64 BCLKs (8 μs) of sampling MAKx* negated.

EISA system slots do not have to be capable of supporting an EISA bus master adapter (see section 4.7.3, BUSMASTER Statement). However, EISA slots that do not support EISA bus master adapters should have MAKx permanently in the inactive state (+5 volts) and should not have MREQx* connected.

DRQ<7:5>, DRQ<3:0> - (ISA Connector)

The DRQ<x> lines are used to request a DMA service from the DMA subsystem or for a 16-bit ISA bus master to request access to the system bus. The request is made when DRQ<x> is asserted. The system board allows DRQ<x> to be asserted asynchronously. The requesting device must hold DRQ<x> asserted until the system board asserts the appropriate DAK*<x> signal. For demand mode DMA memory-read I/O-write cycles, DRQ<x> is sampled on the rising edge of BCLK, one BCLK from the end of the cycle (the rising edge of IOWC*). For demand mode DMA memory-write I/O-read cycles, DRQ<x> is sampled on the rising edge of BCLK, 1.5 BCLKs from the end of the cycle (the rising edge of IORC*). For demand mode Burst DMA, DRQ<x> is sampled each cycle on the rising edge of BCLK. For 16-bit ISA bus masters, DRQ<x> is sampled on the rising edge of BCLK, two BCLKs before the system board negates DAK*<x>. The trailing edge of DRQ<x> must meet the setup and hold time to the sampling point for proper system operation.

DAK*<7:5>, DAK*<3:0> - (ISA Connector)

The system board asserts a DMA channel's DAK*<x> to indicate that the channel has been granted the bus. A DMA device is selected if it decodes DAK*<x> with IORC* or IOWC* asserted. DAK*<x> can also be used to acknowledge grant of bus access to a 16-bit ISA bus master. The bus master must assert MASTER16* after sampling DAK*<x> asserted. Address and cycle control signals must be floated and MASTER16* must be negated before the system board negates DAK*<x>. For EISA block or demand mode DMA transfers, DAK*<x> remains asserted until the transfer completes or until the centralized arbitration controller preempts the DMA process. The preemption occurs after another device requests the bus and 4 μs elapse.

T-C - (ISA Connector)

This signal is bidirectional, acting in one of two modes, depending on the programming of the channel. In the output mode, the system board asserts T-C to indicate that a DMA channel's word count has reached terminal count. Terminal count is indicated when the decrementing word count "rolls over" from zero to FFFFFFh. The system board asserts T-C only while asserting the channel's DAK*<x>. A DMA device decodes T-C with the appropriate DAK*<x> asserted to determine when the transfer has completed.

In the input mode, T-C can be used by a DMA slave to stop a DMA transfer. During ISA Compatible, Type "A", or Type "B", transfers, T-C is sampled by the system while IORC* or IOWC* is asserted. During Burst cycles, T-C is sampled at the same time as the DRQ<x> input, on the rising edge of BCLK. If it is sampled asserted the transfer is terminated, and if auto-initialize is programmed, the transfer restarts at the beginning.

MASTER16* - (ISA Connector)

A bus master asserts MASTER16* to indicate 16-bit data size. A bus master can assert MASTER16* after the system board asserts DAK*<x> or MAKx*. The 16-bit EISA bus master negates MASTER16* after completing the last transfer. An ISA master negates MASTER16*, immediately when the system board negates DAK*<x>. A 32-bit bus master can assert MASTER16* during START* to disable automatic 32-to-16-bit data size translation for 16-bit EISA memory Burst slaves. It can then perform 16-bit Burst cycles to a 16-bit EISA slave.

REFRESH* - (ISA Connector)

REFRESH* is used to indicate (when low) a refresh cycle in progress. REFRESH* causes SA<15:0> (or LA<15:2>) to drive the row address inputs of all DRAM banks so that when MRDC* (or CMD*) is asserted, the entire system memory is refreshed at one time.

2.1.4 Utility Signal Group

This section describes a variety of general utility signals. These signals are all on the ISA connector.

OSC - (ISA Connector)

OSC is a clock for use in timing applications. Its frequency is 14.31818 MHz with a 50 percent duty cycle.

RESDRV - (ISA Connector)

Assertion of RESDRV causes a hardware reset of ISA and EISA expansion boards. RESDRV is asserted by the reset controller during power up or after a bus timeout. Software can cause assertion of RESDRV by setting I/O port 0461h bit 0 to a "1". RESDRV is negated when the software resets this bit to a zero. RESDRV has a minimum pulse width equivalent to 9 BCLK periods ~~(the minimum time between two ISA I/O write cycles)~~. All devices that can prevent operation of the CPU, memory or system board I/O must use RESDRV for hardware reset. **When RESDRV is asserted, these devices must float all 3-state outputs which drive the EISA bus. These outputs are not permitted to drive the EISA bus after RESDRV has been asserted until the device is initialized.** Slaves that insert wait states based on internal state machines, devices that require software initialization, and DMA devices are examples of hardware that reset after sampling RESDRV asserted.

IRQ<15:14>, IRQ<12:9>, IRQ<7:3> - (ISA Connector)

The IRQx lines are used to interrupt the CPU to request some service. In compatible mode, the interrupt is recognized when IRQx goes from a low to a high and remains there until the appropriate interrupt service routine is executed. If programmed to level-sensitive mode, the interrupt is recognized when the IRQx signal is asserted (low). Another interrupt is generated at the end of the interrupt service routine if the IRQx signal is still held low, allowing a single line to be shared by more than one device. IRQ<15:3> are pulled up by the system board. A floated interrupt line is guaranteed to stabilize at a TTL "high" after 500 ns. Interrupt service routines must reset the interrupt latch (which floats the interrupt line), then wait at least 500 ns before issuing the end-of-interrupt command and enabling interrupts.

IOCHK* - (ISA Connector)

An EISA or ISA expansion board can assert IOCHK* to signal the main CPU that a serious error has occurred. Assertion of IOCHK* causes an NMI if Port 061h bit 3 is ~~set to "1"~~ a "0" and NMIs are enabled. Parity errors and uncorrectable system errors exemplify problems that might cause an expansion board to assert IOCHK*.

2.1.5 Summary of Signals

The following tabulation shows the EISA bus connector signals added for EISA support:

Bus Pins	Signal Name	Description
16	D<31:16>	data lines
8	LA*<31:24>	address lines
15	LA<16:2>	address lines
4	BE*<3:0>	byte enables
1	LOCK*	bus lock
1	EX32*	32-bit EISA slave indicator
1	EX16*	16-bit EISA slave indicator
1	START*	EISA start of cycle control
1	CMD*	EISA end of cycle control
1	M-IO	EISA memory or I/O indicator
1	W-R	EISA write or read indicator
1	EXRDY	EISA ready indicator
1	MREQx*	slot specific bus request
1	MAKx*	slot specific bus grant
1	SLBURST*	Burst cycle indicator from slave
1	MSBURST*	Burst cycle control from master

55		Total new pins on EISA connector

2.1.6 Signal Usage by System, Masters and Slaves

The following three tables indicate typical signal usage by an EISA system board, ISA bus masters, ISA slaves, EISA bus masters and EISA slaves.

Table Legend:

I/O = Input and Output
I = Input
O = Output
- = Signal Not Needed

Subscript "$_m$" indicates that one or more of the signals in the group may be implemented.

An I/O shown in parentheses () indicates that the signal is optional for this device.

The following notes are referenced in one signal usage tables:

1. SLBURST and MSBURST are implemented together or both are omitted.
2. Only DMA devices that implement Burst cycles use EXRDY.
3. EISA DMA devices can be 8-, 16-, or 32-bits wide.
4. ISA DMA device can be either 8- or 16-bits wide.
5. DMA devices need not monitor BE* <3:0> unless they support partial-width data transfers.
6. Only EISA slaves that support COMPRESSED cycles assert NOWS*.
7. EISA I/O slaves that need to be accessed by 16-bit ISA bus masters must **decode LA<11:2> and AEN, and latch the result in a transparent latch which is open when BALE is high. The output from the latch is used to control the assertion of IO16*.** ~~assert IO16* when addressed.~~
8. An 8-bit memory slave is assumed to only decode the SA<19:0> address lines (1 megabyte maximum address). If a full decode is done, LA<23:17>, MRDC*, IORC*, and BALE are also used.
9. BCLK is only required if the slave device supports Burst cycles or uses EXRDY.
10. A 16-bit EISA bus master that does not drive the full 32-bit address will be limited to 16 megabyte addressing.
11. A 32-bit EISA bus Burst master that can "downshift" to a 16-bit EISA Burst memory slave asserts MASTER16* during START*.

EISA/ISA Signal Usage - System Board +

Signal Name	System Board	Signal Name	System Board
AENx	O	M-IO	I/O
BALE	O	M16*	I/O
BCLK	O	MAKx*	O
BE*<3:0>	I/O	MASTER16*	I
CHRDY	I/O	MRDC*	I/O
CMD*	O	MREQx*	I
D<31:0>	I/O	MSBURST*	I/O
DAK*<7:0>	O	MWTC*	I/O
DRQ<7:0>	I	NOWS*	I
EX16*	I/O	OSC	O
EX32*	I/O	REFRESH*	I/O
EXRDY	I/O	RESDRV	O
IO16*	I	SA<19:0>	I/O
IOCHK*	I	SBHE*	I/O
IORC*	I/O	SLBURST*	I
IOWC*	I/O	SMRDC*	O
IRQ<15:3>	I	SMWTC*	O
LA*<31:24>	I/O	START*	I/O
LA<23:2>	I/O	T-C	I/O
LOCK*	O	W-R	I/O

+ The signals listed are required to support EISA functions. Additional signals are required if the system board also contains EISA or ISA slaves.

ISA Signal Usage - ISA Expansion Boards

Signal Name	ISA Bus Master	ISA 16-bit Mem Slave	ISA 16-bit I/O Slave	ISA 8-bit Mem Slave	ISA 8-bit I/O Slave	ISA DMA Device	Notes
AENx	-	-	I	-	I	-	
BALE	-	I	-	(I)	-	-	8
BCLK	(I)	(I)	(I)	(I)	(I)	(I)	
CHRDY	I	(O)	(O)	(O)	(O)	-	
D<7:0>	I/O	I/O	I/O	I/O	I/O	I/O	4
D<15:8>	I/O	I/O	I/O	-	-	(I/O)	4
DAK*<7:0>	I_m	-	-	-	-	I_m	
DRQ<7:0>	O_m	-	-	-	-	O_m	
IO16*	I	-	O	-	-	-	
IOCHK*	(O)	(O)	(O)	(O)	(O)	(O)	
IORC*	O	-	I	-	I	I	
IOWC*	O	-	I	-	I	I	
IRQ<15:3>	(O_m)	(O_m)	(O_m)	(O_m)	(O_m)	(O_m)	
LA<23:17>	O	I	-	(I)	-	-	8
M16*	I	O	-	-	-	-	
MASTER16*	O	-	-	-	-	-	
MRDC*	O	I	-	(I)	-	-	8
MWTC*	O	I	-	(I)	-	-	8
NOWS*	-	(O)	-	(O)	(O)	-	
OSC	(I)	(I)	(I)	(I)	(I)	(I)	
REFRESH*	(O)	I	-	I	-	-	
RESDRV	I	I	I	I	I	I	
SA<16:0>	O	I	I	I	I	-	
SA<19:17>	-	(I)	-	(I)	-	-	8
SBHE*	O	I	I	-	-	-	
SMRDC*	-	-	-	I	-	-	
SMWTC*	-	-	-	I	-	-	
T-C	-	-	-	-	-	(I)	

EISA connector signals are not used by ISA expansion boards and are not included in the preceding table.

EISA/ISA Signal Usage - EISA Expansion Boards

Signal Name	32-bit EISA Bus Master	32-bit EISA Mem Slave	32-bit EISA I/O Slave	16-bit EISA Bus Master	16-bit EISA Mem Slave	16-bit EISA I/O Slave	EISA DMA Device	Notes
AENx	-	-	I	-	-	I	-	
BALE	-	-	**(I)**	-	-	**(I)**	-	**7**
BCLK	I	(I)	(I)	I	(I)	(I)	(I)	9
BE*<3:0>	O	I	I	O	I	I	(I)	5
CMD*	-	I	I	-	I	I	-	
D<7:0>	I/O	I/O	I/O	I/O	I/O	I/O	I/O	
D<15:8>	I/O	I/O	I/O	I/O	I/O	I/O	(I/O)	3
D<31:16>	I/O	I/O	I/O	-	-	-	(I/O)	3
DAK*<7:0>	-	-	-	-	-	-	I_m	
DRQ<7:0>	-	-	-	-	-	-	O_m	
EX16*	-	-	-	I	O	O	-	
EX32*	I	O	O	I	-	-	-	
EXRDY	I	(O)	(O)	I	(O)	(O)	I	2
IO16*	-	-	(O)	-	-	(O)	-	7
IOCHK*	(O)	(O)	(O)	(O)	(O)	(O)	(O)	
IORC*	-	-	-	-	-	-	I	
IOWC*	-	-	-	-	-	-	I	
IRQ<15:3>	(O_m)	(O_m)	(O_m)	(O_m)	(O_m)	(O_m)	(O_m)	
LA<15:2>	O	I	I	O	I	I	-	
LA<23:16>	O	I	-	O	I	-	-	
LA*<31:24>	O	I	-	(O)	I	-	-	10
LOCK*	(O)	(I)	(I)	(O)	(I)	(I)	-	
M-IO	O	I	I	O	I	I	-	
MAKx*	I	-	-	I	-	-	-	
MASTER16*	(O)	-	-	O	-	-	-	11
MREQx*	O	-	-	O	-	-	-	
MSBURST*	(O)	(I)	-	(O)	(I)	-	-	1
NOWS*	-	(O)	(O)	-	(O)	(O)	-	6
OSC	(I)	(I)	(I)	(I)	(I)	(I)	(I)	
REFRESH*	-	I	-	-	I	-	-	
RESDRV	I	I	I	I	I	I	I	
SLBURST*	(I)	(O)	-	(I)	(O)	-	-	1
START*	O	I	I	O	I	I	-	
T-C	-	-	-	-	-	-	(I/O)	
W-R	O	I	I	O	I	I	-	

Many ISA signals are not used by EISA expansion boards and are not included in the preceding table.

2.2 ISA Devices~~Cycles~~

2.2.1 CPU Cycles

ISA systems provide different timing for cycles to and from 8- and 16-bit memory and I/O slaves. ISA systems generate a default 6 BCLK memory or I/O cycle for 8-bit slaves and a default 3 BCLK memory or I/O cycle for 16-bit slaves. All cycles can be extended by the slave by negating CHRDY. Additionally, memory or I/O slaves can shorten most cycles (except 16-bit I/O cycles) by asserting NOWS*. **A slave may not negate CHRDY and assert NOWS* in the same cycle. A system board that samples CHRDY negated and NOWS* asserted should honor CHRDY.** ~~If both CHRDY is negated and NOWS* is asserted, then wait states will be added.~~

ISA cycles begin with the system presenting a valid address on LA<23:17>, and one BCLK period later, asserting BALE and presenting a valid SA<19:0> address.

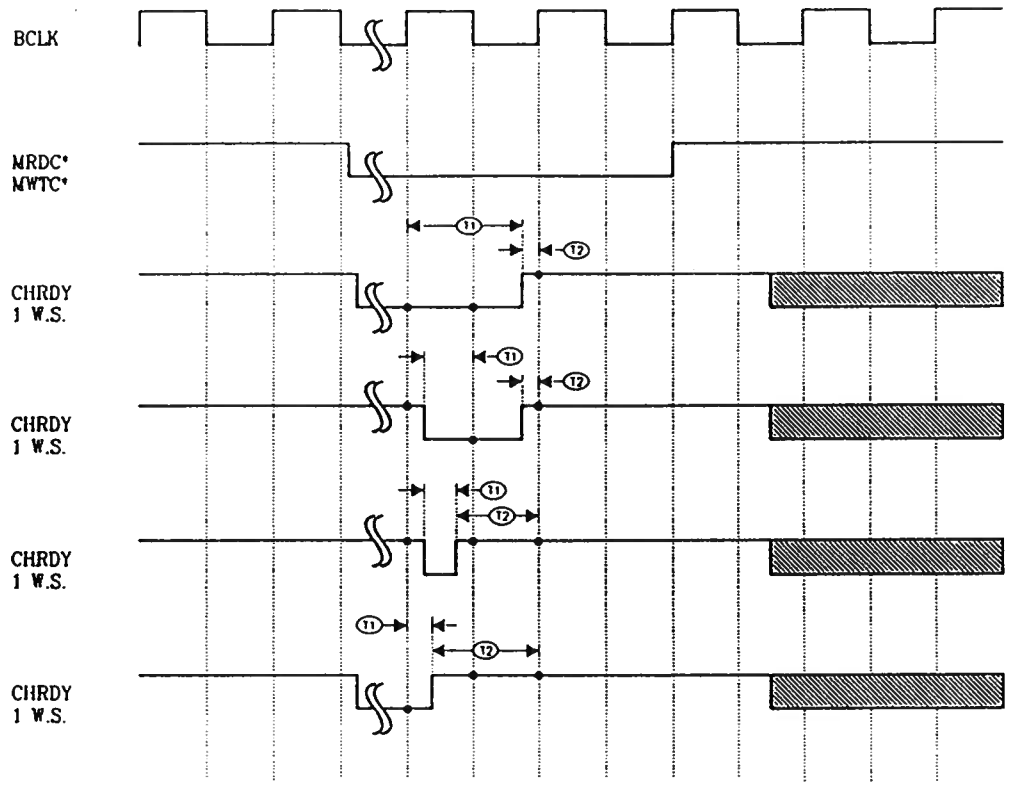
For 16-bit memory accesses, the system asserts MRDC*, MWTC*, SMRDC*, or SMWTC* on the first rising BCLK edge after SA<19:0> become valid. For 8-bit memory accesses, and for all I/O accesses, the system delays an extra one-half BCLK period before asserting the ISA command signal to allow extra time for address decode.

During write cycles, the system presents valid data on the first rising BCLK edge after SA<19:0> become valid. The slave can latch the data after the specified data valid delay or on the trailing edge of the ISA command signal. During read cycles, a slave presenting valid data, drives the data bus after receiving the ISA command signal. The system latches the read data on the edge of BCLK on which the ISA command signal is negated.

NOWS* is sampled on each falling edge of BCLK during the time that the ISA command signal is asserted. This allows 8-bit slaves to shorten a standard 6 BCLK cycle to a 3, 4 or 5 BCLK cycle. A 16-bit memory slave can shorten a standard 3 BCLK cycle to a 2 BCLK cycle. A 16-bit I/O slave cannot shorten cycles, since the ISA command signal is delayed one-half BCLK period; therefore, NOWS* cannot be generated early enough to shorten the cycle.

Systems built according to the EISA specification implement a sampling window for CHRDY, instead of a distinct sample point. To guarantee the insertion of one wait state, CHRDY must be held negated for a minimum time period while BCLK is high. If CHRDY is negated before the rising edge of BCLK, it must be held for the specified hold time past the rising edge. If CHRDY is negated after the rising edge of BCLK, then it must be held negated for a specified pulse width. In either case, CHRDY may then be re-asserted with setup to the next rising BCLK edge. Negation and assertion of CHRDY must meet the pulse width, setup and hold time requirements specified in the ISA signal timing parameter table.

Figure 1 - CHRDY "Sample Window"



NOTES:

1. T1 = CHRDY negated hold time.
2. T2 = CHRDY asserted setup to BCLK rising edge.
3. See ISA Bus Timing Parameters for specific timing values.

The CPU or master can extend the length of the cycle beyond the minimum requirements indicated by the slave by keeping the ISA command signals asserted. Both memory and I/O slaves are required to extend the end of the cycle until the ISA command signals are negated.

The system indicates the size of a memory or I/O transfer being attempted by using SBHE* and SA<0>. The following table shows the size of transfer for each combination and which byte lane contains the data. Byte lanes not included in this table must not be driven by the slaves during read cycles, and must be left unmodified during write cycles.

SBHE*	SA0	SIZE	BYTE LANES
0	0	2	D<15:8>, D<7:0>
0	1	1	D<15:8>
1	0	1	D<7:0>
1	1	0	**Reserved for future revisions of the EISA spec**

2.2.2 Memory Slaves

Memory slaves can be either 8- or 16-bits wide. An 8-bit memory slave can use either 20 address bits (SA<19:0>) or 24 address bits (LA<23:17>, SA<19:0>). When using 20 address bits, the 8-bit slave must use SMRDC* and SMWTC* to guarantee that only cycles to the first 1 MB of memory will be performed. A 16-bit memory slave must use 24 address bits and normally uses MRDC* and MWTC*.

A 16-bit memory slave asserts M16* after decoding LA<23:17>. The decode for M16* must not include SA<19:0>, SBHE*, or any other control signals, since the timing requirements for M16* cannot be assured if control signals are included.

Memory slaves can shorten default cycles by asserting NOWS*, or extend them by negating CHRDY. However, the slave cannot control the maximum length of any cycle, and is required to extend the length of write cycles and to hold read data valid on the bus until the ISA command signals are negated.

2.2.3 I/O Slaves

I/O slaves can be either 8- or 16-bit wide. I/O slaves decode addresses SA<9:0> and AENx. A 16-bit I/O slave asserts IO16* when it decodes a valid address with AENx low. The decode for IO16* should not include any control signals.

I/O slaves can shorten default 8-bit cycles by asserting NOWS*, or extend 8- or 16-bit cycles by negating CHRDY. However, the slave cannot control the maximum length of any cycle, and is required to extend the length of write cycles and to hold read data valid on the bus until the ISA command signals are negated.

2.2.4 Bus Masters

The ISA bus master device driver programs a DMA channel for cascade mode. The ISA bus master asserts DRQ<x> for that channel to request control of the bus. The system board performs the bus arbitration and asserts DAK*<x>, granting control of the bus to the 16-bit ISA bus master and disabling the system board address, data, and control lines. The system board does not assert AENx during DAK*<x> to disable I/O accesses. Consequently, an ISA bus master can perform normal I/O and slot-specific I/O accesses. BALE is asserted with DAK*<x> to indicate valid address on the LA<31:2> bus.

An ISA bus master asserts MASTER16*, but this line is ignored in EISA systems. The ISA bus master then waits at least one BCLK before driving address, data, and control lines to allow the system board to float its drivers. An ISA bus master presents LA<23:17> and SA<19:0>, driving the same address on LA<19:17> and SA<19:17>. ISA bus masters cannot pipeline addresses since the system board holds BALE asserted while the ISA bus master drives the bus.

EISA does not assume that ISA masters are synchronized to BCLK. The EISA system board assumes that they are asynchronous. However, ISA masters should synchronize control signals to BCLK if they are required to be compatible with ISA slaves designed prior to the EISA specification which generate wait states synchronous with BCLK.

EISA requires that all ISA masters monitor CHRDY and add wait states when CHRDY is negated. An ISA master may optionally use NOWS* to shorten default cycles. If both NOWS* is asserted and CHRDY is negated, then the ISA master must insert wait states.

If an ISA master must run refresh cycles without releasing the bus, then it floats the address buses and command lines and asserts REFRESH* with an open collector type driver. The master must then wait for 1 BCLK period after MRDC* has been asserted and negated before floating REFRESH* and driving the address and command buses. EISA systems require ISA masters to wait for the end of MRDC* before regaining the bus during refresh cycles, if proper operation is to be assured.

An ISA bus master releases the bus by floating its address, data, and control signals, negating DRQ<x> and floating MASTER16*. The system board samples DRQ<x> negated on the rising edge of BCLK. The system board negates DAK*<x> on the third rising edge of BCLK after sampling DRQ<x> negated. The ISA bus master negates (floats) MASTER16* (if still asserted) when it samples DAK*<x> negated. On the next BCLK the system board asserts the bus grant signal for the device that wins the bus arbitration.

ISA bus masters use the same combinations of SBHE* and SA<0> as indicated for CPU cycles to indicate the size of the transfer and the location of the data. It is the bus master's responsibility to convert 16-bit transfers into two 8-bit transfers if a 16-bit slave does not respond. However, the system board will provide data copying from D<7:0> to D<15:8> for odd-address reads from a byte slave, and from D<15:8> to D<7:0> for odd-address writes to a byte slave.

2.3 ISA CPU and Bus Master Cycles

The following comments apply to all ISA cycle description diagrams:

Note 1: Heavy black lines indicate the transfer of control from one bus master to another.

Note 2: Shaded areas indicate a "don't care" signal state.

Note 3: Black dots indicate signal sampling points.

2.3.1 8-bit Memory Cycles

Figures 2, 3, and 4 show the relevant signals for standard cycle (6 BCLK), one wait state ISA Cycle (7 BCLK), and no wait state cycle (3 BCLK) memory accesses to 8-bit ISA slaves.

Figure 2 - Memory Access to 8-bit ISA Slave - Standard Cycle (6 BCLK)

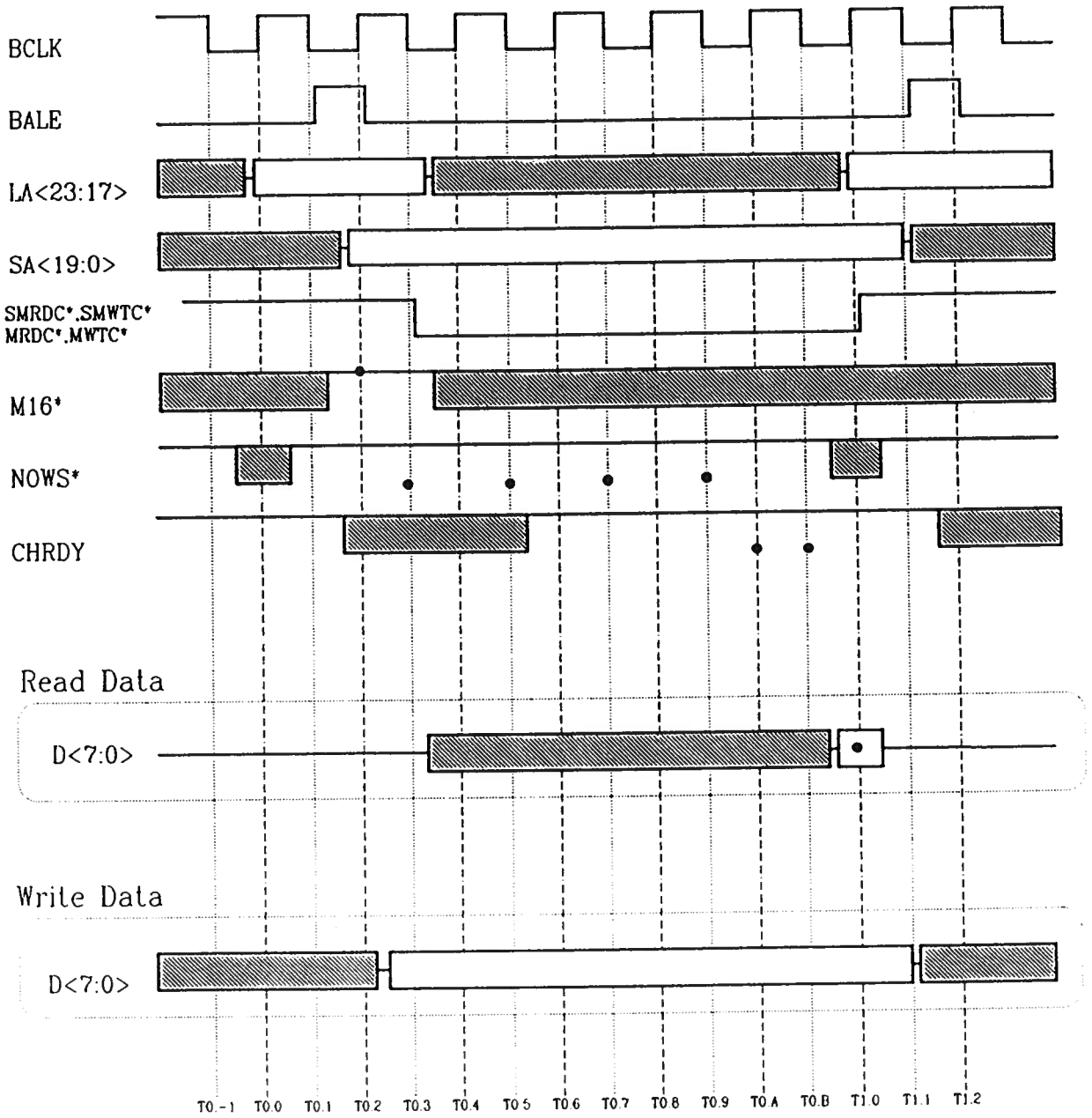


Figure 3 - Memory Access to 8-bit ISA Slave (7 BCLK)

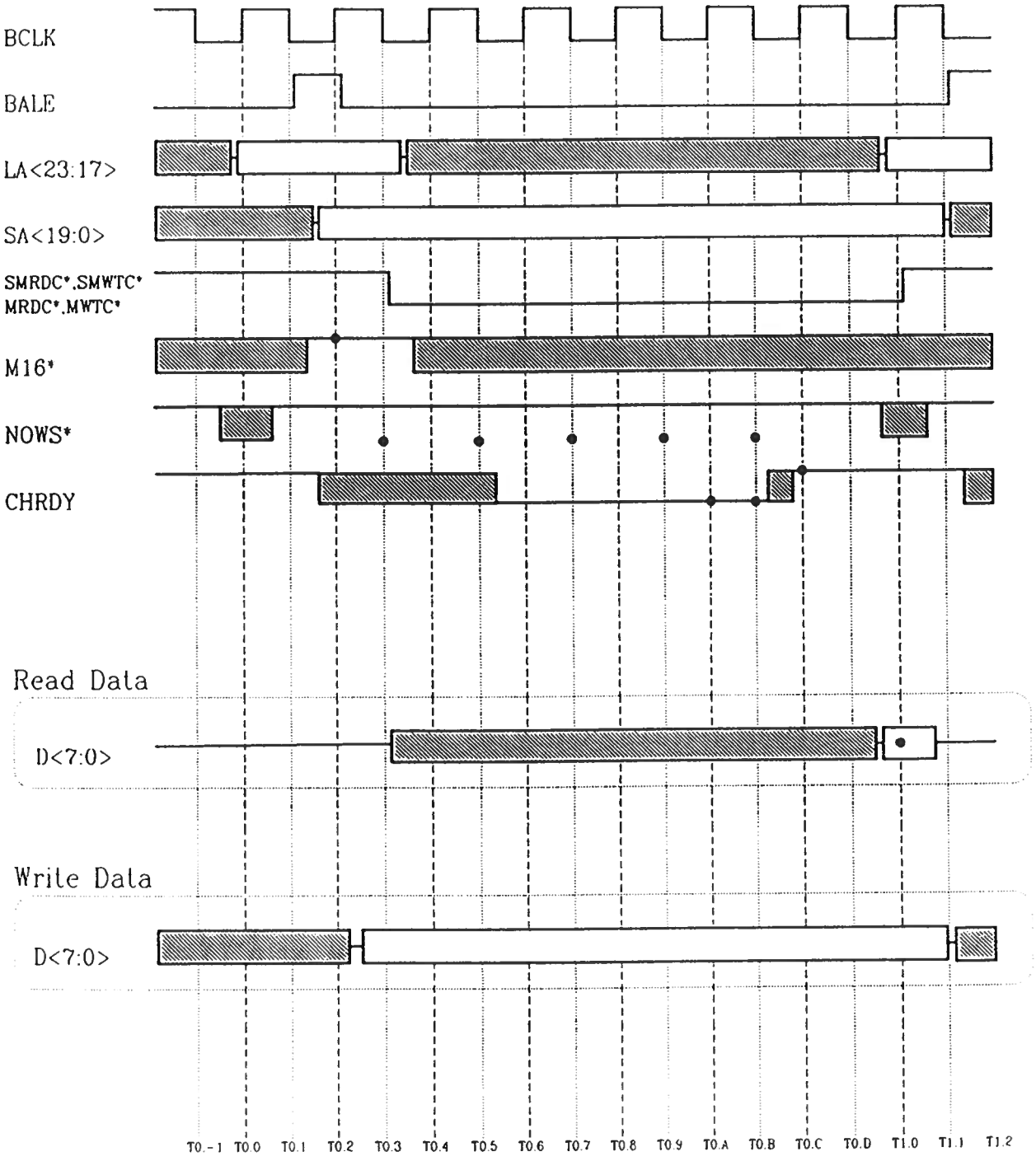
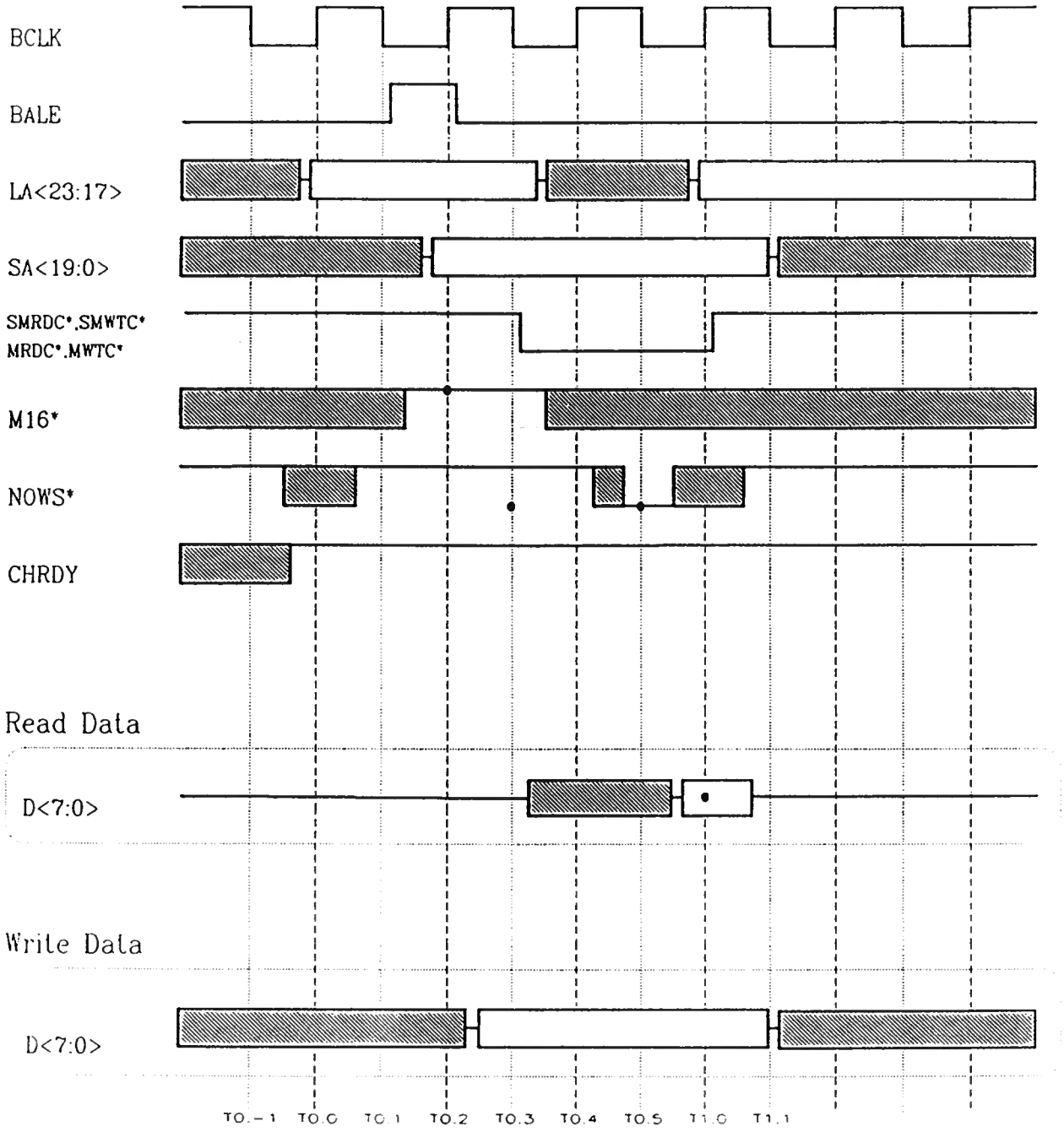


Figure 4 - Memory Access to 8-bit ISA Slave (3 BCLK)



2.3.2 8-bit I/O Cycles

Figures 5, 6, and 7 show the relevant signals for standard cycle (6 BCLK), one wait state ISA cycle (7 BCLK), and no wait state cycle (3 BCLK) I/O, byte accesses to 8-bit ISA slaves.

Figure 5 - I/O Access to 8-bit ISA Slave -
Standard Cycle (6 BCLK)

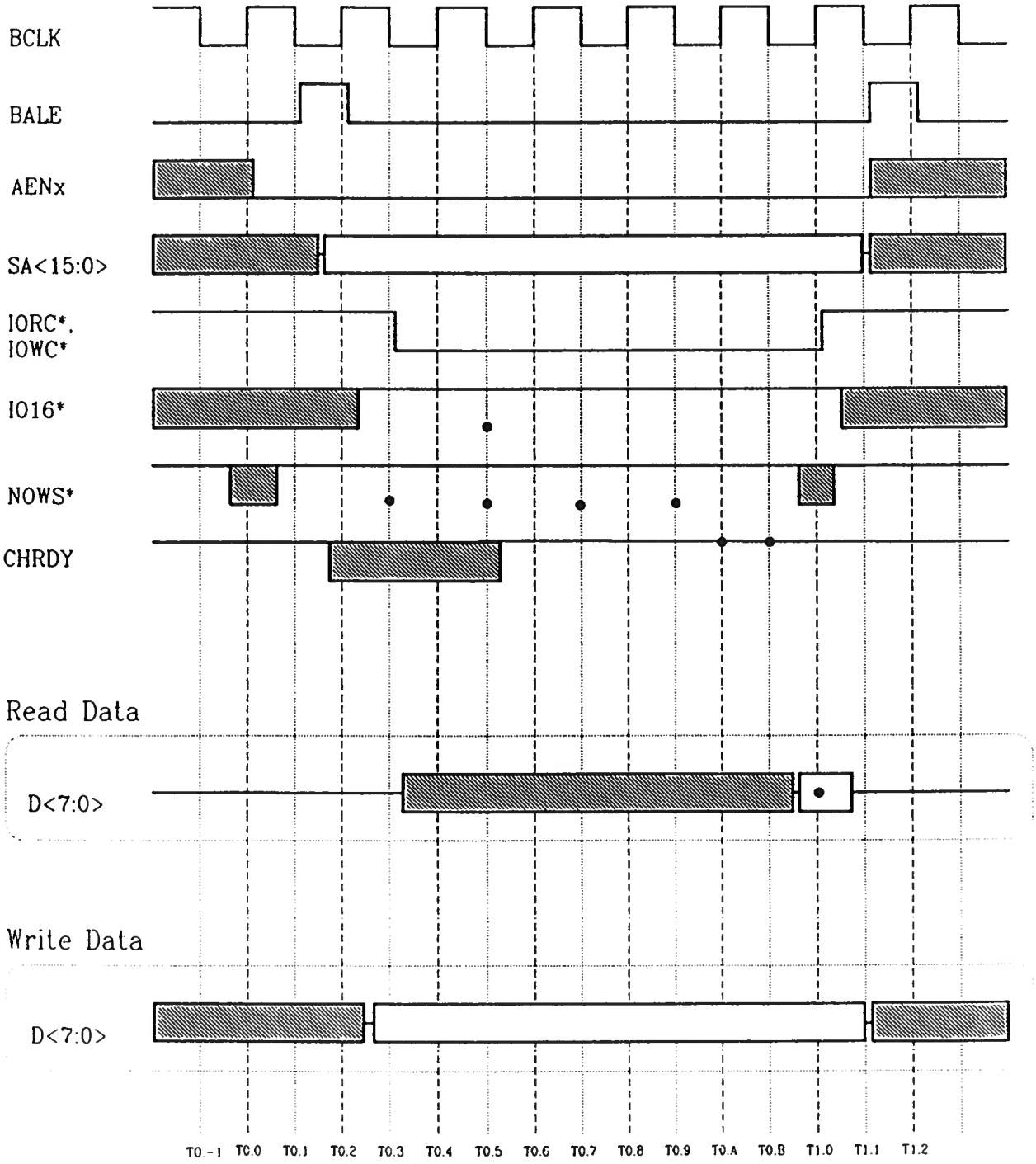


Figure 6 - I/O Access to 8-bit ISA Slave (7 BCLK)

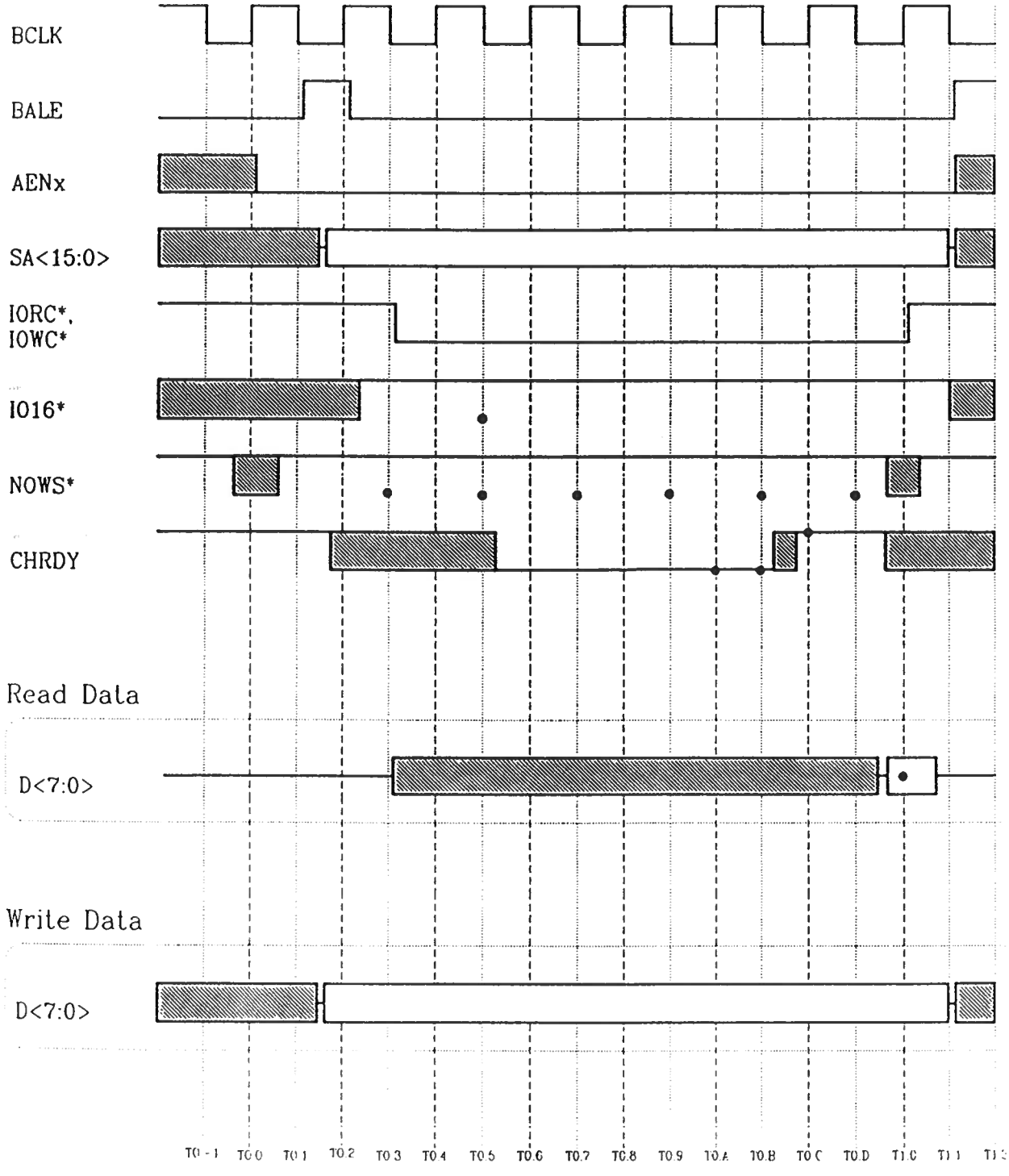
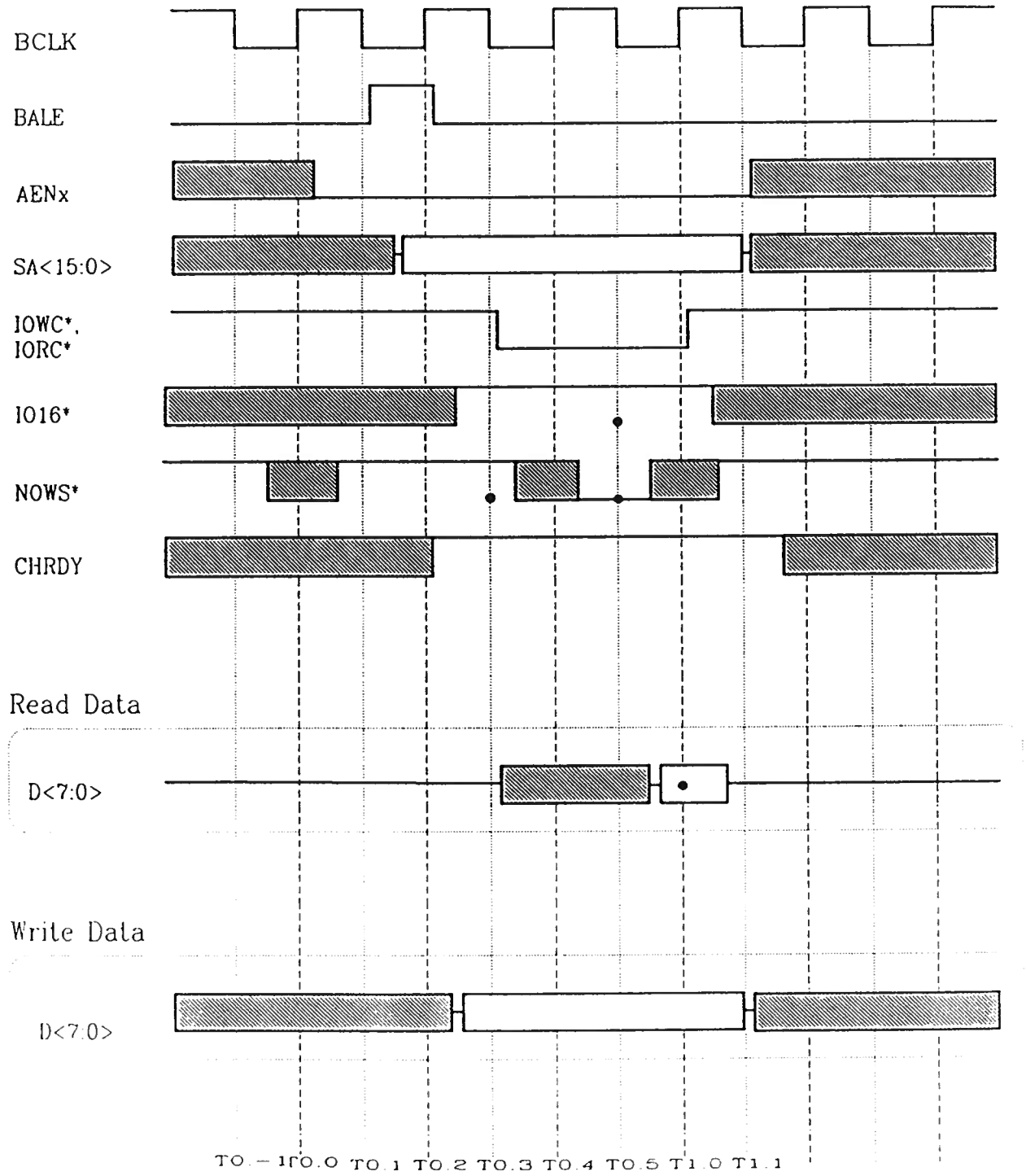


Figure 7 - I/O Access to 8-bit ISA Slave (3 BCLK)



2.3.3 16-bit Memory Cycles

Figures 8, 9, and 10 show the relevant signals for standard cycle (3 BCLK), three wait state ISA cycle (6 BCLK) , and no wait state cycle (2 BCLK) memory, word accesses to 16-bit slaves.

Figure 8 - Memory Access to 16-bit ISA Slave -
Standard Cycle (3 BCLK)

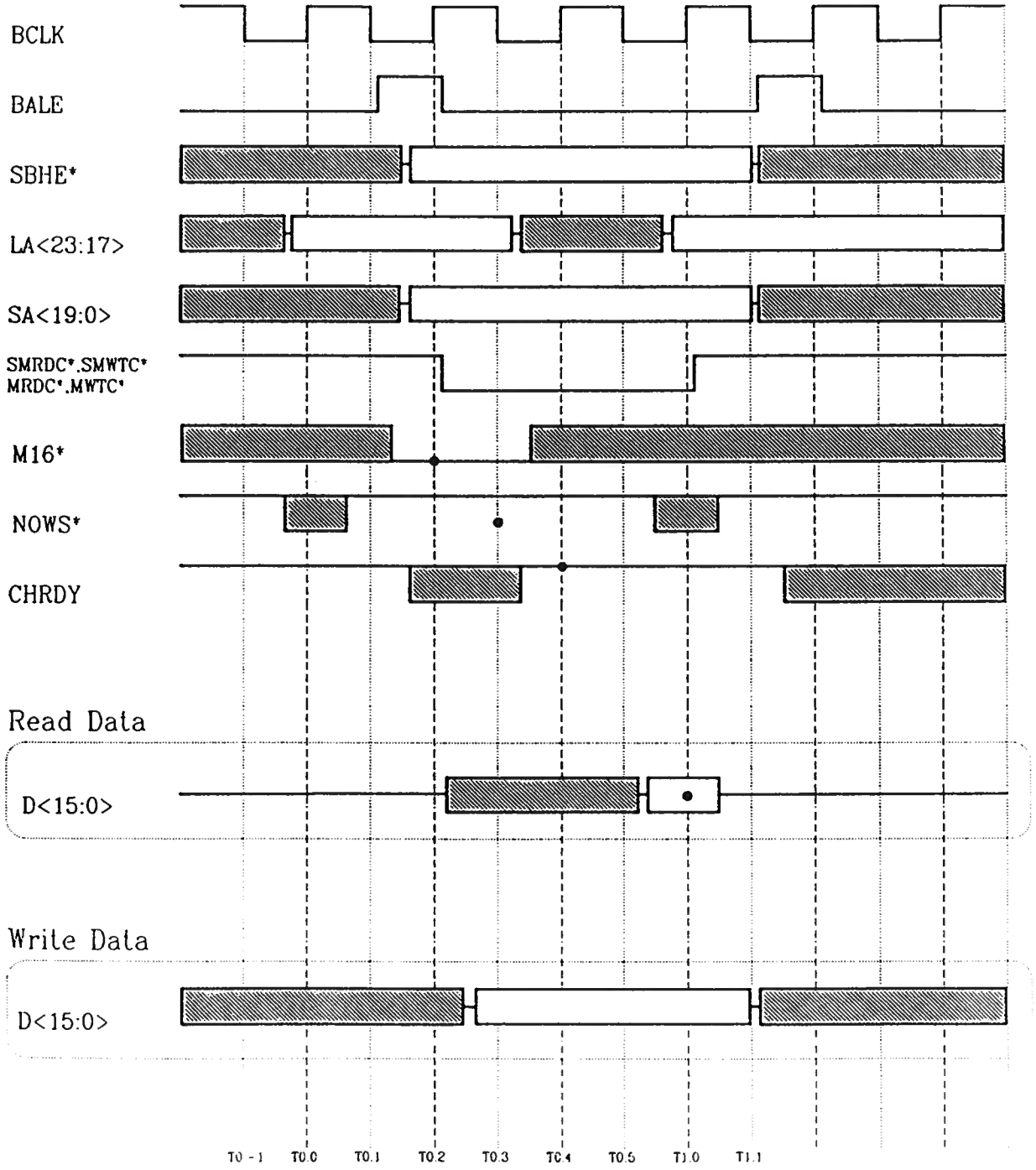


Figure 9 - Memory Access to 16-bit ISA Slave (6 BCLK)

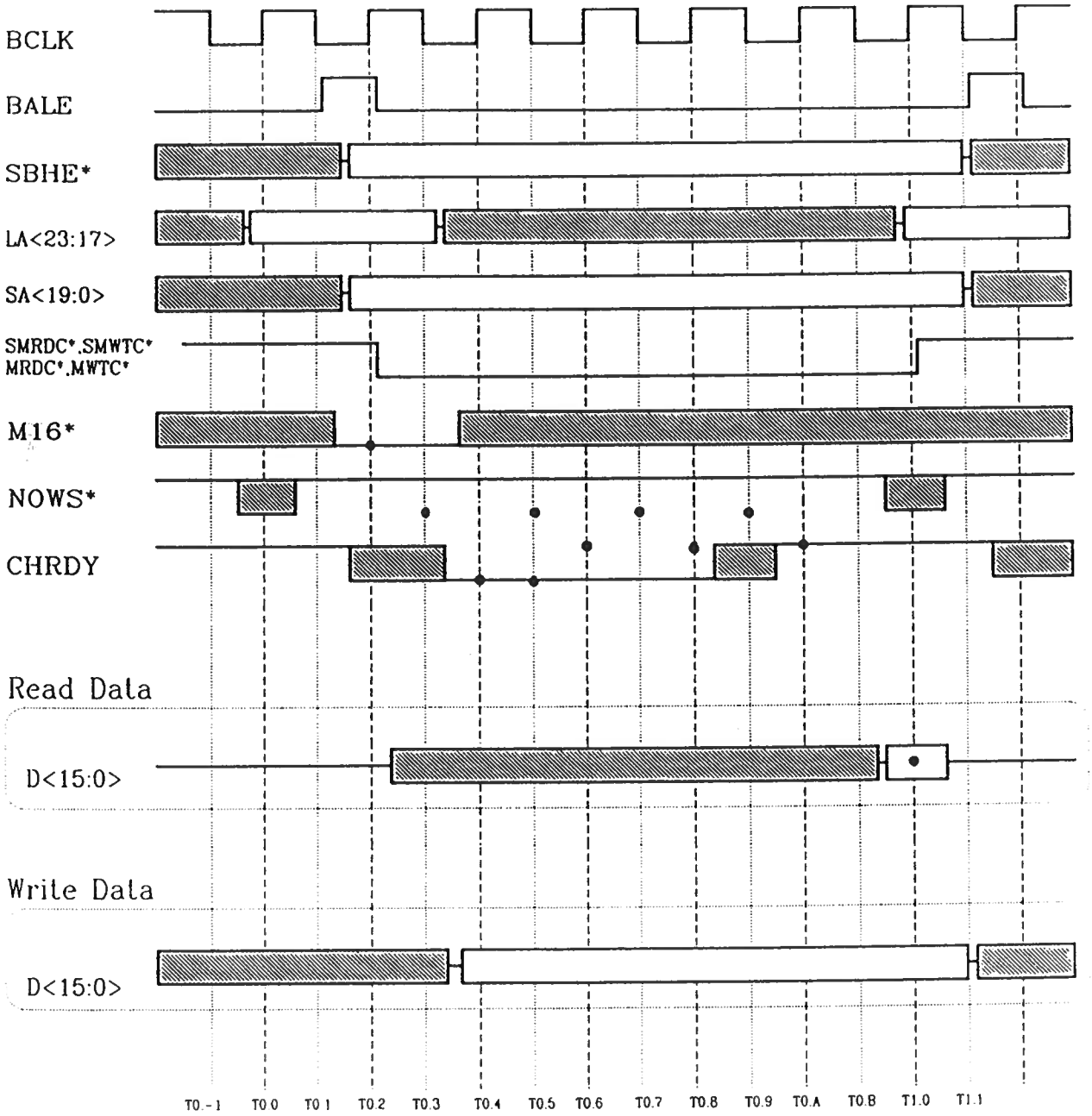
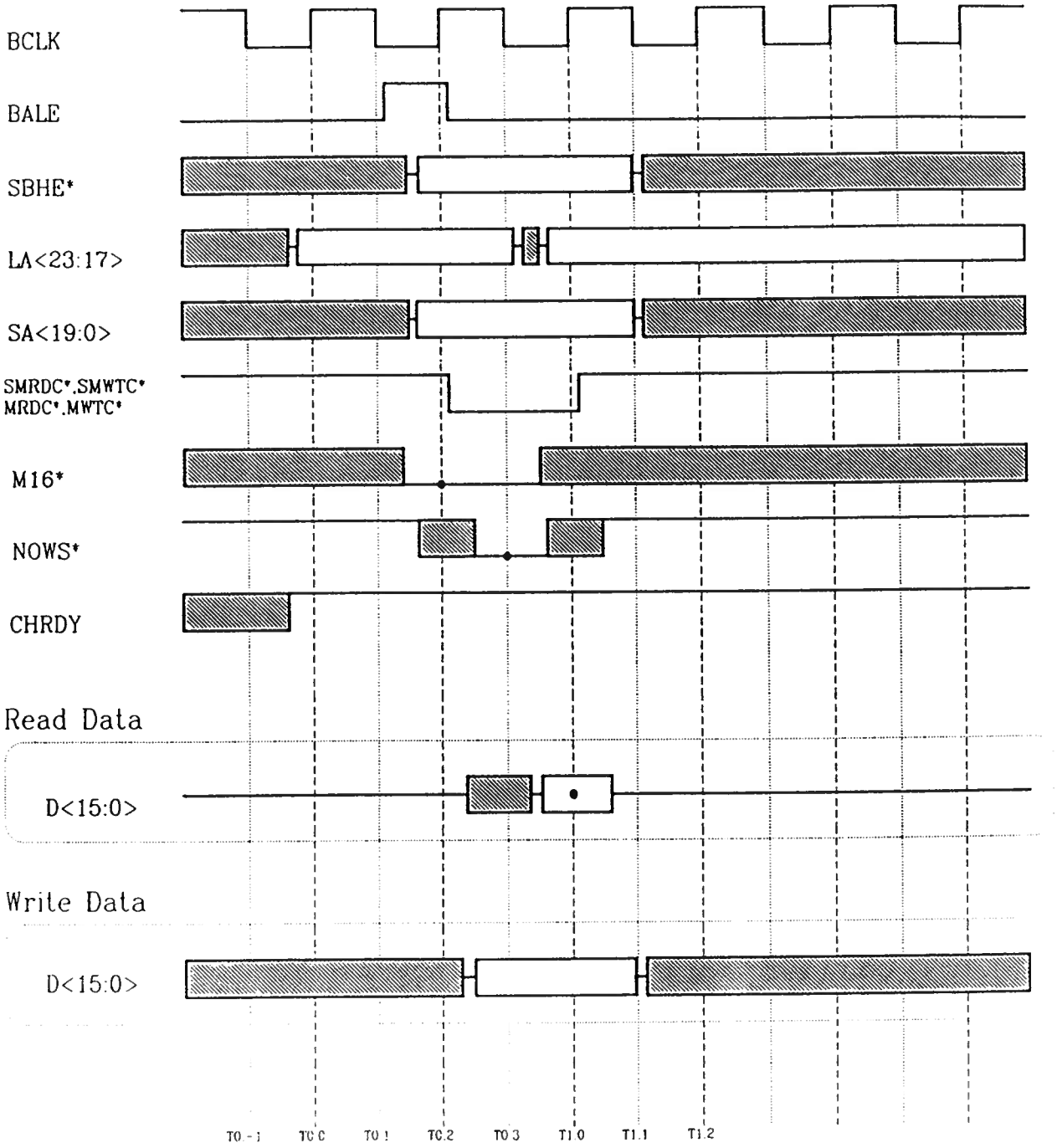


Figure 10 - Memory Access to 16-bit ISA Slave (2 BCLK)



2.3.4 16-bit I/O Cycles

Figures 11 and 12 show the relevant signals for standard cycle (3 BCLK) and three wait state ISA cycle (6 BCLK) I/O word accesses to 16-bit ISA slaves.

Figure 11 - I/O Access to 16-bit ISA Slave -
Standard Cycle (3 BCLK)

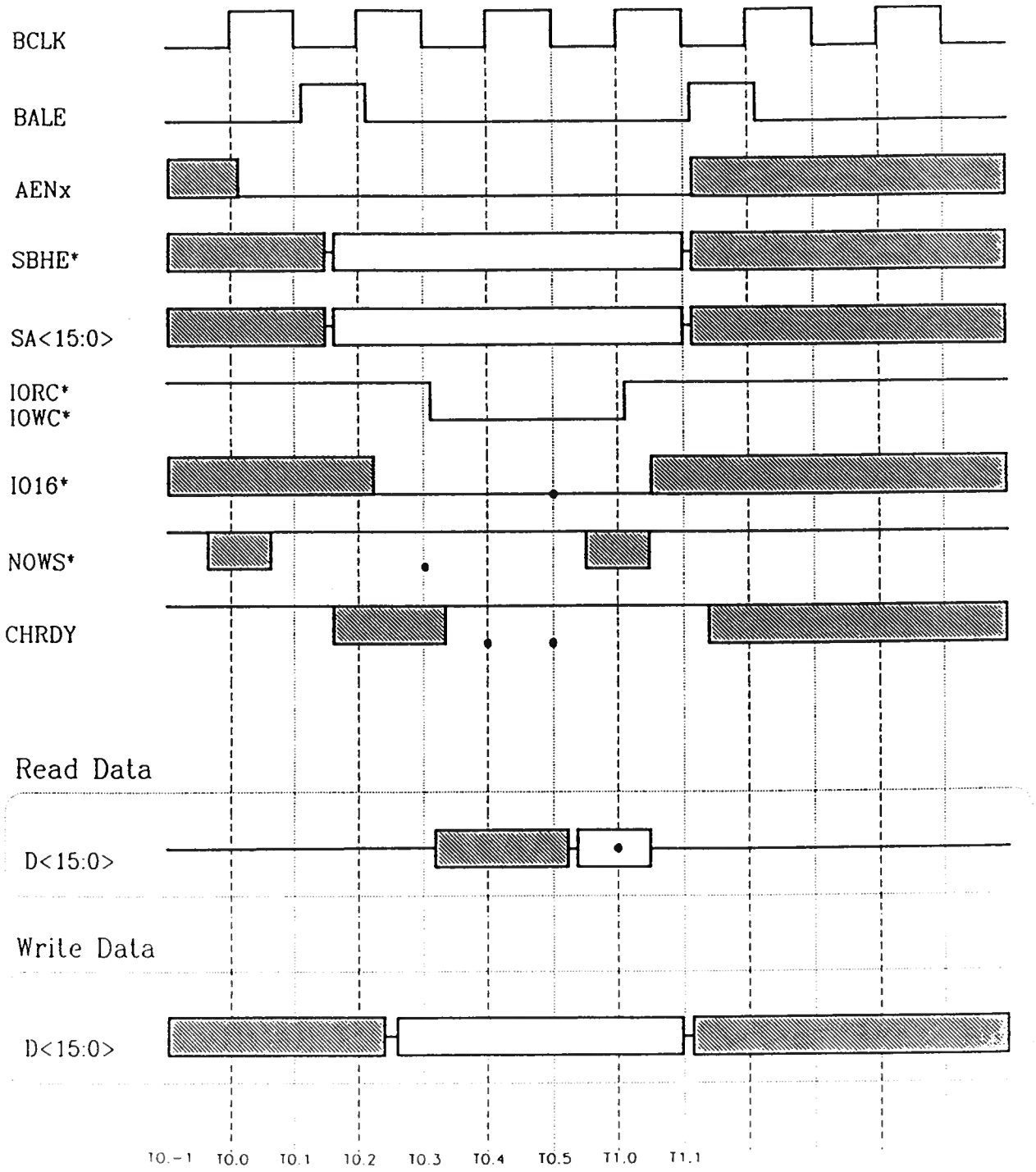
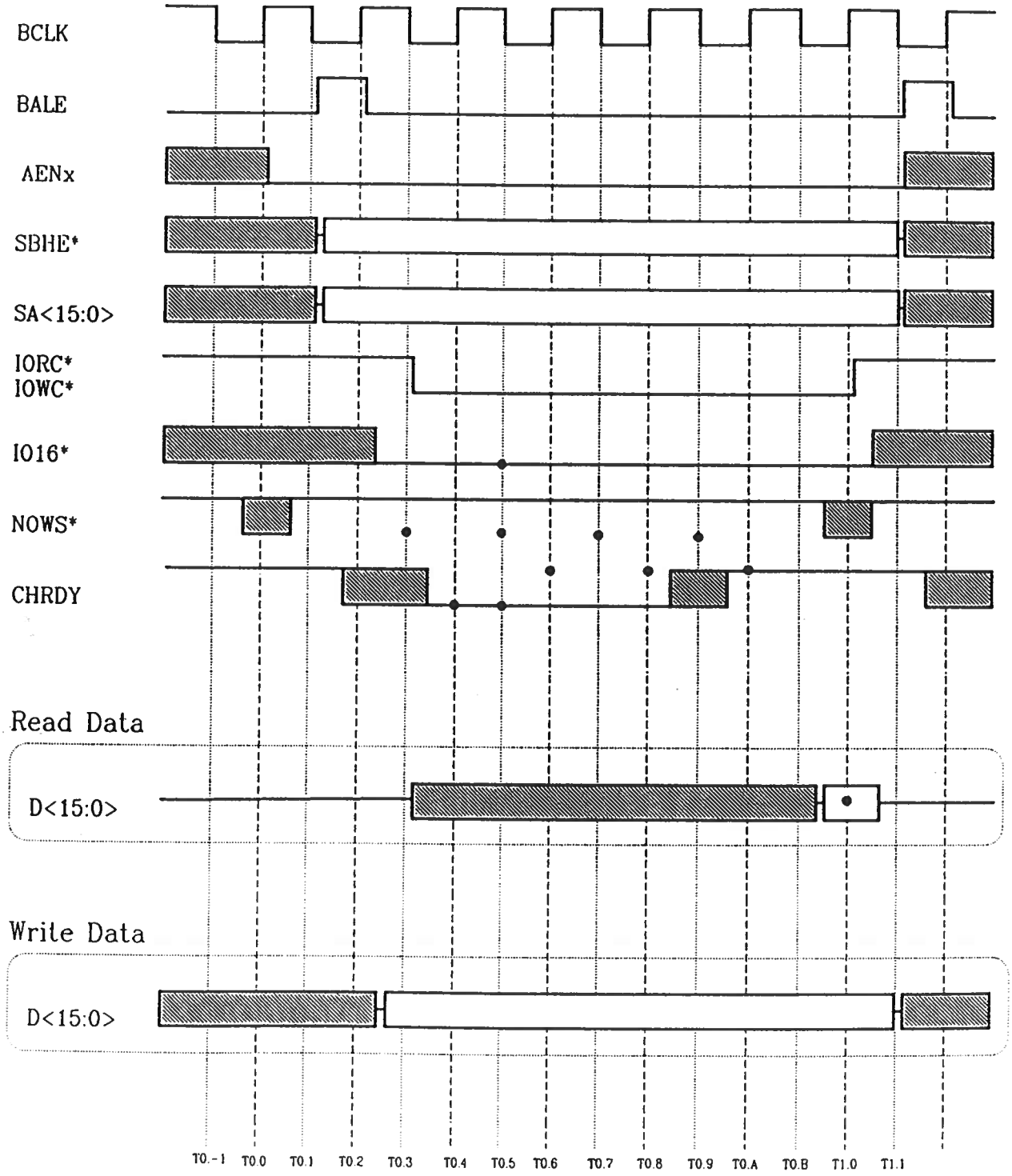


Figure 12 - I/O Access to 16-bit ISA Slave (6 BCLK)



2.4 EISA CPU and Bus Master Cycles

EISA systems provide standard, COMPRESSED and Burst cycle types for data transfers between the main CPU and memory or I/O slaves. EISA bus masters may use standard and Burst cycles, but may not use COMPRESSED cycles.

The following notes apply to the EISA cycle description diagrams:

Note 1: Heavy black lines indicate the transfer of control from one bus master to another.

Note 2: Shaded areas indicate a "don't care" signal state.

Note 3: Black dots indicate signal sampling points.

2.4.1 Standard Memory and I/O Cycles

The standard EISA cycle type completes one transfer each two BCLK periods (zero-wait-state). It can be used to transfer data to or from an EISA memory or I/O slave. Each wait state adds one BCLK period. The total transfer time can be calculated with the following formula:

Total Transfer = $N^*(2+T_W)^*$(1 BCLK period)

Where:

T_W = number wait states in each bus cycle
N^W = number of bus cycles for transfer

For example, an uninterrupted standard transfer of 256 bytes (64 dwords) completes in 15.4 μs for a 32-bit transfer and an 8.33 MHz BCLK. A 16-bit transfer completes in 30.8 μs. This example assumes that no preempts occur during the transfer.

Standard EISA cycles begin with the CPU or bus master presenting a valid address on LA<31:2> and asserting M-IO to indicate a memory or I/O cycle. The address can become valid before the end of the previous cycle to allow address pipelining. The memory or I/O slave decodes the address and asserts the appropriate signals to indicate the type of slave and whether or not it can perform any special timings. The memory or I/O slave asserts EX32* or EX16* to indicate support of EISA cycles. An I/O slave must also decode AENx negated (low) to determine a valid address.

The CPU or bus master asserts START* to indicate the end of the previous cycle and to indicate that the new cycle is now on the bus. The master also asserts W-R to indicate a read or write cycle and BE*<3:0> to indicate the bytes being transferred and their location on the EISA bus. 16-bit transfers use BE*<3:2> (address A1=1) as well as BE*<1:0> (address A1=0) to indicate the bytes to be transferred even though only the low 16-bits of the data bus are used. LA<31:2> and BE*<3:0> remain valid until after negation of START*. A slave that needs to latch the address does so on the trailing edge of START*.

The system board asserts CMD* simultaneously with negation of START* to control the data transfer to or from the slave. If a read cycle is being performed, the slave presents the requested data when CMD* is asserted and holds it valid until CMD* is negated by the system board. For a write cycle, the CPU or bus master presents the data prior to assertion of CMD* and the slave latches it on or before the trailing edge of CMD*. **A memory slave that asserts SLBURST* must sample memory write data on or before a rising BCLK edge with CMD* asserted (regardless of the state of MSBURST*).** The duration of START* and CMD* may vary, depending on the type and speed of the devices performing the transfer.

Wait states can be added to the cycle by slow EISA memory or I/O devices. The slave negates EXRDY after it decodes a valid address and samples START* asserted. The slave may hold EXRDY negated for a maximum of 2.5 μs to complete the transfer, but must release EXRDY synchronous to the falling edge of BCLK to allow the cycle to complete.

The slave *must* allow EXRDY to float high (asserted) synchronously with the BCLK falling edge and must not hold EXRDY negated longer than 2.5 μs.

An EISA I/O slave must assert IO16* as well as EX32* (or EX16*) when addressed if 16-bit ISA bus master compatibility is necessary. IO16* is asserted after decoding a valid address on the LA<31:2> address bus and is latched while CMD* is asserted. M-IO is not included in the address decode for IO16*. EISA I/O slaves that do not need 16-bit ISA bus master compatibility may assert EX32* (or EX16*) only.

The system board develops M16* from EX32* (or EX16*) to assure compatibility with ISA bus masters. An EISA memory slave should not drive M16*.

EISA standard memory and I/O cycles are illustrated in flow diagrams. The flow diagram is a hybrid diagram combining aspects of flow charts and timing diagrams. The flow diagram is intended to demonstrate the basic concepts for various cycles performed on the EISA bus. At least one sample of every possible "action" (such as wait states and Burst termination) is provided, although, of course, every possible combination of bus cycle is not shown.

The flow diagrams consist of flow-chart-like blocks and arrows, with board-specific actions enclosed in the blocks. Line types (solid, dotted, bold) are used to differentiate between the parts of the system involved (such as system board, slave, and bus controller). The flow diagram is divided into horizontal sections, each section representing the BCLK edge or level during which the enclosed action occurs. Note that at the beginning of many cycle types BCLK may not be active. In this case the BCLK states are drawn with dotted lines.

Flow diagrams do not follow the conventions of normal flow charts in that there is no "decision" block. In essence, the flow diagrams answer a question such as "To design a 32-bit one-wait-state EISA memory board, what signals apply during an access to the board." The designer would then follow the flow diagrams for accesses to 32-bit memory, and when a branch labeled "Wait states needed," appeared that branch would be followed to add the desired number wait states.

Flow diagrams should be used to gain an initial understanding of the EISA bus cycles. They also provide a means of following the sequence of signals when reading the timing diagrams. Once the designer understands the basic cycle types, specific information on timing and special cases should be obtained from the timing diagrams themselves. In the event of a conflict of information, the timing diagrams should be assumed to be correct.

Figures 13 and 14 illustrate the flow of a data transfer from a 32-bit master to 32-bit slave memory (read and write cycles). The figures include standard and COMPRESSED cycles. Data transfers from a 16-bit master to a 16-bit slave are the same except for the use of EX16* instead of EX32*.

Figure 15 shows the relevant signals for 2 and 3 BCLK EISA slave accesses.