The Zorro III Bus Specification

A General Purpose Expansion Bus for
High Performance Amiga Computers

Document Revision 1.10

Vernal Equinox Release

by Dave Haynie
March 20, 1991

IMPORTANT INFORMATION

"A life spent making mistakes is not only more honorable but more useful than a life spent doing nothing."
-George Bernard Shaw

This Document Contains Preliminary Information

The information contained here, while a honest attempt to get as much Zorro III information down on paper as early and accurately as possible, is still somewhat preliminary in nature and subject to possible errors and omissions. Being early in the life of the Zorro III bus, very few Zorro III cards have yet been designed, so some features described here have not actually been tested in a system, or in some cases, actually implemented as of this writing. That, of course, is one major reason for having a specification in the first place.

Commodore Technology reserves the right to correct any mistake, error, omission, or viscious lie. Corrections will be published as updates to this document, which will be released as necessary in as developer-friendly a manner as possible. Revisions will be tracked via the revison number that appears on the front cover. New revisions will always list the corrections up front, and developers will be kept up to date on released revisions via the normal CATS channels.

ACKNOWLEDGEMENTS

"Art is I; science is we."
-Claude Bernard

I'd like to acknowledge the following people and groups, without whom this new stuff would have been impossible:

- The original Amiga designers, for designing the first microcomputer bus with support for multiple masters, software board configuration, and room to grow.

- The rest of the A3000 Engineers: Greg Berlin, Hedley Davis, Scott Hood, and Scott Schaeffer; PCB master Terry Fisher; and the lab maniacs George Terbush, Brian Fenimore, and Dan Faust. And of course the A3000 boss men, Jeff Porter and Henri Ruben, who let it all happen.

- The folks who helped review the original version of this document, overnight: Joe Augenbraun, Dan Baker, Hedley Davis, Bryce Nesbitt, and Jeff Porter. And to the numerous folks who've helped out with questions, corrections, and other feedback ever since.

- The Commodore-Amiga software group, and the Commodore Semiconductor Group, for excellent support in their respective areas. Though, about that Rev H chip.....

- Commodore's Developer Support people from both sides of the Atlantic.

- Gold Disk, for some good and relatively bug free electronic publishing software.

- Iggy; an excellent cat, an excellent foot warmer.

TABLE OF CONTENTS

CHAPTER 7 MECHANICAL SPECIFICATIONS

CHAPTER 8 AUTOCONFIG®

APPENDICES

TABLES AND FIGURES

CHAPTER 1

INTRODUCTION

"Welcome, my son. Welcome to The Machine."
-Pink Floyd

This document describes the complete Zorro III bus, first implemented in the Amiga 3000 Computer. The Zorro III bus is a performance 32 bit expansion bus that is also upward compatible with the Zorro II bus (Amiga 2000 expansion bus). The main intent of the Zorro III bus is to allow fast 32 bit peripherals and memory devices to be added to a high performance Amiga, such as the Amiga 3000, while at the same time allowing standard Zorro II devices to be used wherever they make sense in such a system. This compatibility also insures that the Amiga 3000 will have a number of hardware and software compatible expansion devices available upon introduction, and that Amiga 2000 owners will be able to take their expansion card investment along with them should they migrate to a higher performance Amiga.

1.1 Intended Audience

This document was written primarily for hardware engineers interested in designing Plug In Cards for the Zorro III expansion bus. While it may occasionally be of use to software engineers interfacing to such Zorro III PICs, Amiga system software provides an interface layer (*expansion.library* in the Amiga OS) which manages the needs of most card-level software. A reasonable level of microcomputer knowledge is prerequisite to get much meaning out of these pages. A good understanding of the Motorola 680x0 processors will be quite useful, as will be an understanding of the Zorro II expansion bus used on earlier Amiga computers such as the Amiga 2000.

1.2 Bug Reports

This is the second major publication of the *Zorro III Bus Specification*. While every effort has been made to keep it as accurate as possible, there is certainly the possibility that some errors have made it into this document. Anyone finding any error is encouraged to contact Commodore at the address below:

Dave Haynie/A3000 Systems Engineering
1200 Wilson Drive
West Chester, PA 19380

Bugs can also be reported on BIX or via Usenet; on BIX, use the "amiga.com/hardware" conference, or contact Dave Haynie directly as "hazy"; for Usenet users, bug reports can be sent to the address "{uunet,rutgers}!cbmvax!bugs" (use "cbmvax.cbm.commodore.com" if you like domain names); please also copy any such reports to "{uunet,rutgers}!cbmvax!daveh".

1.3 Amiga Bus History

The original Amiga computer, the Amiga 1000, was introduced in 1985. While it had no built-in standard for expandability, the capability for some form of expansion was considered extremely important; personal computer history up to that date had shown several times that an open hardware expansion capability was often critical to a personal computer's success and to its capability to adapt to new or unusual applications. The A1000 was designed with a connector giving access to the internal 68000 bus and a few other system signals. Shortly after introduction, the formal expansion specification for a card chassis that would connect to the A1000 was published. This bus became commonly known as the Zorro bus*. While the backplane specification was very easy to implement with 1985 PAL technology based on the existing 68000 signals, the specification did incorporate a number of advanced features. Far more sophisticated than the IBM-XT/AT and Apple II buses in common use at the time, the Zorro bus allowed any slot to master the bus, and it linked expansion cards with the system software. Addressing jumpers were eliminated, the card's address instead being assigned by software, and cards could easily be identified by software and linked with appropriate driver programs, all with a minimum of user intervention.

With the introduction of the Amiga 2000 system, the Zorro bus was changed slightly. Additional discrete interrupt lines were added, replacing the encoded lines that couldn't easily be used by any bus resident device. As it turns out, these additional encoded lines weren't any more useful, as they couldn't be disabled by software, and as such, they're no longer considered an official part of the Zorro II bus specification (they are supported as part of Zorro III). Finally, the form factor was changed to match that of the IBM PC-AT card, acting as both a cost reduction and allowing the Zorro II bus to offer the PC-AT bus as one optional secondary bus extension. This modified specification became commonly known as the Zorro II bus, and it's the

* The original "Zorro" name comes from the code name of one of the A1000 prototype boards. The "Zorro" board was the one that followed the "Lorraine", and was the board in the works when much of the expansion specifications were worked up. Since everyone uses the "Zorro" name, and no one's suggested a better name, I stick with it throughout this document.

Amiga bus standard that's been in use for most of the Amiga's life. And it's a bus standard that will continue to be important.

1.4 The Zorro III Rationale

With the creation of the Amiga 3000, it became clear that the Zorro II bus would not be adequate to support all of that system's needs. The Zorro II bus would continue to be quite useful, as the current Amiga expansion standard, and so it would have to be supported. A few unused pins on the Zorro II bus and the option of a bus controller custom LSI, gave rise to the Zorro III design, which supports the following features:

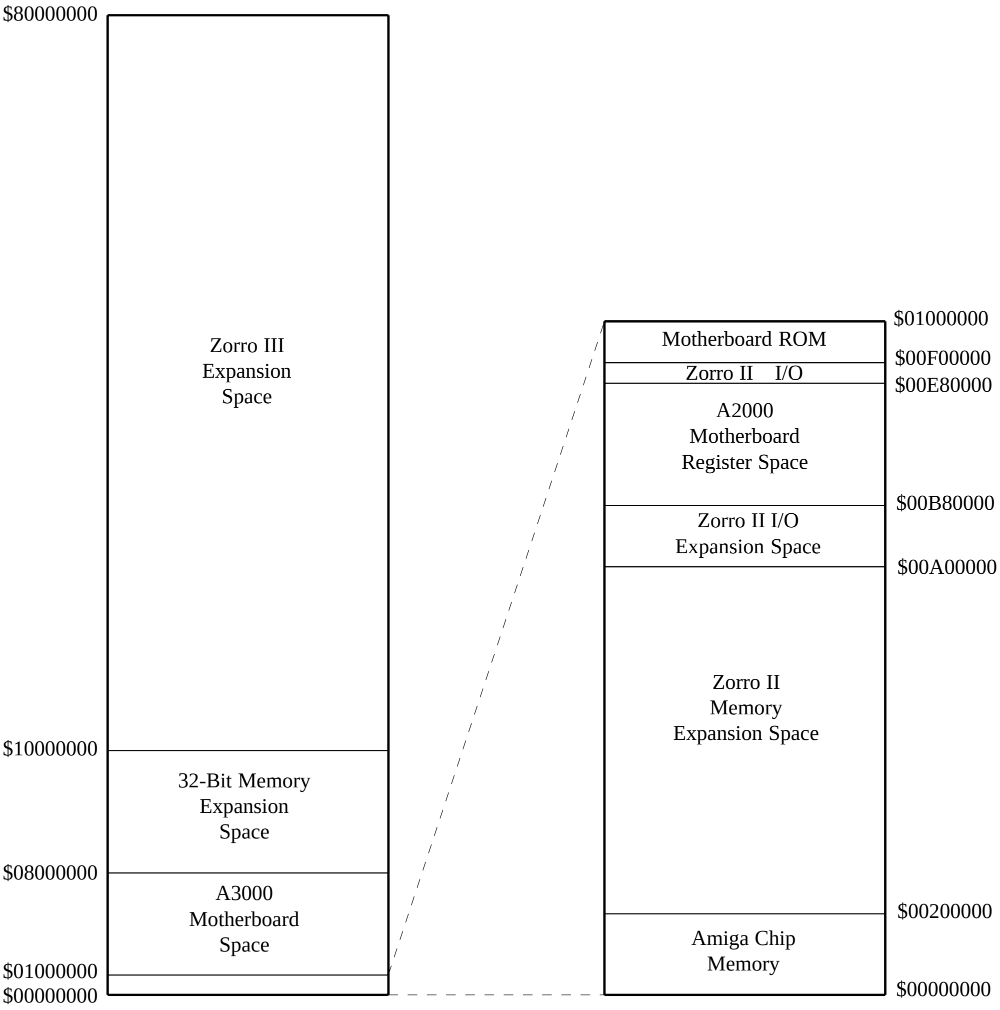
- Compatibility with all Zorro II devices.
- Full 32 bit address path for new devices.
- Full 32 bit data path for new devices.
- Bus speed independent of host system CPU speed.
- High speed bus block transfer mode.
- Bus locking for multiprocessor support.
- Cache disable for simple cache support.
- Fair arbitration for all bus masters.
- Cycle by cycle bus abitration mode.
- High speed interrupt mode.

Some of the advanced features, such as burst modes, are designed in such a way as to make them optional; both master and slave arbitrate for them. In addition, it is possible with a bit of extra cleverness, to design a card that automatically configures itself for either Zorro II or Zorro III operation, depending on the status of a sensing pin on the bus.

The Zorro III bus is physically based on the same 100 pin single piece connector as the Zorro II bus. While some bus signals remain unchanged throughout bus operation, other signals change based on the specific bus mode in effect at any time. The bus is geographically mapped into three main sections, *Zorro II Memory Space*, *Zorro II I/O Space,* and *Zorro III Space*. The memory map in *Figure 1-1* shows how these three spaces are mapped in the A3000 system. The Zorro II space is limited to a 16 megabyte region, and since it has DMA access by convention to chip memory, it is in the original 68000 memory map for any bus implementation. The Zorro III space can physically be anywhere in 32 bit memory.

The Zorro III bus functions in one of two different major modes, depending on the memory address on the bus. All bus cycles start with a 32 bit address, since the full 32 bit address is required for proper cycle typing. If the address is determined to be in Zorro II space, a Zorro II compatible cycle is initiated, and all responding slave devices are expected to be Zorro II compatible 16 bit PICs. Should a Zorro III address be detected, the cycle completes when a Zorro III slave responds or the bus times out, as driven by the motherboard logic. It is very important that no Zorro III device respond in Zorro III mode to a Zorro II bus access; as the following chapters will reveal, the two types of cycles make very different use of many of the expansion bus lines, and serious buffer contention can result if the cycle types are somehow mixed up. The Zorro III bus of course started with the Zorro II bus as its necessary base, but the Zorro III bus mechanisms were designed as much as possible to solve specific needs for high end Amiga systems, rather than extend any particular Zorro II philosophy when that philosophy no longer made any sense. There are actually several variations of the basic Zorro III cycle, though they all work on the same principles. The variations are for optimization of cycle times and for service of interrupt vectors. But all of this in due time.

Figure 1-1: A3000 Memory Map



1.5 Document Revision History

While there’s significantly more real Zorro III hardware actually in existence at the time of this writing than when the first revision of this document was created, various Zorro III issues are still, from time to time, changing. In order to document these changes, this section was created. Although revision histories often discuss revisions in reverse chronological order, it’s done here in chronological order to keep the subsection numbers consistent between revisions of this document.

1.5.1 Changes for Rev 0.90

The major changes in Rev 0.90 are actually additions. Specifically, the remaining parts of the Zorro III Timing (Chapter 5) and Mechanical (Chapter 7) specifications have been incorporated into this document. Additionally, the Zorro III design example in Appendix A.4 has been deleted. This simple and somewhat kludgy example has been surplanted by a more useful, straightforward, and throughly explained example, available as the separate document *BIGRAM 8/32: A Complete Zorro III Design Example*. In general, we expect both documents to be distributed together, but as always, CATS can assist in the procurement of any missing information.

1.5.2 Changes for Rev 0.91

In the Introduction (Chapter 1), the official revision history has been added as a standard part of this document. The Zorro III Bus Architecture (Chaper 3), section 3.5, has been changed to reflect the revised Quick Interrupt vector allocation mehanism. In the Timing specification (Chapter 5), corrections have been made: timing parameter 6 was left out of the section 5.3 timing, and timing parameter 19 was incorrectly specificed in section 5.4. In the AUTOCONFIG® specification (Chapter 8), corrections have been made to the addressing tables for registers **44** and **48**. Also the Quick Interrupt Enable bit (register **08**:4) and Vector Register (register **50**) have been deleted from the specification. Quick Interrupt Vector allocation is now handled via an Exec call, a single configuration unit can have several vectors, and the means of storage on a PIC is up to the designer.

1.5.3 Changes for Rev 1.00

In the AUTOCONFIG® specification (Chapter 8), bit 4 of register **08** has been changed to always read 1 for Zorro III PICs. This change was necessary for compatibility with 1.3, due to a bug in the 1.3 expansion.library. Also, the nybble write configuration mode for the Zorro III configuration block has been eliminated, only byte and word writes are now supported.

1.5.4 Changes for Rev 1.01

The AUTOCONFIG® specification change listed in 1.5.3 was missing from Chapter 8 in Rev 1.00 of the spec, now it's actually there. Additionally, some clarification on the proper action of slave cards and bus error conditions has been added to Chapter 4.

1.55 Changes for 1.10

The /INT$_1$, /INT$_4$, /INT$_5$, and /INT$_7$ lines have been eliminated from the Zorro III bus specification. Although the A3000 hardware supports these, some AmigaOS software conventions makes their use impossible under the AmigaOS. These lines are now considered reserved. Also, in section 3.5, the vector poll command code was given as 16, where it's actually 15; this has been corrected.

CHAPTER 2

ZORRO II COMPATIBILITY

"In Jersey anything's legal, as long as ya don't get caught."
- Traveling Wilburys

The A3000 bus is a rather extensive superset of the A2000 bus design. The compatibility is based on distinct bus modes, rather than a simple extension to the existing bus mechanisms. Through the use of an integrated bus controller (the Fat Buster chip), the expansion bus configures itself differently for the 16 bit A2000-compatible Zorro II modes than the 32 bit Zorro III modes. As a result, while there are still only 100 pins on the expansion bus, some pins change function considerably depending on the bus activity that's currently in progress. While the Zorro II modes of the Zorro III bus are as compatible as possible with the Zorro II bus specification (especially the A2000 implementation of this specification), there are some small differences between the two expansion buses.

Aside from these differences, in general, it's important to understand the Zorro II bus in order to understand the Zorro III bus. The general features of the A3000 bus, like autoconfiguration, the master-slave bus architecture, and the physical attributes come from the Zorro II expansion bus. Other features of the Zorro III bus address shortcomings of the Zorro II architecture, but Zorro II has a hand in how some of these shortcomings are solved under Zorro III. Those with a full understanding of the Zorro II bus will mainly be concerned with the possible bus incompatibilities listed here.

2.1 Changes From The A2000 Bus

While much effort has been made to assure that the Zorro II mode of the A3000 bus is as compatible as possible with the A2000 bus, there are a few points to consider here. Primarily, the A3000's Zorro II modes are driven with a state machine that emulates the 68000 bus protocol. This emulation must be based on the published Motorola specifications detailing 68000 bus behavior. While this has the interesting effect of changing the Zorro II bus from CPU dependent to CPU independent, there's some margin for trouble. Zorro II PICs also designed to these specifications should have no trouble in the A3000 bus in most cases. However, anything designed based on observed 68000 behavior rather than documented 68000 operation is at serious risk of failing in an A3000 bus, as one might expect. There are also actual documented differences, which are listed below.

2.1.1 6800 Bus Interface

A major difference between the A3000 expansion bus in Zorro II mode and the A2000 bus are the absence of the signals /VPA and /VMA, which comprise the 6800/6502 peripheral support mechanism that's part of the 68000 bus interface. This mechanism was never a supported part of the Zorro II specification, however, and it should not be used by any PIC. Any Zorro II PIC that depends on /VPA or /VMA will not work in the A3000 bus. It was, in fact, impossible to legally use this on the A2000 bus. The E clock is, however, supported on the Zorro III bus, though its duty cycle may vary in some situations.

2.1.2 Bus Memory Mapping and Cache Support

Another change to the Zorro II implementation is that the bus mapping logic works a little differently. Zorro II address space is broken up into memory and I/O address space. Memory space is the standard 8 megabyte space from $00200000-$009FFFFF. The I/O address space is mapped at $00E80000-$00EFFFFF, and a new 1.5 megabyte section (previously reserved for motherboard devices) from $00A00000-$00B7FFFF. Zorro II cycles are not generated for non-Zorro II address space, even for 68000 space resources on the local bus. So, for example, a CPU access to chip memory would be visible to a Zorro II PIC in an A2000 backplane, but invisible to that same PIC in an A3000 backplane. Since this extra information on the Zorro II backplane can't be legally used by any PIC anyway, it should not be used by any existing A2000 PICs.

The reason for the two distinct mapping regions is for cache support of Zorro II PICs. All access by the local bus[*] master to Zorro II memory space results in the local bus cache enable signal being driven and a full port read (eg, both bytes) regardless of the actual data transfer size being requested. A local bus access to Zorro II I/O space results in the local bus cache disable signal being driven and the data strobes for reads indicating the requested transfer size. This cache mapping mechanism was first implemented in the A2630 coprocessor card, so it's not an entirely new concept.

[*] The *local bus*, motherboard bus, and CPU bus are the same thing; the immediate 680x0 bus connected directly to the CPU in an Amiga computer. Current Amiga computers typically support three distinct buses; the expansion bus, local bus, and chip bus. From the point of view of the expansion bus, the local and chip buses appear as a unified device which may be master or slave to the expansion bus.

2.1.3 Bus Synchronization Delays

Due to the asynchronous nature of the local-to-expansion bus interface for Zorro II cycles, extra wait states may occasionally be added for local to expansion or expansion to local cycles. These are generally manifested as delays between consecutive cycles, since the bus controller is not going to require extra waiting during the cycle -- things will have already been synchronized at that point. The synchronization problems get more difficult for Zorro II master access to local bus slaves, and as a result, wait states here are very common. The actual number of wait states generated in any case will be based on the particular implementation.

2.1.4 Zorro II Master Access to Local Slaves

The only supported local bus resource that's guaranteed accessible to a Zorro II expansion bus master as a slave device is chip bus memory. All I/O devices are implementation dependent and not supportable via DMA. Any attempted access to unsupported local bus resources as expansion slaves will result in an error condition being signalled on both the local and the expansion buses. Most other local bus resources, such as local bus fast memory, are located outside of Zorro II space on most systems and obviously not available to Zorro II masters.

2.1.5 Bus Arbitration and Fairness

The Zorro II bus is now arbitrated fairly. The normal slot-based order of precedence is given to requesting devices, just as in the A2000 implementation. As always, once a bus master assumes bus mastership, it has the bus for as long as it wants the bus (of course, trouble can result if a device takes the bus over for too long). Once a master gives up the bus, it will not be granted it back until all subsequent requests have been serviced. Bus arbitration at its best will be slightly slower than in the A2000 implementation, due to the fairness logic, but it is impossible to jam the arbiter with asynchronous bus requests as in the A2000. The new style arbiter also holds off bus grants while hidden local bus cycles are in progress, so there's no guarantee of a minimum time between bus request and bus grant specified.

2.1.6 Intelligent Cycle Spacing

In order to permit a free intermix of Zorro II and Zorro III cycles, the bus control logic is capable of making intelligent decisions when spacing bus cycles. In somc cases, a Zorro II cycle has some component that would naturally extend into a following cycle. The cycle spacing logic detects such a condition, and refuses to start a new cycle until the current one is complete, even if this extends beyond the defined bounds of a Zorro II cycle. For Zorro II PICs that really follow the Zorro II specifications, this should have no effect. However, any Zorro II PIC that holds signals much beyond the end of a cycle, especially critical signals like /SLAVE and /DTACK, will likely incur additional wait states on the Zorro III bus. This is not intended as a license for making sloppy expansion card designs, just an acknowledgement that some Zorro II devices may cause a conflict with the faster Zorro III bus timings, and the best thing to do about such cases is to make them work, even with a possible performance penalty.

2.1.7 Bus Drive and Termination

Finally, the Zorro III bus uses different bus termination than that in the A2000. The Zorro II specification didn't specify the termination expected; backplanes were built that didn't even have termination. The A2000 bus used a circuit consisting of a capacitor in series with a resistor to ground for most of the bus signals. This has good reflection cancelling properties without increasing crosstalk (a major concern on the 2-layer A2000 motherboard), but it does slow things down measureably. The main reason for the change on the A3000 backplane is to support the faster Zorro III bus modes. The multi-layer A3000 motherboard permits a reasonably high current bus without undue crosstalk. The thevenin termination makes switching logic levels start from a midpoint instead of a rail, especially for a bus coming out of tri-state (which, based on the Zorro III design, happens constantly). This should not cause problems with Zorro II cards, but it's conceivable that some cards may need to be adjusted to work in this bus (the Zorro III bus requires somewhat higher current capability than the Zorro II bus does. The A3000 does not support enough slots for loading to be a likely problem, but future Zorro III backplanes will have more slots and make this an important consideration).

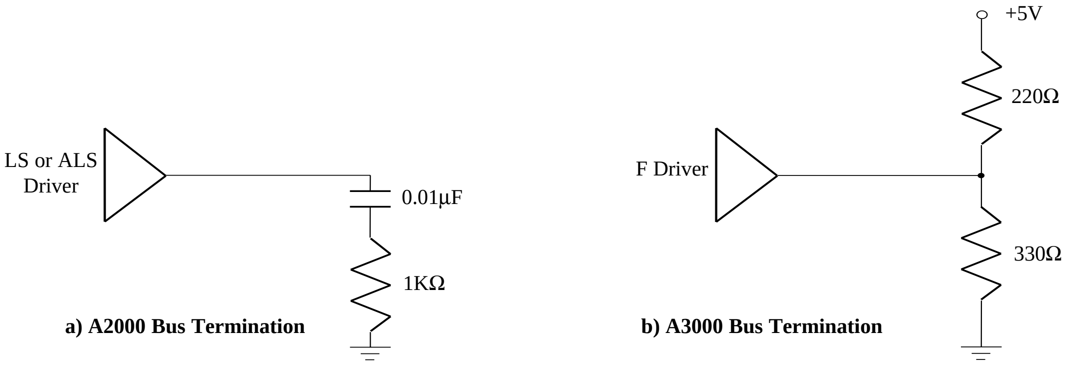


Figure 2-1: A2000 vs. A3000 Bus Termination

2.1.8 DMA Latency and Overlap

Zorro II bus masters in a Zorro III backplane will, in many cases, receive a bus grant much sooner than they would in a standard Zorro II backplane. Additionally, in some cases, expansion bus cycles will overlap local bus cycles. The latency incurred on the Zorro II bus during heavy custom chip activity has been greatly reduced for any Zorro III bus master. This should be transparent to the card in question, though it's a good thing to be aware of.

2.1.9 Power Supply Differences

The Zorro II bus is defined as supplying +5VDC @ 2 Amps to each slot, with one slot per backplane supplying 5.0VDC @ 4.0 Amps. The Zorro III bus only provides the 5.0VDC @ 2.0 Amps for each slot.

2.2 Bus Architecture

The Zorro II bus is a simple extention of the 68000 processor bus. Those without a good knowledge of the 68000 local bus will find *The 68000 User's Manual* from Motorola an excellent reference for many Zorro II issues. The *A500/A2000 Technical Reference Manual* from Commodore-Amiga is also required reading for any Zorro II design issues, as it includes a complete description of all the Commodore-Amiga details that aren't part of the 68000 specification.

The basic Zorro II bus is a buffered version of the 68000 processor bus, physically provided on a 100 pin one-piece connector. The bus is 16 bits wide, and provides 24 bits of addressing information. A bus cycle looks exactly like a 68000 bus cycle. The cycle is defined by an address strobe, terminated by a data transfer strobe, and qualified by a read/write strobe, some memory space qualifiers, and one or two byte selection strobes. The basic bus cycle runs for a total of four cycles of a 7.16MHz clock, though it can be extended to add wait states when required.

The Zorro II bus adds a number of features to the basic 68000 CPU bus. It supplies some Amiga system signals that are useful for expansion card designs, such as many of the Amiga system clocks. The bus provides a default data transfer signal, which expansion cards can easily use and modify rather than go to the trouble of creating their own. It provides a number of discrete interrupt lines which are mixed to provide the 68000 with its standard encoded interrupts. The 68000 bus arbitration protocol is used to allow multiple bus masters; arbitration of the bus requests are managed by the Zorro II bus controller to avoid contention between multiple masters. And of course the bus supplies a number of supply voltages for powering cards.

A powerful aspect of the Zorro II bus is its convention for automatically configuring expansion cards, AUTOCONFIG®. On system powerup, the system software interrogates each board to determine what kind of board is installed and how much memory space it needs on the bus. The software then tells each board where to reside in memory. The bus provides hardware lines to allow the boards to be configured in a daisy chained fashion regardless of which slots they occupy and to prevent damage to boards if accidently configured to reside at the same memory location. Firmware standards also permit software to autoboot or autoinitialize any board, to match soft-loaded device drivers with individual boards, and to link memory boards into the appropriate system memory lists.

2.3 Signal Description

The Zorro II bus can be broken down into various logical signal groups. Some of these groups are unchanged in the Zorro III bus modes, others are drastically different. This section makes note of the original Zorro II name for each signal and the current Zorro III physical pin name for each signal, where different. Some of this information will be repeated in the Zorro III chapters, where appropriate; nothing in this chapter is considered critical to understanding the Zorro III bus, but it is useful. As previously mentioned, the A2000 bus signals unsupported by the Zorro II specification have been deleted from the Zorro III specification and the A3000 implementation of Zorro III; this section will, however, document those signals for reference purposes. Please see Appendix A for a complete list with pin numbers of the various logical signals that appear on the physical bus during the different phases of the Zorro II and Zorro III bus cycles.

2.3.1 Power Connections

The Zorro III expansion bus provides several different voltages designed to supply expansion devices. There are no changes here that affect Zorro II cards.

Digital Ground (Ground)

This is the digital supply ground used by all expansion cards as the return path for all expansion supplies.

Main Supply (+5VDC)

This is the main power supply for all expansion cards, and it is capable of sourcing large currents; each expansion slot can draw up to 2.0 Amps @ +5VDC. The extra power for one card in any backplane drawing up to 4.0 Amps @ +5VDC is no longer supported.

Negative Supply (-5VDC)

This is a negative version of the main supply, for small current loads only. There is no maximum load specified for the Zorro II bus on a per-slot basis; the A2000 implementation specifies 0.3 Amps @ -5VDC for the entire system.

High Voltage Supply (+12VDC)

This is a higher voltage supply, useful for communications cards and other devices requiring greater than digital voltage levels. This is intended for relatively small current loads only. There is no maximum load specified for the Zorro II bus on a per-slot basis; the A2000 implementation specifies 8.0 Amps @ +12VDC for the entire system, most of which is normally devoted to floppy and hard disk drive motors, not slots.

Negative High Supply (-12VDC)

Negative version of the high voltage supply, also commonly used in communications applications, and similarly intended for small loads only. There is no maximum load specified for the Zorro II bus on a per-slot basis; the A2000 implementation specifies 0.3 Amps @ -12VDC for the entire.

2.3.2 Clock Signals

The Zorro III expansion bus provides clock signals for expansion boards. These clocks are for synchronous Zorro II designs and for other synchronous activity such as bus arbitration. While originally based on Amiga local bus clocks, these have no guaranteed relationship to any local bus activity in newer Amiga computers, but are maintained in Amiga computers as part of the expansion bus specifiation. The relationship between these clocks is illustrated in *Figure 2-2.*

/C1 Clock

This is a 3.58 MHz clock (3.55 MHz on PAL systems) that’s synched to the falling edge of the 7M system clock.

/C3 Clock

This is a 3.58 MHz clock (3.55 MHz on PAL systems) that's synched to the rising edge of the 7M system clock.

CDAC Clock

This is a 7.16 MHz system clock (7.09 MHz on PAL systems) which trails the 7M clock by 90° (approximately 35ns).

E Clock

This is the 68000 generated "E" clock, used for 6800 family peripherals driven by "E" and 6502 peripherals driven by Φ_2. This clock is four 7M clocks high, six clocks low, as per the 68000 spec. Note that the bus does not support the rest of the 68000's 6800/6502 compatible interface; there may be better ways to clock such devices.

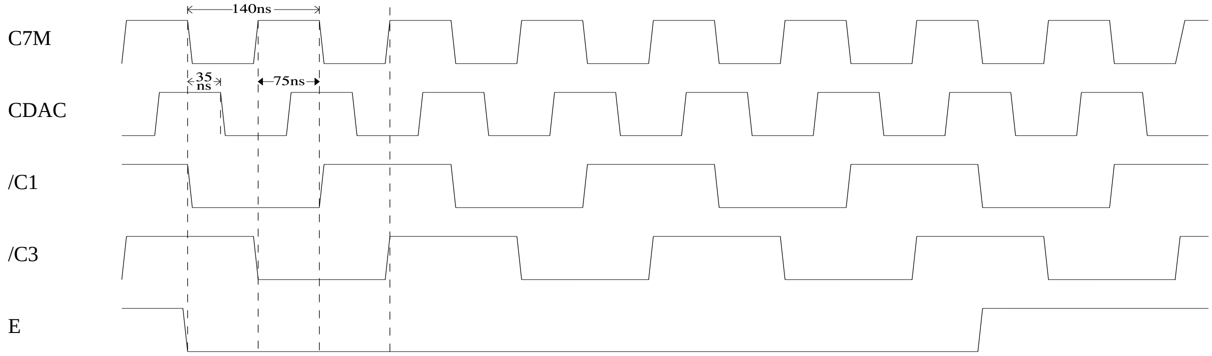


Figure 2-2: Expansion Bus Clocks

7M Clock

This is the 7.16 MHz system clock (7.09 MHz on PAL systems). This clock forms the basis for all Zorro II/68000 compatible activity, and for various other system functions, such as bus arbitration.

2.3.3 System Control Signals

The signals in this group are available for various types of system control; most of these have an immediate or near immediate effect on expansion cards and/or the system CPU itself.

Bus Error (/BERR)

This is a general indicator of a bus fault condition. Any expansion card capable of detecting a hardware error relating directly to that card can assert /BERR when that bus error condition is detected, especially any sort of harmful hardware error condition. This signal is the strongest possible indicator of a bad situation, as it causes all PICs to get off the bus, and will usually generate a level 2 exception on the host CPU. For any condition that can be handled in software and doesn't pose an immediate threat to hardware, notification via a standard processor interrupt is the better choice. The bus controller will drive /BERR in the event of a detected bus collision or DMA error (an attempt by a bus master to access local bus resources it doesn't have valid access permission for). All cards must monitor /BERR and be prepared to tri-state all of their on-bus output buffers whenever this signal is asserted. The current bus master should, if possible, retry the bus cycle after /BERR is negated unless conditions warrant otherwise. Since any number of devices may assert /BERR, and all bus cards must monitor it, any device that drives /BERR must drive with an open collector or similar device capable of sinking at least 12ma, and any device that monitors /BERR should place a minimal load on it (1 "F" type load or less). This signal is pulled high by a passive backplane resistor.

System Reset (/RST, /BUSRST) ≡ (/RESET, /IORST) for Zorro III

The bus supplies two versions of the system reset signal. The /RST signal is bidirectional and unbuffered, allowing an expansion card to hard reset the system. It should only be used by boards that need this reset capability, and is driven only by an open collector or similar device. The /BUSRST signal is a buffered output-only version of the reset signal that should be used as the normal reset input to boards not concerned with resetting the system on their own. All expansion devices are required to reset their autoconfiguration logic when /BUSRST is asserted. This signal is pulled high by a passive backplane resistor.

System Halt (/HLT)

This signal is similar to the 68000 processor halt signal, and is driven by a PIC with an open-collector or similar gate only. Its main use is to indicate a full-system reset. Based on the 68000 conventions, an I/O-only reset, such as initiated by the 680x0 RESET instruction, will drive only /RST and /BUSRST on the bus. A full-system reset, such as a powerup reset or a keyboard reset, drives /HLT low as well. PICs that wish to reset the system CPU as well as the bus and I/O devices drive /RST and /HLT, some bus devices such as processor cards may internally reset only on full-system resets. This signal is pulled high by a passive backplane resistor.

System Interrupts

Six of the decoded, level sensitive 680x0 interrupt inputs were originally available on the expansion bus, and these are labelled as /INT$_2$, /INT$_6$, /EINT$_1$, /EINT$_4$, /EINT$_5$, /EINT$_7$ on the Zorro II bus. Only the /INT$_2$ and /INT$_6$ interrupt inputs are actually supported by Commodore-Amiga as part of the Zorro II specification; the A2000 hardware did not provide the to software the required support mechanisms for the safe use of these lines. Each of these interrupt lines are shared by wired ORing, thus each line must be driven by an open-collector or equivalent output type, and all are pulled high by passive backplane resistors.

2.3.4 Slot Control Signals

This group of signals is responsible for the control of things that happen between expansion slots.

Slave (/SLAVE$_N$)

Each slot has its own /SLAVE output, driven actively, all of which go into the collision detect circuitry. The "N" refers to the expansion slot number of the particular /SLAVE signal. Whenever a Zorro II PIC is responding to an address on the bus, it must assert its /SLAVE output within 35ns of /AS asserted. The /SLAVE output must be negated at the end of a cycle within 50ns of /AS negated. Late /SLAVE assertion on a Zorro II bus can result in loss of data setup times and other problems. A late /SLAVE negation for Zorro II cards can cause a collision to be detected on the following cycle. While the Zorro III sloppy cycle logic eliminates this fatal condition, late /SLAVE negation can nonetheless slow system performance unnecessarily. If more than one /SLAVE output occurs for the same address, or if a PIC asserts its /SLAVE output for an address reserved by the local bus, a collision is registered and results in /BERR being asserted.

Configuration Chain (/CFGIN$_N$, /CFGOUT$_N$)

The slot configuration mechanism uses the bus signals /CFGOUT$_N$ and /CFGIN$_N$, where "N" refers to the expansion slot number. Each slot has its own version of each, which make up the configuration chain between Slots. Each subsequent /CFGIN is a result of all previous /CFGOUTs, going from slot 0 to the last slot on the expansion bus. During the AUTOCONFIG® process, an unconfigured Zorro PIC responds to the 64K address space starting at $00E80000 if its /CFGIN signal is asserted. All unconfigured PICs start up with /CFGOUT negated. When configured, or told to "shut up", a PIC will assert its /CFGOUT, which results in the /CFGIN of the next slot being asserted. The backplane passes on the state of the previous /CFGOUT to the next /CFGIN for any slot not occupied by a PIC, so there’s no need to sequentially populate the expansion bus slots.

Data Output Enable (DOE)

This signal is used by an expansion card to enable the buffers on the data bus. The main Zorro II use of this line is to keep PICs from driving data on the bus until any other device is completely off the bus and the bus buffers are pointing in the correct direction. This prevents any contention on the data bus.

2.3.5 DMA Control Signals

There are various signals on the expansion bus that coordinate the arbitration of bus masters. Native Zorro III bus masters use some of the same logical signals, but their arbitration protocol is considerably different.

PIC is DMA Owner (/OWN)

This signal is asserted by an expansion bus DMA device when it becomes bus master. This output is to be treated as a wired-OR output between all expansion slots, any of which may have a PIC signalling bus mastership. Thus, this should be driven with an open-collector or similar output by any PIC using it. This signal is the main basis for data direction calculations between the local and expansion busses, and is pulled up by a backplane resistor.

Slot Specific Bus Arbitration (/BR$_N$, /BG$_N$)

These are the slot-specific /BRN and /BGN signals, where "N" refers to the expansion slot number. The bus request from each board is taken in by the bus controller and ultimately used to take over the system from 680x0 on the local bus. The bus controller eventually returns one bus grant to the winner among all requesting PICs. From the point of view of the individual PIC, the protocol is very similar to that of the 68000 arbitration mechanism. The PIC asserts /BRN on the rising edge of 7M; some time later, /BGN is returned on the falling edge of 7M. The PIC waits for all bus activity to finish, asserts /OWN followed by /BGACK, then negates /BRN, assuming bus mastership. It retains mastership until it negates /BGACK followed by /OWN.

Bus Grant Acknowledge (/BGACK)

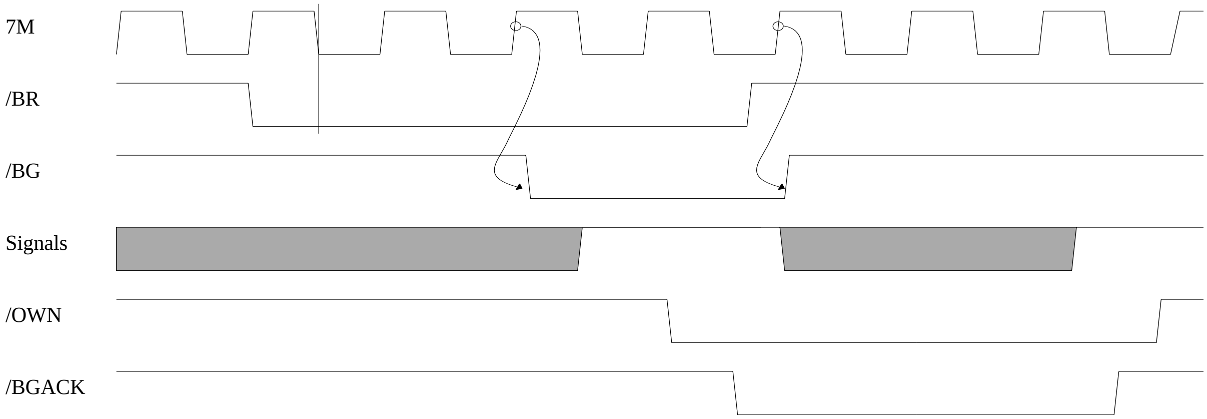


Figure 2-3: Zorro II Bus Arbitration

Any Zorro II PIC that receives a bus grant asserts this signal as long as it maintains bus mastery. This signal may never be asserted until the bus grant has been received, /AS is negated, /DTACK is negated, and /BGACK itself is negated, indicating that all other potential bus masters have relinquished the bus. This output is driven as a wired-OR output, so all PICs must drive it with an open collector or equivalent device, and a passive pullup is supplied by the backplane.

Bus Want/Clear (/GBG) ≡ (/BCLR) for Zorro III

This signal is asserted by the bus controller to indicate that a PIC wants to master the bus. A bus master assumes that the host CPU wants the bus, and that any time wasted as master is stealing time from the CPU. To avoid such waste, a master should use cache or FIFO to grab slow-coming data, and then transfer it all at once. /BCLR is asserted to indicate that additionally, another PIC wants the bus, and the current bus master should get off as soon as possible. This signal is equivalent to /GBG on the A2000 bus.

2.3.6 Addressing and Control Signals

These signals are various items used for the addressing of devices in Zorro II mode by the local bus and any expansion DMA devices. Most of these signals are very much like 68000 generated bus signals bi-directionally buffered to allow any DMA device on the bus to drive the local bus when such a device is the bus master.

Read Enable (READ)

This is the read enable for the bus, which is equivalent to the 68000's R/W output. READ asserted during a bus cycle indicates a read cycle, READ negated indicates a write cycle. Note that this signal may become valid in a cycle earlier than a 68000 R/W line would, but it remains valid at least as long at the cycle's end.

Address Bus (A_1-A_{23})

This is logically equivalent to the 68000's address bus, providing 16 megabytes of address space, although much of that space is not assigned to the expansion bus (see the memory map in *Figure 1-1*).

Address Strobe (/AS) ≡ (/CCS) for Zorro III

This is equivalent to the 68000 /AS, called /CCS, for Compatibility Cycle Strobe, in the Zorro III nomenclature. The falling edge of this strobe indicates that addresses are valid, the READ line is valid, and a Zorro II cycle is starting. The rising edge signals the end of a Zorro II bus cycle, signaling the current slave to negate all slave-driven signals as quickly as possible. Note that /CCS, like /AS, can stay asserted during a read-modify-write access over multiple cycle boundaries. To correctly support such cycles, a device must consider both the state of /CCS and the state of the data strobes. Many current Zorro II cards don't correctly support this 680x0 style bus lock.

Data Bus (D_0-D_{15})

This is a buffered version of the 680x0 data bus, providing 16 bits of data accessible by word or either byte. A PIC uses the DOE signal to determine when the bus is to be driven on reads, and the data strobes to determine when data is valid on writes.

Data Strobes (/UDS, /LDS) ≡ (/DS_3, /DS_2) for Zorro III

These strobes fall on data valid during writes, and indicate byte select for both reads and writes. The lower strobe is used for the lower byte (even byte), the upper strobe is used for the upper byte (odd byte). There is one slight difference between these lines and the 68000 data strobes. On reads of Zorro II memory space, both /DS_3 and /DS_2 will be asserted, no matter what the actual size of the requested transfer is. This is required to support caching of the Zorro II memory space. For Zorro II I/O space, these strobes indicate the actual, requested byte enables, just as would a 68000 bus master.

Data Transfer Acknowledge (/DTACK)

This signal is used to normally terminate both Zorro bus cycles. For Zorro II modes, it is equivalent to the 68000's Data Transfer Acknowledge input. It can be asserted by the bus slave during a Zorro II cycle at any time, but won't be sampled by the bus master until the falling edge of the S_4 state on the bus. Data will subsequently be latched on the S_6 falling edge after this, and the cycle terminated with /AS negated during S_7. If a Zorro II slave does nothing, this /DTACK will be driven by the bus controller with no wait states, making the bus essentially a 4 cycle synchronous bus. Any slow device on the bus that needs wait states has two options. It can modify the automatic /DTACK negating XRDY to hold off /DTACK. Alternately, it may assert /OVR to inhibit the bus controller's generation of /DTACK, allowing the slave to create its own /DTACK. Any /DTACK supplied by a slave must be driven with an open-collector or similar type output; the backplane provides a passive pullup.

Processor Status (FC_0-FC_2)

These signals are the cycle type or memory space bits, equivalent for the most part with the 68000 Processor Status outputs. They function mainly as extensions to the bus address, indicating which type of access is taking place. For Zorro II devices, any use of these lines must be gated with /BGACK, since they are not driven valid by Zorro II bus masters. However, when operating on the Zorro III backplane, Zorro II masters that don't drive the function codes will be seen generating an FC_1 = 0, which results in a valid memory access. Zorro II cycles are not generated for invalid memory spaces when the CPU is the bus master.

/DTACK Override (/OVR)

This signal is driven by a Zorro II slave to allow that slave to prevent the bus controller's /DTACK generation. This allows the slave to generate its own /DTACK. The previous use of this line to disable motherboard memory mapping, which was unsupported on the A2000 expansion bus, has now been completely removed. The use of XDRY or /OVR in combination with /DTACK is completely up to the board designer -- both methods are equally valid ways for a slave to delay /DTACK. In Zorro III mode, this pin is used for something completely different.

External Ready (XRDY)

This active high signal allows a slave to delay the bus controller's assertion of /DTACK, in order to add wait states. XRDY must be negated within 60ns of the bus master's assertion of /AS, and it will remain negated until the slave wants /DTACK. The /DTACK signal will be asserted by the bus controller shortly following the assertion of XRDY, providing the bus cycle is a S_4 or later. XRDY is a wired-OR from all PICs, and as such, must be driven by an open collector or equivalent output. In Zorro III mode, this pin is used for something completely different.

CHAPTER 3

BUS ARCHITECTURE

"We follow in the steps of our ancestory, and that cannot be broken."
-Midnight Oil

While the Zorro II bus design was based in a large part on an already existing bus cycle, the 68000 cycle, the Zorro III bus design had a much different set of preconditions. It is not modeled after any particular CPU specific bus protocol, but instead it's a logical outgrowth of both the need to support Zorro II cards on the same bus and the need to achieve various modern feature and preformance goals. These goals were summarized in Chapter 1, now they'll be covered in greater detail here.

3.1 Basic Zorro III Bus Cycles

The basic Zorro III bus cycle is a multiplexed address/data cycle which supplies a full 32 bits worth of address and data per simple cycle. The cycle is a fully asynchronous cycle. The bus master for a given cycle supplies strobes to indicate when address is valid, write data is valid, and read data may be driven. In return, the bus slave for a cycle supplies a strobe to indicate that it is responding to a bus address, and a strobe to indicate that it is done with the bus data for a write cycle, or has supplied valid bus data for a read cycle. The minimum theoretical bus speed is governed only by setup and hold time requirements for the various bus signals. Actual bus speeds is always a function of the bus master and bus slave active for a given cycle. This is considerably different than the way things work under the Zorro II bus, and for several good reasons, which are explained below.

3.1.1 Design Goals

For any computer bus, there are two basic possibilities concerning the fundamental operation of the bus; it's either synchronous or asynchronous. The difference is simple -- the synchronous bus is ultimately tied to a clock of some sort, while the asynchronous bus has no defined relationship to any clock signal. While Motorola specifies the 68000 bus cycle as an asynchronous cycle, they're really refering to the fact that most 68000 inputs are internally synchronized with the bus clock, and therefore, synchronous setup times on the bus do not have to be met to avoid metastability. But the 68000 bus, and the Zorro II bus by extension, are synchronous buses, based on a single bus clock (called E7M on the Zorro II bus). Most Zorro II signals are asserted relative to an edge of the bus clock, and most Zorro II inputs are sampled on an edge of the bus clock. The minimum Zorro II cycle is four bus clocks long, and every wait state added, regardless of the method, will result in a single additional bus clock wait, regardless of the asynchronous appearance of the termination and wait signals on the Zorro II bus.

The Zorro III bus is a fully asynchronous bus, in that all bus events are driven by strobes, and there is no reference clock. The choice of an asynchronous versus a synchronous bus design is governed by the intended application of the bus. Synchronous designs are preferred when a CPU and a memory system (eg, master and slave) can be very tightly coupled to each other. Such designs generally require a tight adherence to timing based on the specific CPU. This is optimal for tightly coupled systems, such the fast memory on the A3000 local bus. Synchronous designs can also be easier to do accurately, as the designer can use clock edges for scheduling events, and there's never any need to waste time in synchronizers to achieve a reliable design.

The design goals for an expansion bus are considerably different. While a fast memory circuit on a system motherboard can change for every new and better design, it's not feasable to require redesign of any significant number of expansion cards every time an improved motherboard design is created. And while a synchronous transfer can be optimal for matched clocks, it can be very inefficient for mismatched CPU and expansion clocks, as synchronizer delays must be introduced for any reliable operation. The A3000 project started with the need to support CPU systems at 16MHz and at 25MHz, and it's obvious that the growth of CPU clock speed will be here for some time to come. Zorro III cards are based on asynchronous handshaking between master and slave in both directions. This means that, as long as masters and slaves manage their own needs, any slave can work with any master. But as masters and slaves improve with technology, bus transfer speeds can automatically increase, without rendering any slower cards obsolete. The Zorro III bus attempts to address the needs of device expansion as much as the needs of memory expansion.

3.1.2 Simple Bus Cycle Operation

The normal Zorro III bus cycle is quite different than the Zorro II bus in many respects. *Figure 3-1* shows the basic cycle. There is no bus clock visible on the expansion bus; the standard Zorro II clocks are still active during Zorro III cycles, but they have no relationship to the Zorro II bus cycle. Every bus event is based on a relationship to a particular bus strobe, and strobes are alternately supplied by master and slave.

A Zorro III cycle begins when the bus master simultaneously drives addressing information on the address bus and memory space codes on the FC_N lines, quickly following that with the assertion of the Full Cycle Strobe, /FCS; this is called the *address phase* of the bus. Any active slaves will latch the bus address on the falling edge of /FCS, and the bus master will tri-state the addressing information very shortly after /FCS is asserted. It's necessary only to latch A_{31}-A_8; the low order A_7-A_2 addresses and FC_N codes are non-multiplexed.

As quickly as possible after /FCS is asserted, a slave device will respond to the bus address by asserting its $/SLAVE_N$ line, and possibly other special-purpose signals. The autoconfiguration process assigns a unique address range to each PIC base on its needs, just as on the Zorro II bus. Only one slave may respond to any given bus address; the bus controller will generate a /BERR signal if more than one slave responds to an address, or if a single slave responds to an address reserved for the local bus (this is called a bus collision, and should never happen in normal operation). Slaves don't usually respond to CPU memory space or other reserved memory space types, as indicated by the memory space code on the FC_N lines (see Chapter 4 for details)!

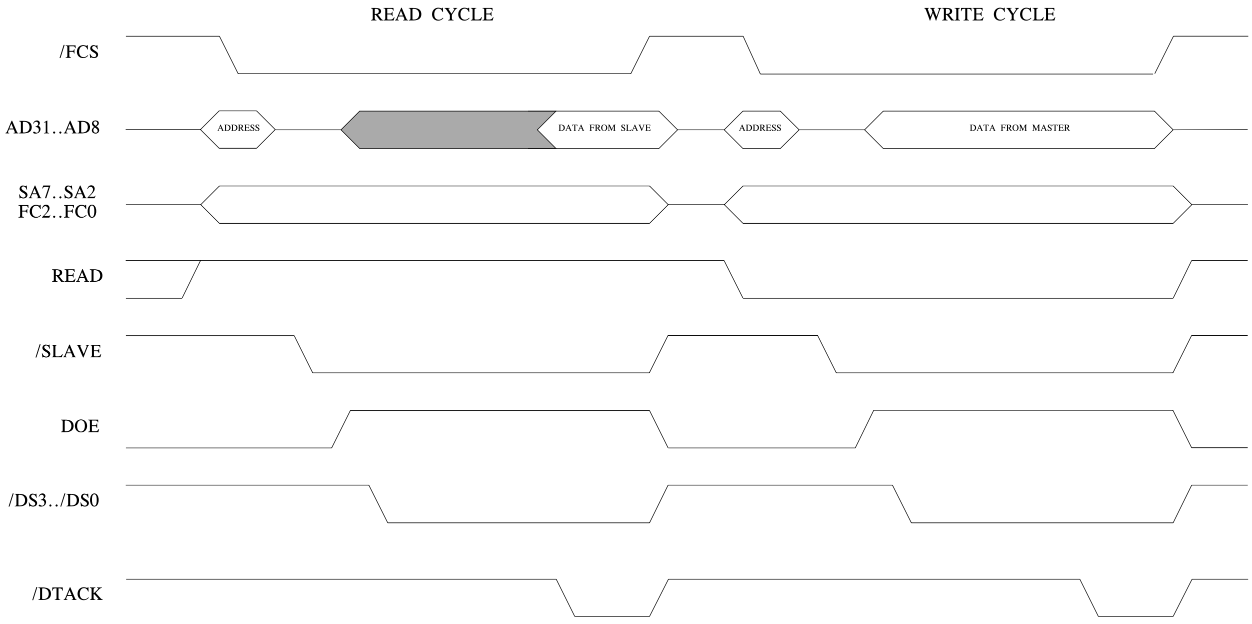


Figure 3-1: Basic Zorro III Cycles

The *data phase* is the next part of the cycle, and it's started when the bus master asserts DOE onto the bus, indicating that data operations can be started. The strobes are the same for both read and write cycles, but of course the data transfer direction is different.

For a read cycle, the bus master drives at least one of the data strobes $/DS_N$, indicating the physical transfer size requested (however, cachable slaves must always supply all 32 bits of data). The slave responds by driving data onto the bus, and then asserting /DTACK. The bus master then terminates the cycle by negating /FCS, at which point the slave will negate its $/SLAVE_N$ line and tri-state its data. The cycle is done at this point. There are a few actions that modify a cycle termination, those will be covered in later sections.

The write cycle starts out the same way, up until DOE is asserted. At this point, it's the master that must drive data onto the bus, and then assert at least one /DSN line to indicate to the slave that data is valid and which data bytes are being written. The slave has the data for its use until it terminates the cycle by asserting /DTACK, at which point the master can negate /FCS and tri-state its data at any point. For maximum bus bandwidth, the slave can latch data on the falling edge of the logically ORed data strobes; the bus master doesn't sample /DTACK until after the data strobes are asserted, so a slave can actually assert /DTACK any time after /FCS.

3.2 Advanced Mode Support Logic

The Zorro III bus provides support for some more advanced things that weren't generally handled correctly on the Zorro II bus. Amiga computers have traditionally been supporting things that the more mainstream personal computers haven't. High speed DMA transfers and expansion coprocessors such as the Bridge Cards have been with the Amiga since the early days, and high performance main system CPUs with cache memory are now becoming common. The Zorro II bus never properly or easily supported such devices; the Zorro III bus attempts to make support of cache and coprocessor both possible and relatively straightforward. Other new features are covered in later sections.

3.2.1 Bus Locking

The first advanced modification of the basic bus cycle is bus locking, via the /LOCK signal. Bus locking is a hardware convention that allows a bus master to guarantee several cycles will be atomic on the bus. This is necessary to support the sharing of special "mail-box" memory between a bus master and an alternate PIC-based processor; Bridge Cards are an example of this kind of device. The Zorro II bus itself supports bus locking via the 68000 convention. However, the 68000 style of bus locking is often difficult to implement, and support for it was often ignored in Zorro II designs, especially those not directly concerned with multiprocessor support.

The Zorro III mechanism involves no change to the basic bus cycle, other than the monitoring of this /LOCK signal, and as such is much more reasonable to support. The /LOCK signal is asserted by a bus master at address time and maintained across cycles to lock out shared memory coprocessors, allowing hardware backed semaphores to easily be used between such coprocessors. We expect multiprocessing will be a greater concern on the Zorro III bus than it is at present; video coprocessors, RISC devices, and special purpose processors for image processing or mathematics should find a comfortable home on the Zorro III bus.

3.2.2 Cache Support

The other advanced cycle modifier on the Zorro III bus is the cache inhibit line, /CINH. On the Zorro II bus, there was originally no caching envisioned, and therefore no real support for caching of Zorro II PICs. First in the A2630 and later in the Zorro III bus's emulation of Zorro

II, conventions were adopted to permit caching of Zorro II cards. These conventions aren't perfect; MMU tables will sometimes have to supplant this geographic mapping. While Zorro III doesn't have any cache consistency mechanisms for managing caches between several caching bus masters, it does allow cards that absolutely must not be cached to assert a cache inhibit line, /CINH, on a per-cycle basis (asserted at slave time by a responding slave). This cache management is basically the lowest level of a cache management system, mainly useful for support of I/O and other devices that shouldn't be cached. Software will be required for the higher levels of cache management.

3.3 Multiple Transfer Cycles

The multiplexed address/data design of the Zorro III bus has some definite advantages. It allows Zorro III cards to use the same 100 pin connector as the Zorro II cards, which results in every bus slot being a 32 bit slot, even if there's an alternate connector in-line with any or all of the system slots; current alternate connectors include Amiga Video and PC-AT (now sometimes called ISA, for *Industry Standard Architecture*, now that it's basically beyond the control of IBM) compatible connectors. This design also makes implementation of the bus controller for a system such as the A3000 simpler. And it can result in lower cost for Zorro III PICs in many cases.

The main disadvantage of the multiplexed bus is that the multiplexing can waste time. The address access time is the same for multiplexed and non-multiplexed buses, but because of the multiplexing time, Zorro III PICs must wait until *data time* to assert data, which places a fixed limit on how soon data can be valid. The Zorro III Multiple Transfer Cycle is a special mode designed to allow the bus to approach the speed of a non-multiplexed design. This mode is especially effective for high speed transfers between memory and I/O cards.

As the name implies, the Multiple Transfer Cycle is an extension of the basic full cycle that results in multiple 32 bit transfers. It starts with a normal full cycle *address phase* transaction, where the bus master drives the 32 bit address and asserts the /FCS signal. A master capable of supporting a Multiple Transfer Cycle will also assert /MTCR at the same time as /FCS. The slave latches the address and responds by asserting its /SLAVE$_N$ line. If the slave is capable of multiple transfers, it'll also assert /MTACK, indicating to the bus master that it's capable of this extended cycle form. If either /MTCR or /MTACK is negated for a cycle, that cycle will be a basic full cycle.

Assuming the multiple transfer handshake goes through, the multiple cycle continues to look similar to the basic cycle into the data phase. The bus master asserts DOE (possibly with write data) and the appropriate /DS$_N$, then the slave responds with /DTACK (possibly with read data at the same time), just as usual. Following this, however, the cycle's character changes. Instead of terminating the cycle by negating /FCS, /DS$_N$, and DOE, the master negates /DS$_N$ and /MTCR, but maintains /FCS and DOE. The slave continues to assert /SLAVE$_N$, and the bus goes into what's called a *short cycle.*

The short cycle begins with the bus master driving the low order address lines A_7-A_2; these are

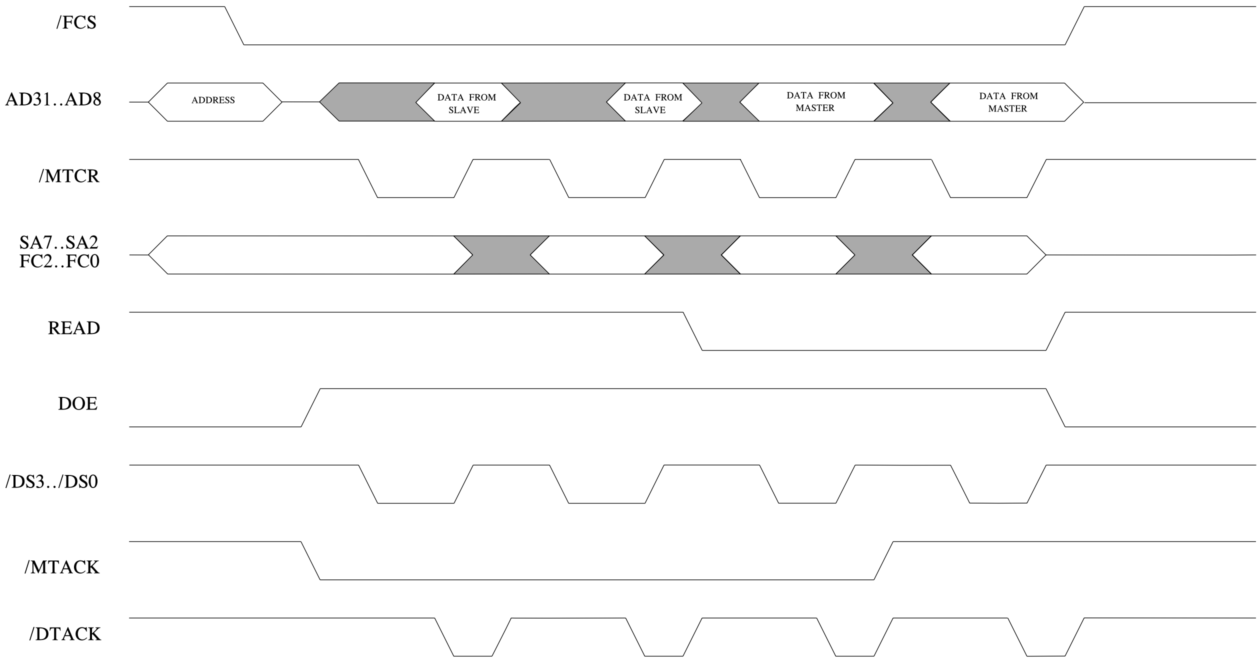


Figure 3-2: Multiple TransferCycles

the non-multiplexed addresses and can change without a new *address phase* being required (this is essentially a page mode, fully random accesses on this 256 byte page). The READ line may also change at this time. The master will then assert /MTCR to indicate to the slave that the short cycle is starting. For reads, the appropriate /DSN are asserted simultaneously with /MTCR, for writes, data and /DSN are asserted slightly after /MTCR. The slave will supply data for reads, then assert /DTACK, and the bus will will terminate the short cycle and start into either another short cycle or a full cycle, depending on the multiple cycle handshaking that has taken place.

The question of whether a subsequent cycle will be a full cycle or a short cycle is answered by multiple cycle arbitration. If the master can't sustain another short cycle, it will negate /FCS and DOE along with /MTCR at the end of the current short cycle, terminating the full cycle as well. The master always samples the state of /MTACK on the falling edge of /MTCR. If a slave can't support additional short cycles, it negates /MTACK one short cycle ahead of time. On the following short cycle, the bus master will see that no more short cycles can be handled by the slave, and fully terminate the multiple transfer cycle once this last short cycle is done.

PICs aren't absolutely required to support Multiple Transfer Cycles, though it is a highly recommended feature, especially for memory boards. And of course, all PICs must act intelligently about such cycles on the bus; a card doesn't request or acknowledge any Multiple Transfer Cycle it can't support.

3.4 Quick Bus Arbitration

The Zorro II bus does an adequate job of supporting multiple bus masters, and the Zorro III bus extends this somewhat by introducing fair arbitration to Zorro II cards. However, some desirable features cannot be added directly to the Zorro II arbitration protocol. Specifically, Zorro III bus arbitration is much faster than the Zorro II style, it prohibits bus hogging that's possible under the Zorro II protocol, and it supports intelligent bus load balancing.

Load balancing requires a bit of explanation. A good analogy is to that of software multitasking; there, an operating system attempts to slice up CPU time between all tasks that need such time; here, a bus controller attempts to slice up bus time between all masters that need such time. With preemptive multitasking such as in the Amiga and UNIX OSs, equal CPU time can be granted to every task (possibly modified by priority levels), and such scheduling is completely under control of the OS; no task can hog the CPU time at the expense of all others. An alternate multitasking scheme is a popular add-on to some originally non-multasking operating systems lately. In this scheme, each task has the CPU until it decides to give up the CPU, basically making the effectiveness of the CPU sharing at the mercy of each task. This is exactly the same situation with masters on the Zorro II bus. The Zorro III arbitration mechanism attempts to make bus scheduling under the control of the bus controller, with masters each being scheduled on a cycle-by-cycle basis.

When a Zorro III PIC wants to master the bus, it *registers* with the bus controller. This tells the bus controller to include that PIC in its scheduling of the expansion bus. There may be any number of other PICs registered with the bus controller at any given time. The CPU is always scheduled expansion bus time, and other local bus devices, such as a hard disk contoller, may be registered from time to time.

Once registered, a PIC sits idle until it receives a *grant* from the bus controller. A grant is permission from the bus controller that allows the PIC to master the Zorro III bus for one full cycle. A PIC always gets one full cycle of bus time when given a grant, and assuming it stays registered, it may receive additional full cycles. Within the full cycle, the PIC may run any number of Multiple Transfer Cycles, assuming of course the responding slave supports such cycles. For multiprocessor support, a PIC will be granted multiple atomic full cycles if it locks the bus. This feature is only for support of hardware semaphores and other such multiprocessor needs; it is not intended as a means of bus hogging!

Figure 3-3 shows the basics of Zorro III bus arbitration, which is pretty simple. While it uses some of the same signals as the 680x0 inspired Zorro II bus arbitration mechanism, it has nothing to do with 680x0 bus arbitration; the /BRN and /BGN signals should be thought of as completely new signals. In order to register with the bus controller as a bus master, a PIC asserts its private /BRN strobe on the rising edge of the 7M clock, and negates it on the next rising edge. The bus controller will indicate mastership to a registered bus master by asserting its /BGN. Once granted the bus, the PIC drives only the standard cycle signals: addresses, /FCS, /EDSN, data, etc. in a full cycle. The bus controller manages the assertion of /OWN and /BGACK, which are important only for bus management and Zorro II support. While a scheduling scheme isn't part of this bus specification, the bus master will only be guaranteed one bus cycle at a time. The /BGN line is negated shortly after the master asserts /FCS unless the bus controller is planning to grant multiple full cycles to the master. The only thing that'll force the controller to grant

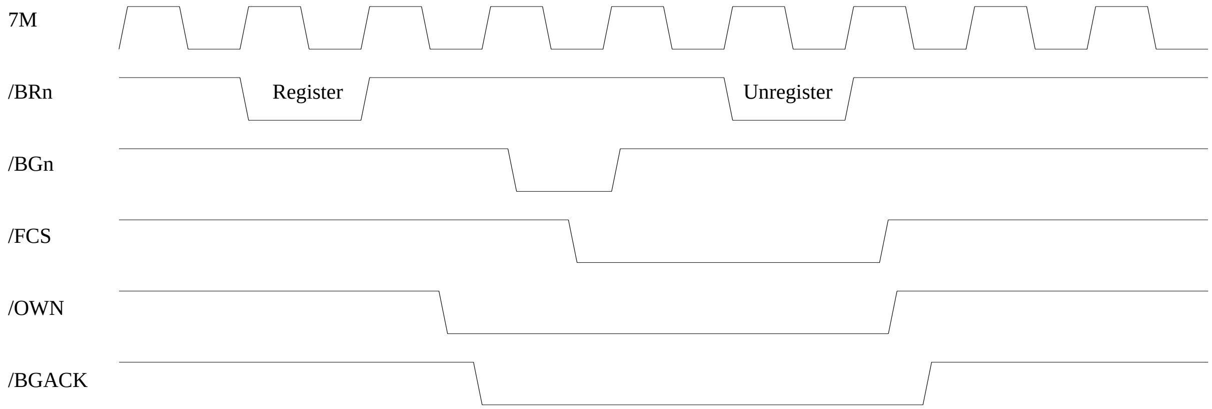


Figure 3-3: Zorro III Bus Arbitration

multiple full cycles is a locked bus. Any master that works better with multiple cycles, such as devices with buffers to empty into memory, should run a Multiple Transfer Cycle to transfer several longwords during the same full cycle. For this reason, slave cards are encouraged to support Multiple Transfer Cycles, even if they don't necessarily run any faster during them.

Once a registered bus master has no more work to do, it unregisters with the bus controller. This works just like registering -- the PIC asserts /BRN on the rise of 7M, then negates it on next rising 7M. This is best done during the last cycle the bus master requires on the bus. If a registered master gets a grant before unregistering and has no work to do, it can unregister without asserting /FCS, to give back the bus without runing a cycle. It's always far better to make sure that the master unregisters as quickly as possible. Bus timeout causes an automatic unregistering of the registered master that was granted that timed-out cycle; this guarantees that an inactive registered master can't drag down the system. If a master sees a /BERR during a cycle, it should terminate that cycle immediately and re-try the same cycle. If the retried cycle results in a /BERR as well, nothing more can be done in hardware; notification of the driver program is the usual recourse.

The bus controller may have to mix Zorro II style bus arbitration in with Zorro III arbitration, as Zorro II and Zorro III cards can be freely mixed in a backplane. Because of this, Multiple Transfer Cycles, and the self-timed nature of Zorro III cards, there's no way to guarantee the latency between bus grants for a Zorro III card. The bus controller does, however, make sure that all masters are fairly scheduled so that no starvation occurs, if at all possible. Zorro III cards must use Zorro III style bus arbitration; although current Zorro III backplanes can't differentiate between Zorro II and Zorro III cards when they request (other than by the request mechanism), it can't be assumed that a backplane will support Zorro III cycles with Zorro II mastering, or visa-versa.

3.5 Quick Interrupts

While the Zorro II bus has always supported shared interrupts, the Zorro III bus supports a mechanism wherein the interrupting PIC can supply its own vector. This has the potential to make such vectored interrupts much faster than conventional Zorro II chained interrupts, arbitrating the interrupting device in hardware instead of software.

A PIC supporting quick interrupts has on-board registers to store one or more vector numbers; the numbers are obtained from the OS by the device driver for the PIC, and the PIC/driver combination must be able to handle the situation in which no additional vectors are available. During system operation, this PIC will interrupt the system in the normal manner, by asserting one of the bus interrupt lines. This interrupt will cause an interrupt vector cycle to take place on the bus. This cycle arbitrates in hardware between all PICs asserting that interrupt, and it's a completely different type of Zorro III cycle, as illustrated in *Figure 3-4*.

The bus controller will start an interrupt vector cycle in response to an interrupt asserted by any PIC. This cycle starts with /FCS and /MTCR asserted, a FC code of 7 (CPU space), a CPU space cycle type, given by address lines A_{16}-A_{19}, of 15, and the interrupt number, which is on A_1-A_3 (A_1 is on the /LOCK line, as in Zorro II cycles). The interrupt numbers 2 and 6 are currently defined, corresponding to /INT$_2$ and /INT$_6$ respectively; all others are reserved for future use. At this point, called the *polling phas*e, any PIC that has asserted an interrupt and wants to supply a vector will decode the FC lines, the cycle type, match its interrupt number against the one on the bus, and assert /SLAVE$_N$ if a match occurs. Shortly thereafter, the /MTCR line is negated, and the slaves all negate /SLAVE$_N$. But the cycle doesn't end.

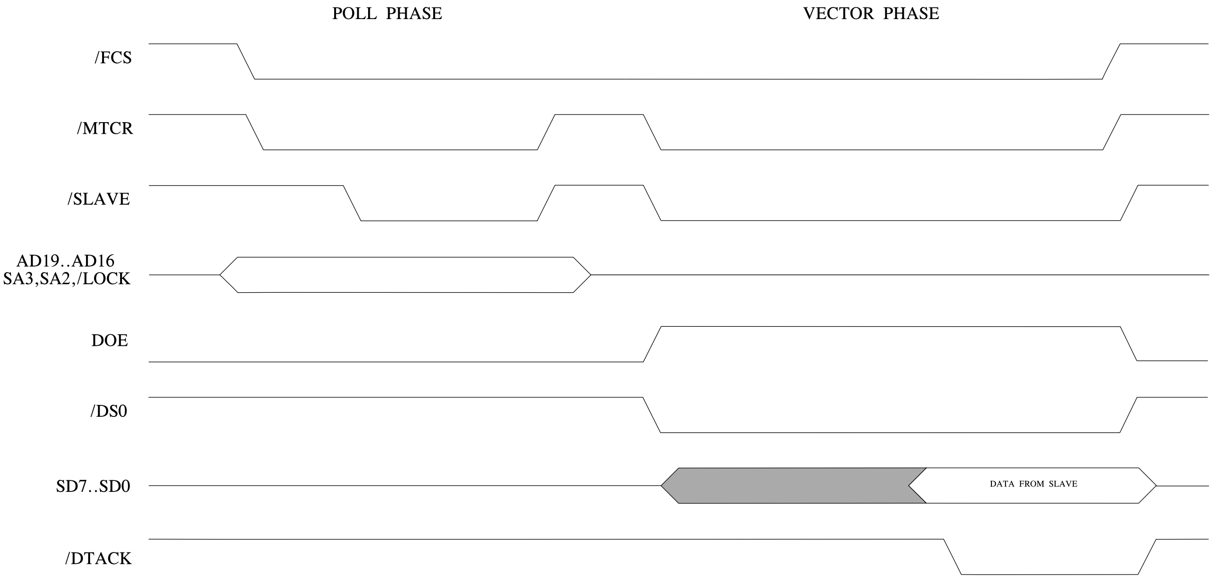


Figure 3-4: Interrupt Vector Cycle

The next step is called the *vector phase*. The bus controller asserts one /SLAVE$_N$ back to one of the interrupting PICs, along with /MTCR and /DS$_0$, but no addresses are supplied. That PIC will then assert its 8 bit vector onto the logical D_0-D_7 (physically AD_{15}-AD_8) of the 32 bit data bus and /DTACK, as quickly as possible, thus terminating the cycle. The speed here is very critical; an automatic autovector timeout will occur very quickly, as any actual waiting that's required for the quick interrupt vector is potentially delaying the autovector response for Zorro II style interrupts. A PIC stops driving its interrupt when it gets the response cycle; it must also be possible for this interrupt to be cleared in software (eg, the PIC must make choice of vectoring vs. autovectoring a software issue).

3.6 Compatibility with Zorro II Devices

As detailed in Chapter 2, the Zorro III bus supports a bus cycle mode very similar to the 68000-based Zorro II bus, and is expected to be compatible with all properly designed Zorro II PICs. As shown in *Figure 1-1*, Zorro II and Zorro III expansion spaces are geographically mapped on the Zorro III bus. The mapping logic resides on the bus, and operates on the bus address presented for any cycle. Every cycle starts out assuming a Zorro III cycle, but the mapping logic will inscribe a Zorro II cycle within the Zorro III cycle if the address range is right. *Figure 3-5* details the bus action for this mode.

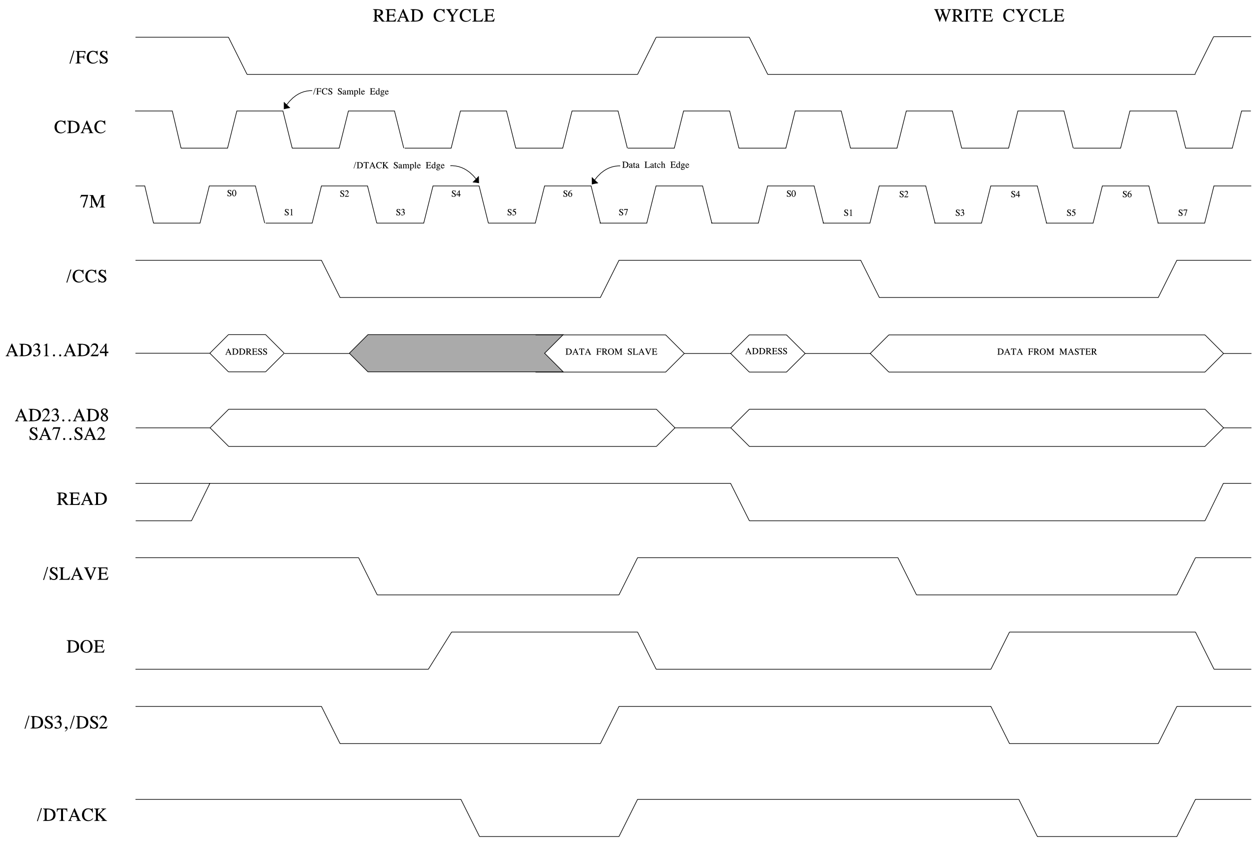


Figure 3-5: Zorro II Within Zorro III

The cycle starts out with the usual address phase activity; the bus master asserts /FCS after asserting the full 32 bit address onto the address bus. The bus decoder maps the bus address asynchronously and quickly, so that by the time /FCS is asserted, the memory space is determined. A Zorro II space access will cause A_8-A_{23} to remain asserted, rather than being tri-stated along with A_{24}-A_{31}, as the Zorro III cycle normally does. The bus controller synchs the asynchronous /FCS on the falling edge of CDAC, then drives /CCS (the /AS equivalent) out on the rising edge of 7M, based on that synched /FCS. For a read cycle, /DS$_3$ and/or /DS$_2$ (the

/UDS and /LDS replacements, respectively) would be asserted along with /CCS; write cycles see those lines asserted on the next rising edge of 7M, at S_4 time. The DOE line is also asserted at the start of S_4.

The bus controller starts to sample /DTACK on the falling edge of 7M between S_4 and S_5, adding wait states until /DTACK is encountered. As per Zorro II specs, the PIC need not create a /DTACK unless it needs that level of control; there are Zorro II signals to delay the controller-generated /DTACK, or take it over when necessary. The controller will drive its automatic /DTACK at the start of S_4, leaving plenty of time for the sampling to come at S_5. Once a /DTACK is encountered, cycle termination begins. The controller latches data on the falling 7M edge between S_6 and S_7, and also negates /CCS and the /DS_N at this time. Shortly thereafter, the controller negates /DTACK (when controlling it), DOE, and tri-states the data bus, getting ready for the next cycle.

CHAPTER 4

SIGNAL DESCRIPTION

"Pushing back the limits of human achievement, reaching for the stars,
that's not something we do. It's what we are."
-Michael Swaine

The signals detailed here are the Zorro III mode signals. While some of this information is the same in as the Zorro II signal description of Chaper 2, many like-seeming bus signals behave differently in Zorro III mode than Zorro II mode. These can be a very important differences; thus the complete set of signals is detailed here.

4.1 Power Connections

The expansion bus provides several different voltages designed to supply expansion devices. These are basically the same for the Zorro III bus as they were for the Zorro II bus, with the exception of one pin, and that the specification has been clarified a bit. Note that all Zorro III PICs must list their power consumption specifications.

Digital Ground (Ground)

This is the digital supply ground used by all expansion cards as the return path for all expansion supplies.

Main Supply (+5VDC)

This is the main power supply for all expansion cards, and it is capable of sourcing large currents; each PIC can draw up to 2.0 Amps @ +5VDC.

Negative Supply (-5VDC)

This is a negative version of the main supply, for small current loads only; each PIC can draw up to 60 mA @ -5VDC.

High Voltage Supply (+12VDC)

This is a higher voltage supply, useful for communications cards and other devices requiring greater that digital voltage levels. This is intended for relatively small current loads only; each PIC can draw up to 500mA @ +12VDC.

Negative High Supply (-12VDC)

Negative version of the high voltage supply, also used in communications applications, and similarly intended for small loads only; each PIC can draw up to 60 mA @ -12VDC.

4.2 Clock Signals

The expansion bus provides clock signals for expansion boards. The main use for these clocks on Zorro III cards is bus arbitration clocking. There is no relationship between any of these clocks and normal Zorro III bus activity. The relationship between these clocks is illustrated in *Figure 2-2.*

/C1 Clock

This is a 3.58 MHz clock (3.55 MHz on PAL systems) that's synched to the falling edge of the 7M system clock.

/C3 Clock

This is a 3.58 MHz clock (3.55 MHz on PAL systems) that's synched to the rising edge of the 7M system clock.

CDAC Clock

This is a 7.16 MHz system clock (7.09 MHz on PAL systems) which trails the 7M clock by 90° (approximately 35ns).

E Clock

This is the 68000 generated "E" clock, used for 6800 family peripherals driven by "E" and 6502 peripherals driven by Φ_2. This clock is four 7M clocks high, six clocks low, as per the 68000 spec.

7M Clock

This is the 7.16 MHz system clock (7.09 MHz on PAL systems). This clock drives the bus master registration mechanism for Zorro III bus masters.

4.3 System Control Signals

The signals in this group are available for various types of system control; most of these have an immediate or near immediate effect on expansion cards and/or the system CPU itself.

Hardware Bus Error/Interrupt (/BERR)

This is a general indicator of a bus fault or special condition of some kind. Any expansion card capable of detecting a hardware error relating directly to that card can assert /BERR when that bus error condition is detected, especially any sort of harmful hardware error condition. This signal is the strongest possible indicator of a bad situation, as it causes all PICs to get off the bus, and will usually generate a level 2 exception on the host CPU. For any condition that can be handled in software and doesn't pose an immediate threat to hardware, notification via a standard processor interrupt is the better choice. The bus controller will drive /BERR in the event of a detected bus collision or DMA error (an attempt by a bus master to access local bus resources it doesn't have valid access permission for). All cards must monitor /BERR and be prepared to tri-state all of their on-bus output buffers whenever this signal is asserted. An expansion bus master will attempt to retry a cycle aborted by a single /BERR and notify system software in the case of two subsequent /BERR results. Since any number of devices may assert /BERR, and all bus cards must monitor it, any device that drives /BERR must drive with an open collector or similar device, and any device that monitors /BERR should place a minimal load on it. This signal is pulled high by a passive backplane resistor.

Note that, especially for the slave device being addressed, that /BERR alone is not always necessaily an indication of a bus failure in the pure sense, but may indicate some other kind of unusual condition. Therefore, a device should still respond to the bus address, if otherwise appropriate, when a /BERR condition is indicated. It simply tri-states is bus buffers and other outputs, and waits for a change in the bus state. If the /BERR signal is negated with the cycle unterminated, the special condition has been resolved and the slave responds to the rest of the cycle as it normally would have. If the cycle is terminated by the bus master, the resolution of the special condition has indicated that the addressed slave is not needed, and so the cycle terminates without the slave being used.

System Reset (/RESET, /IORST)

The bus supplies two versions of the system reset signal. The /RESET signal is bidirectional and unbuffered, allowing an expansion card to hard reset the system. It should only be used by boards that need this reset capability, and is driven only by an open collector or similar device. The /IORST signal is a buffered output-only version of the reset signal that should be used as the normal reset input to boards not concerned with resetting the system on their own. All expansion devices are required to reset their autoconfiguration logic when /IORST is asserted. These signals are pulled high by passive backplane resistors.

System Halt (/HLT)

This signal is driven, along with /RESET, to assert a full-system reset. A full-system reset is asserted on a powerup reset or a keyboard reset; any PIC that needs to differentiate between full system and I/O reset should monitor /HLT and /IORST unless it also needs to drive a reset condition. This is driven with an open-collector output, or the equivalent, and pulled up by a backplane resistor.

System Interrupts

Two of the decoded, level sensitive 680x0 interrupt inputs are available on the expansion bus, and these are labelled as /INT$_2$ and /INT$_6$. Each of these interrupt lines is shared by wired

ORing, thus each line must be driven by an open-collector or equivalent output type. Zorro III interrupts can be handled Zorro II style, via autovectors and daisy-chained polling, or they can be vectored using the quick interrupt protocol described in Chapter 3. Zorro II and Zorro III systems originally provided /INT$_1$, /INT$_4$, /INT$_5$, and /INT$_7$ lines as well, but as these were never properly supportable by system software, they have been eliminated, those lines now considered reserved for future use in a Zorro III system.

4.4 Slot Control Signals

This group of signals is responsible for the control of things that happen between expansion slots.

Slave (/SLAVE$_N$)

Each slot has its own /SLAVE$_N$ output, driven actively, all of which go into the collision detect circuitry. The "N" refers to the expansion slot number of the particular /SLAVE signal. Whenever a Zorro III PIC is responding to an address on the bus, it must assert its /SLAVE$_N$ output very quickly. If more than one /SLAVE$_N$ output occurs for the same address, or if a PIC asserts its /SLAVE$_N$ output for an address reserved by the local bus, a collision is registered and the bus controller asserts /BERR. The bus controller will assert /SLAVE$_N$ back to the interrupting device selected during a Quick Interrupt cycle, so any device supporting Quick Interrupts must be capable of tri-stating its /SLAVE$_N$; all others can drive SLAVE$_N$ with a normal active output.

Configuration Chain (/CFGIN$_N$, /CFGOUT$_N$)

The slot configuration mechanism uses the bus signals /CFGOUT$_N$ and /CFGIN$_N$, where "N" refers to the slot number. Each slot has its own version of both signals, which make up the *configuration chain* between slots. Each subsequent /CFGIN$_N$ is a result of all previous /CFGOUTs, going from slot 0 to the last slot on the expansion bus. During the autoconfiguration process, an unconfigured Zorro III PIC responds to the 64K address space starting at either $00E80000 or $FF000000 if its /CFGIN$_N$ signal is asserted. All unconfigured PICs start up with /CFGOUT$_N$ negated. When configured, or told to "shut up", a PIC will assert its /CFGOUT$_N$, which results in the /CFGIN$_N$ of the next slot being asserted. Backplane logic automatically passes on the state of the previous /CFGOUT$_N$ to the next /CFGIN$_N$ for any slot not occupied by a PIC, so there’s no need to sequentially populate the expansion bus slots.

Backplane Type Sense (SenseZ3)

This line can be used by the PIC to determine the backplane type. It is grounded on a Zorro II backplane, but floating on a Zorro III backplane. The Zorro III PIC connects this signal to a 1K pullup resistor to generate a real logic level for this line. It’s possible, though more complicated, to build a Zorro III PIC that can actually run in Zorro II mode when in a Zorro II backplane. It’s hardly necessary or required to support this backward compatibility mechanism, and in many cases it’ll be inpractical. The Zorro III specification does require that this signal be used, at least, to shut the card down and pass /CFGIN to /CFGOUT when in a Zorro II backplane.

4.5 DMA Control Signals

There are various signals on the expansion bus that coordinate the arbitration of bus masters. Zorro II bus masters use some of the same logical signals, but their arbitration protocol is considerably different.

PIC is DMA Owner (/OWN)

This is asserted by the bus controller when a master is about to go on the bus and indicates that some master owns the bus. Zorro II bus masters drive this, and some Zorro III slaves may find a need to monitor it, or /BGACK, to determine who's the bus master. This is ordinarily not important to Zorro III PICs, and they may not drive this line.

Slot Specific Bus Arbitration (/BR$_N$, /BG$_N$)

These are the slot-specific /BR$_N$ and /BG$_N$ signals, where "N" refers to the expansion slot number. The bus request from each board is taken in by the bus controller and ultimately used to take over the system from the primary bus master, which is always the local master. Zorro III PICs toggle /BR$_N$ to register or unregister as a master with the bus controller. /BG$_N$ is asserted to one registered PIC at a time, on a cycle by cycle basis, to indicate to the PIC that it gets the bus for one full cycle.

Bus Grant Acknowledge (/BGACK)

Asserted by the bus controller when a master is about to go on the bus. As with /OWN, most Zorro III PICs ignore this signal, and none may drive it.

Bus Want/Clear (/BCLR)

This signal is asserted by the bus controller to indicate that a PIC wants to master the bus; Zorro III cards can use this to determine if any Zorro II bus requests are pending; Zorro III bus requests don't affect /BCLR.

4.6 Address and Related Control Signals

These signals are various items used for the addressing of devices in Zorro III mode by bus masters either on the bus or from the local bus. The bus controller translates local bus signals (68030 protocol on the A3000) into Zorro III signals; masters are responsible for creating the appropriate signals via their own bus control logic.

Read Enable (READ)

Read enable for the bus; READ is asserted by the bus master during a bus cycle to indicate a read cycle, READ is negated to indicate a write cycle. READ is asserted at address time, prior to /FCS, for a full cycle, and prior to /MTCR for a short cycle. READ stays valid throughout the cycle; no latching required.

Multiplexed Address Bus (A_8-A_{31})

These signals are driven by the bus master during address time, prior to the assertion of /FCS. Any responding slave must latch as many of these lines as it needs on the falling edge of

/FCS, as they're tri-stated very shortly after /FCS goes low. These addresses always include all configuration address bits for normal cycles, and the cycle type information for Quick Interrupt cycles.

Short Address Bus (A_2-A_7)

These signals are driven by the bus master during address time, prior to the assertion of /FCS, for full cycles, and prior to the assertion of /MTCR for short cycles. They stay valid for the entire full or short cycle, and as such do not need to be latched by responding slaves.

Memory Space (FC_0-FC_2)

The memory space bits are an extension to the bus address, indicating which type of access is taking place. Zorro III PICs must pay close attention to valid memory space types, as the space type can change the type of the cycle driven by the current bus master. The encoding is the same as the valid Motorola function codes for normal accesses. These are driven at address time, and like the low short address, are valid for an entire short or full cycle.

Table 4-1: Memory Space Type Codes

FC_0	FC_1	FC_2	Address Space Type	Z3 Response
0	0	0	Reserved	None
0	0	1	User Data Space	Memory
0	1	0	User Program Space	Memory
0	1	1	Reserved	None
1	0	0	Reserved	None
1	0	1	Supervisor Data Space	Memory
1	1	0	Supervisor Program Space	Memory
1	1	1	CPU Space	Interrupts

Compatibility Cycle Strobe (/CCS)

This is equivalent to the Zorro II address strobe, /AS. A Zorro III PIC doesn't use this for normal operation, but may use it during the autoconfiguration process if configuring at the Zorro II address. AUTOCONFIG® cycles at $00E8xxxx always look like Zorro II cycles, though of course /FCS and the full Zorro III address is available, so a card can use either Zorro II or Zorro III addressing to start the cycle. However, using the /CCS strobe can save the designer the need to compare the upper 8 bits of address. Data must be driven Zorro II style, though if the /DS_N lines are respected for reads, /CINH is asserted, and /MTACK is negated, the resulting Zorro III cycle will fit within the expected Zorro II cycle generated by the bus controller. Yes, that should sound weird; it's based on the mapping of Zorro II vs. Zorro III signals, and of course the fact that /FCS always starts any cycle. Also note that a bus cycle with /CCS asserted and /FCS negated is always a Zorro II PIC-as-master cycle. Many Zorro III cards will instead configure at the alternate $FF00xxxx base address, fully in Zorro III mode, and thus completely ignore this signal.

Full Cycle Strobe (/FCS)

This is the standard Zorro III full cycle strobe. This is asserted by the bus master shortly after addresses are valid on the bus, and signals the start of any kind of Zorro III bus cycle. Shortly after this line is asserted, all the multiplexed addresses will go invalid, so in general, all slaves latch the bus address on the falling edge of /FCS. Also, /BG$_N$ line is negated for a Zorro III mastered cycle shortly after /FCS is asserted by the master.

4.7 Data and Related Control Signals

The data time signals here manage the actual transfer of data between master and slave for both full and short cycle types. The burst mode signals are here too, as they're basically data phase signals even through they don't only concern the transfer of data.

Data Output Enable (DOE)

This signal is used by an expansion card to enable the buffers on the data bus. The bus master drives this line is to keep slave PICs from driving data on the bus until *data time*.

Data Bus (D$_0$-D$_{31}$)

This is the Zorro III data bus, which is driven by either the master or the slave when DOE is asserted by the master (based on READ). It's valid for reads when /DTACK is asserted by the slave; on writes when at least one of /DS$_N$ is asserted by the master, for all cycle types.

Data Strobes (/DS$_N$)

These strobes fall during *data time*; /DS$_3$ strobes D$_{24}$-D$_{31}$, while /DS$_0$ strobes D$_0$-D$_7$. For write cycles, these lines signal data valid on the bus. At all times, they indicate which bytes in the 32 bit data word the bus master is actually interested in. For cachable reads, all four bytes must be returned, regardless of the value of the sizing strobes. For writes, only those bytes corresponding to asserted /DS$_N$ are written. Only contiguous byte cycles are supported; e.g. /DS$_{3\text{-}0}$ = 2, 4, 5, 6, or 10 is invalid.

Data Transfer Acknowledge (/DTACK)

This signal is used to normally terminate a Zorro III cycle. The slave is always responsible for driving this signal. For a read cycle, it asserts /DTACK as soon as it has driven valid data onto the data bus. For a write cycle, it asserts /DTACK as soon as it's done with the data. Latching the data on writes may be a good idea; that can allow a slave to end the cycle before it has actually finished writing the data to its local memory.

Cache Inhibit (/CINH)

This line is asserted at the same time as /SLAVE$_N$ to indicate to the bus master that the cycle must not be cached. If a device doesn't support caching, it must assert /CINH and actually obey the /DS$_N$ byte strobes for read cycles. Conversely, if the device supports caching, /CINH is negated and the device returns all four bytes valid on reads, regardless of the actual supplied /DS$_N$ strobes.

Multiple Cycle Transfers (/MTCR,/MTACK)

These lines comprise the Multiple Transfer Cycle handshake signals. The bus master asserts /MTCR at the start of *data time* if it’s capable of supporting Multiple Transfer Cycles, and the slave asserts /MTACK with /SLAVEN if it’s capable of supporting Multiple Transfer Cycles. If the handshake goes through, /MTCR strobes in the short address and write data as long as the full cycle continues.

CHAPTER 5

TIMING

"When dealing with the insane, the best method is
to pretend to be sane."
-Hermann Hesse

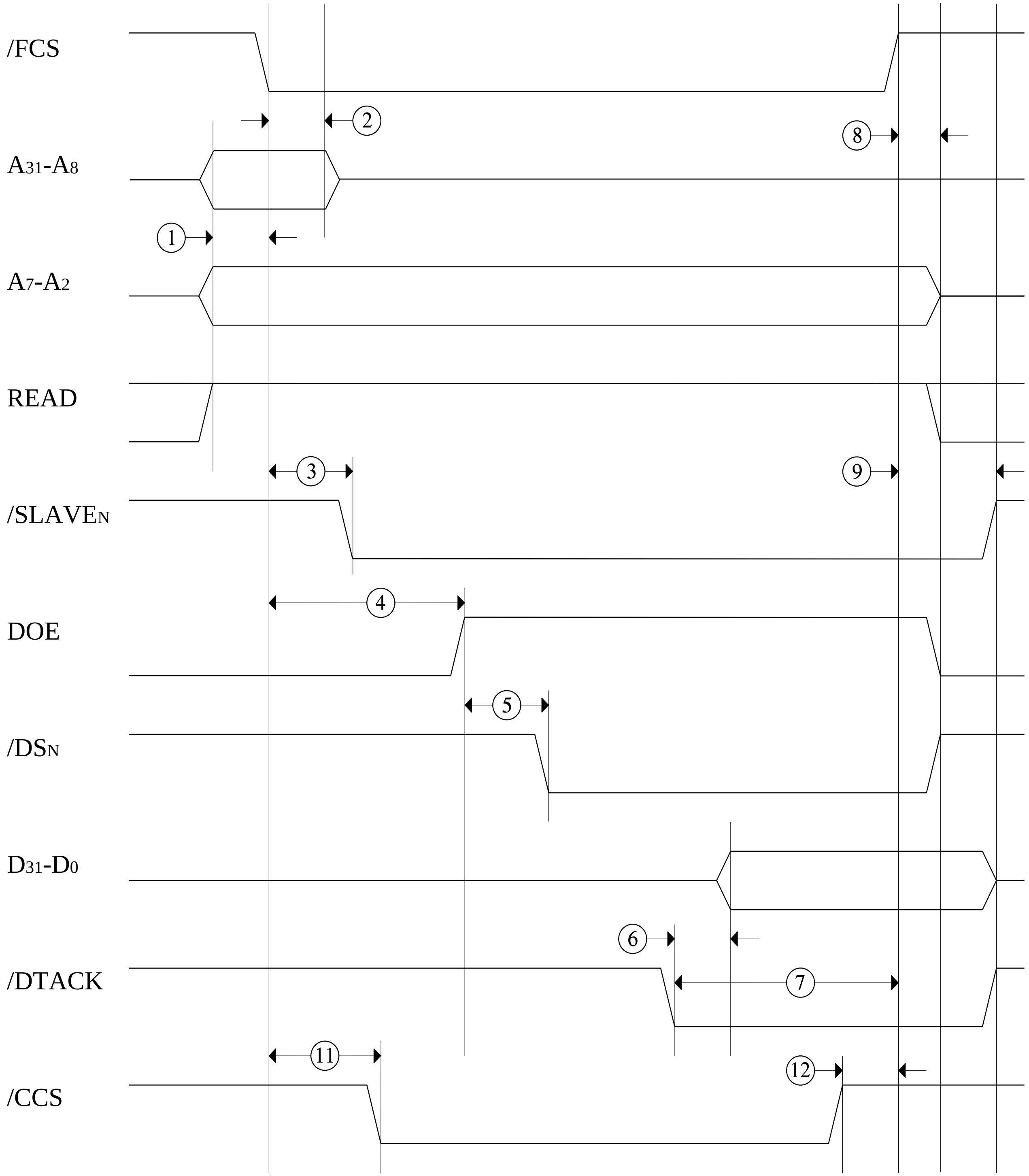
Some of this information is considered preliminary. Nothing is expected to get any more speed critical, but as mentioned previously, the testing of Zorro III designs has just started at the time of this writing, final bus controllers are not yet available, and only a few PIC designs have even been conceived.

This section covers the various timing specifications in detail for different Zorro III operations. It's important to realize that this timing information is a **specification**. Actual Zorro III systems may offer much more relaxed timings. Today. The whole point of the specification is that as long as all Zorro III PICs and all Zorro III backplanes base things on the timings given here, they'll always work together nicely. Any design based on the actual characteristics of any particular backplane will very likely wind up working only on that particular backplane.

The philosophy of timing on the Zorro III bus is to keep things as simple as possible without compromising the performance goals of the bus. Zorro III PICs are expected to be based on F-Series or ACT-series TTL logic, fast PALs, and possibly full custom chip designs. It's very unlikely the designer will meet any of these specifications with the LS parts left over from old Zorro II card designs.

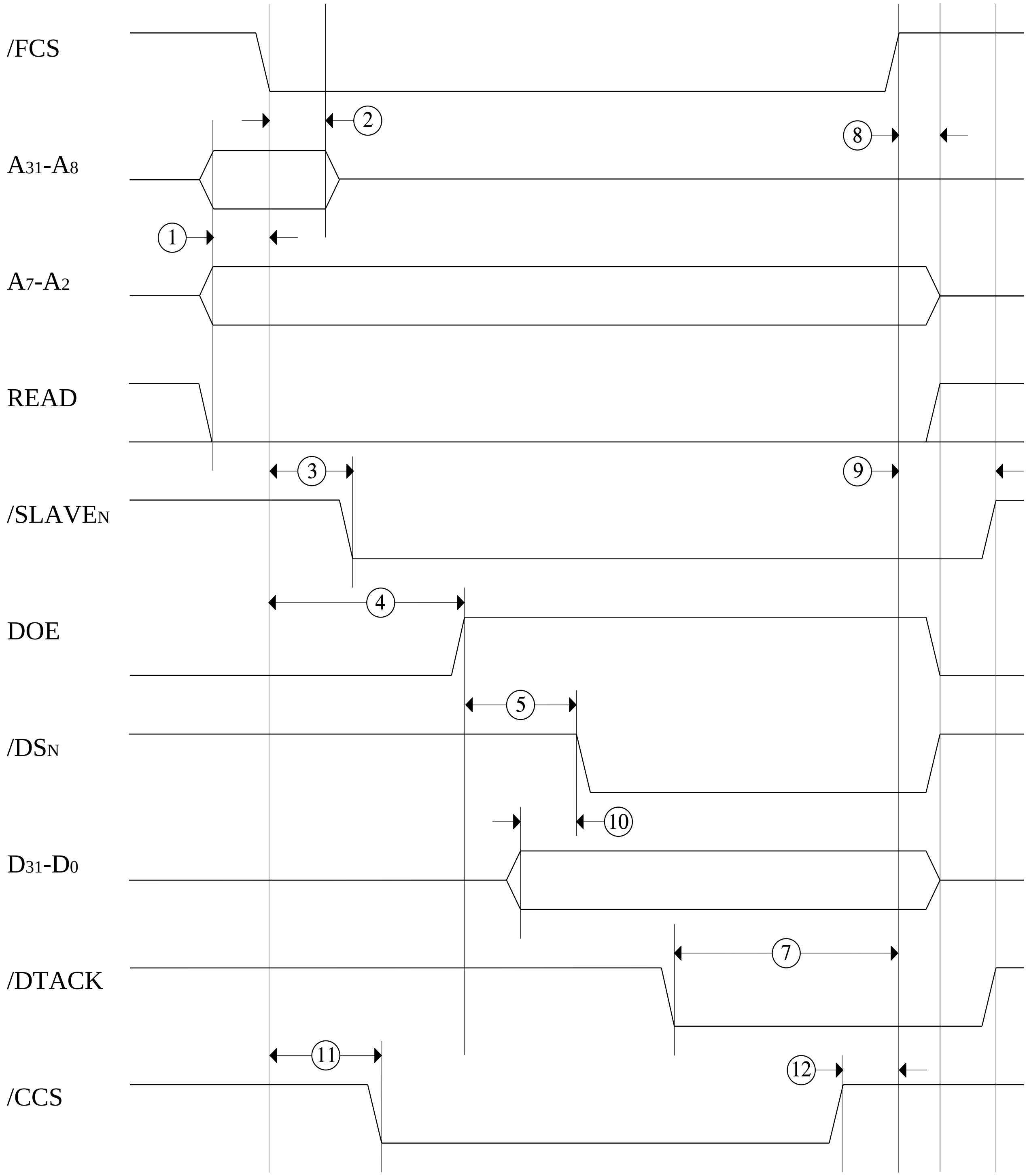
5.1 Standard Read Cycle Timing

No.	Name	Symbol	Min	Max
1	Address setup to /FCS	T_{AFS}	15ns	-----
2	Address hold from /FCS	T_{HAF}	10ns	-----
3	/FCS to /SLAVE$_N$ delay	T_{SLV}	-----	25ns
4	/FCS to DOE delay	T_{DOE}	30ns	-----
5	DOE to /DS$_N$ delay	T_{DS}	10ns	-----
6	Data setup to /DTACK	T_{RDS}	0ns	-----
7	/DTACK to /FCS off	T_{OFF}	10ns	-----
8	Master signal hold from /FCS off	T_{HMC}	0ns	5ns
9	Slave signal hold from /FCS off	T_{HSC}	0ns	15ns
11	/FCS to /CCS delay	T_{CCS}	35ns	175ns
12	/CCS off to /FCS off	T_{OVL}	40ns	-----


/FCS
A31-A8
A7-A2
READ
/SLAVEN
DOE
/DSN
D31-D0
/DTACK
/CCS
1
2
3
4
5
6
7
8
9
11
12

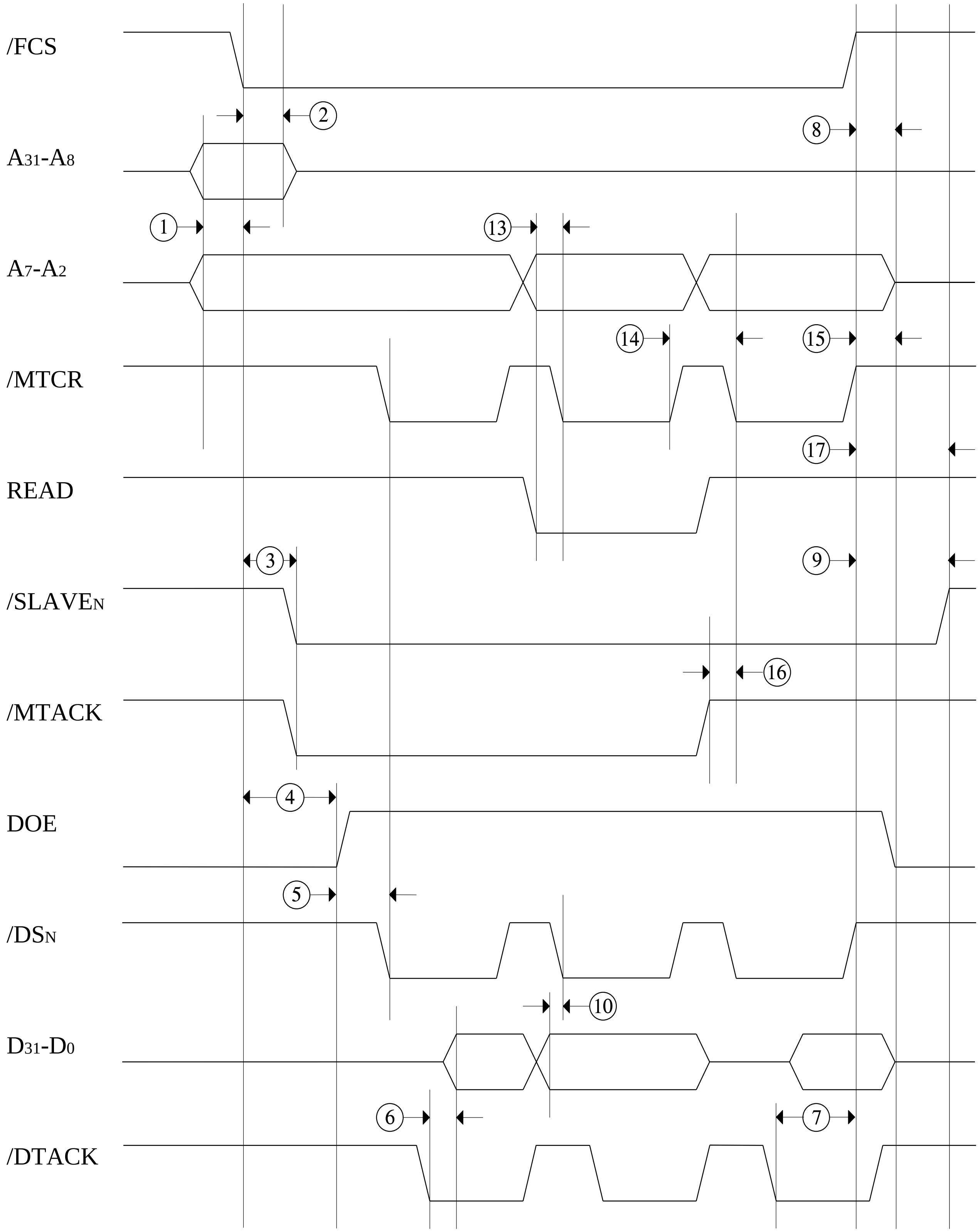
5.2 Standard Write Cycle Timing

No.	Name	Symbol	Min	Max
1	Address setup to /FCS	TAFS	15ns	-----
2	Address hold from /FCS	THAF	10ns	-----
3	/FCS to /SLAVEN delay	TSLV	-----	25ns
4	/FCS to DOE delay	TDOE	30ns	-----
5	DOE to /DSN delay	TDS	10ns	-----
7	/DTACK to /FCS off	TOFF	10ns	-----
8	Master signal hold from /FCS off	THMC	0ns	5ns
9	Slave signal hold from /FCS off	THSC	0ns	15ns
10	Write data setup to /DSN	TWDS	5ns	-----
11	/FCS to /CCS delay	TCCS	35ns	175ns
12	/CCS off to /FCS off	TOVL	40ns	-----


/FCS
A31-A8
A7-A2
READ
/SLAVEN
DOE
/DSN
D31-D0
/DTACK
/CCS
1
2
3
4
5
7
8
9
10
11
12

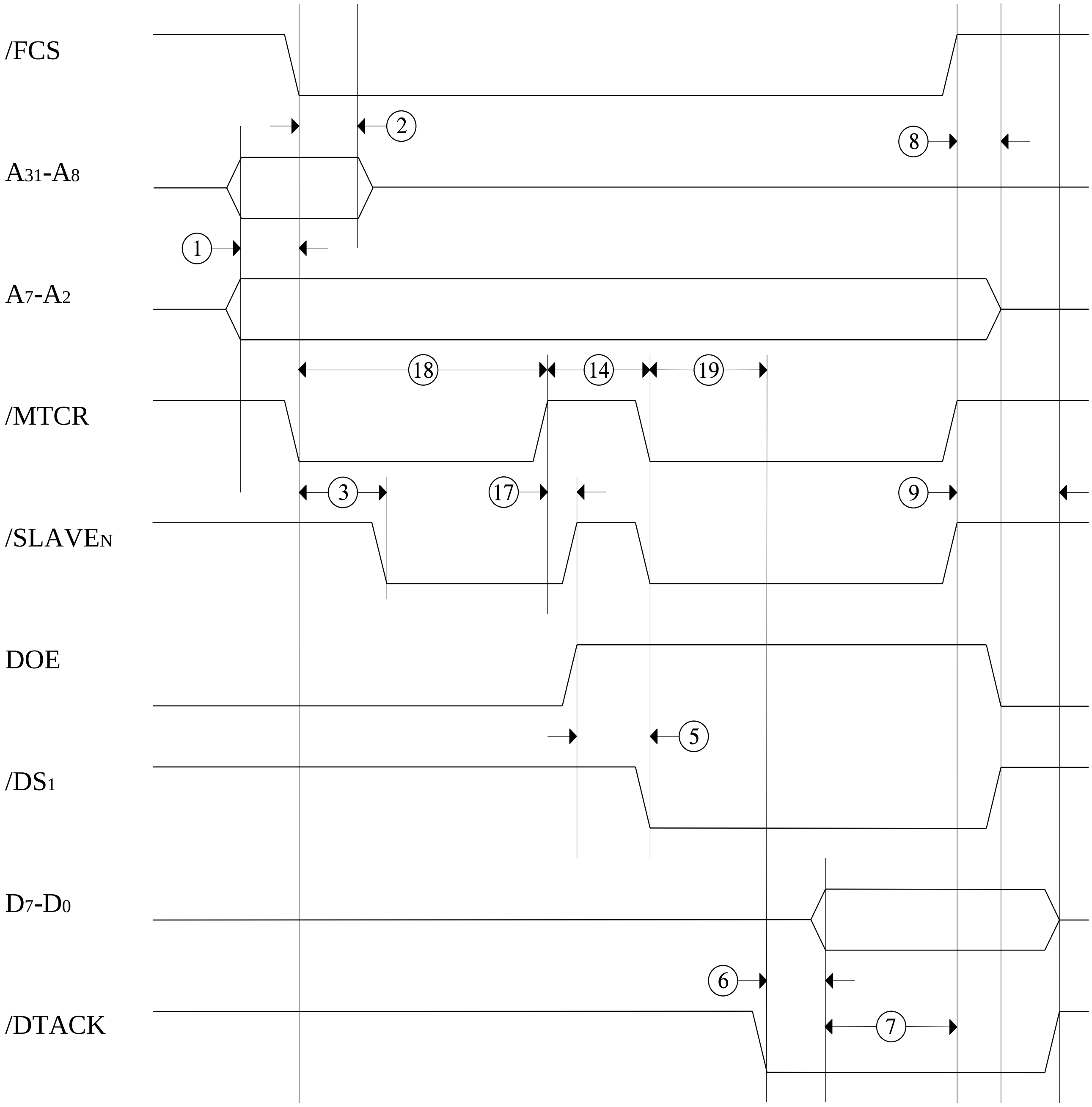
5.3 Multiple Transfer Cycle Timing

No.	Name	Symbol	Min	Max
1	Address setup to /FCS	TAFS	15ns	-----
2	Address hold from /FCS	THAF	10ns	-----
3	/FCS to /SLAVEN, /MTACK delay	TSLV	-----	25ns
4	/FCS to DOE delay	TDOE	30ns	-----
5	DOE to /DSN, /MTCR delay	TDS	10ns	-----
6	Data setup to /DTACK	TRDS	0ns	-----
7	/DTACK to /FCS, /MTCR off	TOFF	10ns	-----
8	Master signal hold from /FCS off	THMC	0ns	5ns
9	Slave signal hold from /FCS off	THSC	0ns	15ns
10	Write data setup to /DSN	TWDS	5ns	-----
13	Address, READ setup to /MTCR	TAMS	5ns	-----
14	/MTCR off to /MTCR on	TREF	10ns	-----
15	Address, READ hold from /MTCR	THAM	0ns	-----
16	/MTACK off to /MTCR	TBCD	10ns	-----
17	Slave signal hold from /MTCR off	THSM	0ns	5ns


/FCS
A31-A8
A7-A2
/MTCR
READ
/SLAVEN
/MTACK
DOE
/DSN
D31-D0
/DTACK
1
2
3
4
5
6
7
8
9
10
13
14
15
16
17

5.4 Quick Interrupt Cycle Timing

No.	Name	Symbol	Min	Max
1	Address setup to /FCS	T_{AFS}	15ns	-----
2	Address hold from /FCS	T_{HAF}	10ns	-----
3	/FCS to /SLAVE$_N$ delay	T_{SLV}	-----	25ns
5	DOE to /DS$_N$ delay	T_{DS}	10ns	-----
6	Data setup to /DTACK	T_{RDS}	0ns	-----
7	/DTACK to /FCS off	T_{OFF}	10ns	-----
8	Master signal hold from /FCS off	T_{HMC}	0ns	5ns
9	Slave signal hold from /FCS off	T_{HSC}	0ns	15ns
14	/MTCR off to /MTCR on	T_{REF}	10ns	-----
17	Slave signal hold from /MTCR off	T_{HSM}	0ns	5ns
18	Poll Phase time	T_{POL}	30ns	100ns
19	Vector Phase start to /DTACK time	T_{VEC}	-----	100ns


/FCS
A31-A8
A7-A2
/MTCR
/SLAVEN
DOE
/DS1
D7-D0
/DTACK
1
2
3
5
6
7
8
9
14
17
18
19

CHAPTER 6

ELECTRICAL SPECIFICATIONS

"...I collected the instruments of life around me, that I might infuse a spark of being into the lifeless thing that lay at my feet"

-Victor Frankenstein

The Zorro III bus has a number of electrical specifications that are very important for PIC designers to consider, along with the timing parameters of course. It's extremely important to base designs on the specification of the backplane, rather than the actual behavior of the backplane. New backplanes for new machines are designed to conform to the specification, they are not necessarily based on previous designs. This is expecially important with the Zorro III bus, since timing is far more critical than in the past, and the bus controller is designed from this specification, rather than the reverse, as in the Amiga 2000.

6.1 Expansion Bus Loading

The Zorro III bus loading is specified based on typical TTL family "F" series buffer devices, though in reality, compatible CMOS devices are likely to be used in some bus controllers or PICs. Thus, it's important to accept the TTL levels as a minimum voltage level, and make sure that all inputs are the appropriate TTL levels, while outputs can be at TTL or CMOS voltage levels as long as they provide the required source and sink.

While some A2000 designs used "LS" or "ALS" buffers instead of "F", the bus will generally work with these older cards, at least with current backplane designs such as the A3000 backplane. However, Zorro III designs must exactly obey these loading rules; it’s very probable that some future Zorro III machines will have a large number of slots. In such machines, PICs built on the Zorro II specification will still work in a lightly loaded bus, but may not function in a fully loaded bus. All Zorro III PICs built to spec will work in any Zorro III backplane, without any loading problems, if all loading and timing rules are followed by the PIC designer. The bus signals are divided up into the four groups shown in Table 6-1, based on the loading characteristics of the particular signal. The signals in each group are given here.

Table 6-1: Zorro III Drive Types

Signal	Direction	High Level	Low Level
Standard	Loading	+140µA @ +2.7VDC	-3.2mA @ +0.4VDC
	Driven	+2.5VDC @ -3.0mA	+0.4VDC @ +64mA
Clock	Loading	+20µA @ +2.7VDC	-1.6mA @ +0.4VDC
O.C.	Loading	+80µA @ +2.7VDC	-3.2mA @ +0.4VDC
	Driven	Not Driven	+0.4VDC @ +20mA
Non-bussed	Loading	+80µA @ +2.7VDC	-1.0mA @ +0.4VDC
	Driven	+2.5VDC @ -0.4mA	+0.4VDC @ +4.0mA

6.1.1 Standard Signals

The majority of signals on the bus are in this group. These are bussed signals, driven actively on the bus by F-series (or compatible) drivers such as 74F245, usually tri-stated when ownership of the signal changed for master and slave, and generally terminated with a 220Ω/330Ω thevenin terminator. PICs can apply two standard loads to each of these signals when necessary.

/FCS /CCS /DS$_0$-/DS$_3$ /LOCK
A$_2$-A$_7$ AD$_8$-AD$_{31}$ SD$_0$-SD$_7$ READ
FC$_0$-FC$_2$ DOE /IORST /BCLR
/MTCR /MTACK

6.1.2 Clock Signals

All clock signals on the bus are in this group. Many designs are very sensitive to clock delay, skew, and rise/fall times, so loading on the clock lines must be kept to a minimum. These are bussed signals, actively driven by the backplane, and source terminated with a low value series resistor. PICs can apply one standard load to each of these signals when necessary. Zorro II cards have the same clock rules, so there should never be clocking problems when using either card type in a backplane.

/C3 CDAC /C1 7M
E Clock

6.1.3 Open Collector Signals

Many of the bus signals are shared via open collector or open drain outputs rather than via tri-stated signals; this is of course required for some asynchronous things like the shared interrupt lines, and it works well for other types of signals as well. Of course, a backplane resistor pulls these lines high, PICs only drive the line low.

/OWN /BGACK /CINH /BERR
/DTACK /RESET /INT$_2$ /INT$_6$
/HLT

6.1.4 Non-bussed Signals

The non-bussed, or slot specific, signals are involved with only one slot on the bus (eg, each slot has its own copy). As a result, the drive requirements are much less for these signals. The backplane provides pullups or pulldowns, as required by the specific signal.

/CFGIN$_N$ /CFGOUT$_N$ /BR$_N$ /BG$_N$
SenseZ$_3$ /SLAVE$_N$

6.2 Slot Power Availability

The system power for the Zorro III bus is totally based on the slot configurations. A backplane is always free to supply extra power, but it must meet the minimum requirements specified here. All PICs must be designed with the minimum specifications in mind, especially the tolerances.

Pin	Supply
5,6	+5 VDC ± 5% @ 2 Amps
8	-5 VDC ± 5% @ 60 mA
10	+12 VDC ± 5% @ 500mA
20	-12 VDC ± 5% @ 60mA

6.3 Temperature Range

The Zorro III bus is specified for operation over a temperature range of 0° C to 70° C.

CHAPTER 7

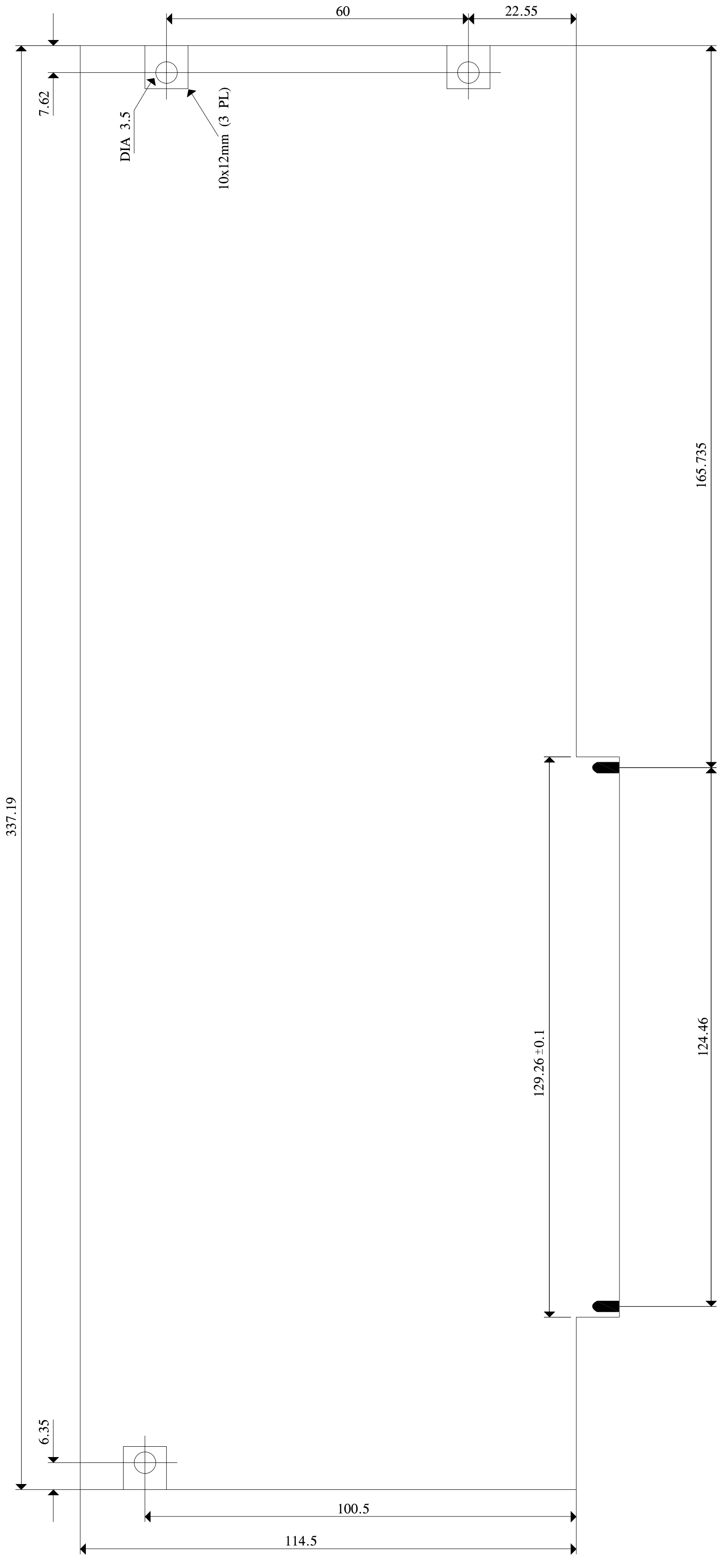
MECHANICAL SPECIFICATIONS

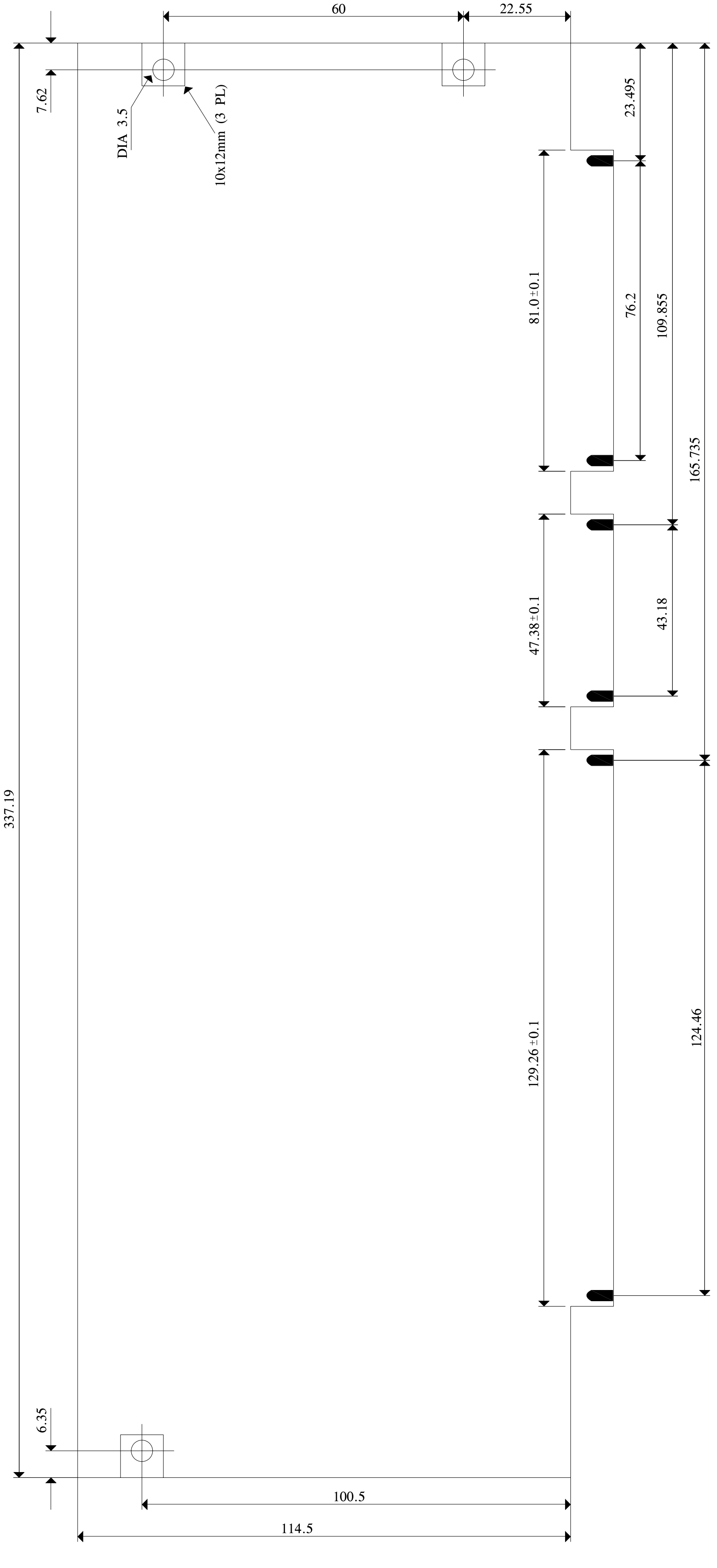
"Never speak more clearly than you think."
-Jeremy Bernstein

This section covers the various mechanical details of Zorro III cards. Note that these specifications are considered preliminary.

7.1 Basic Zorro III PIC

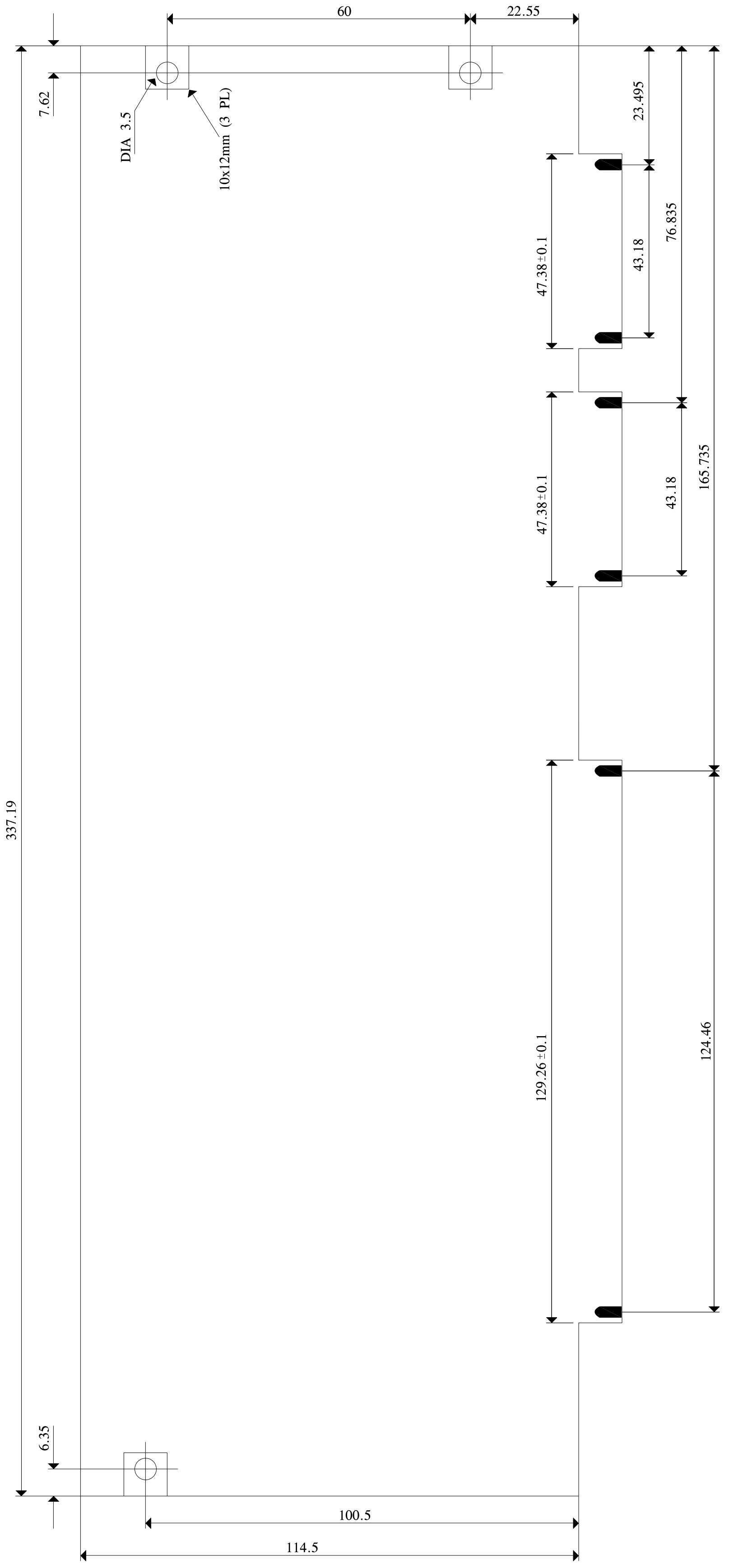
This drawing shows the basic Zorro III PIC. All of the dimensions are in millimeters.





7.2 PIC with ISA Option

This drawing shows the basic Zorro III PIC, with both Zorro III and the ISA Bus fingers specified. All of the dimensions are in millimeters.



7.3 PIC with Video Option

This drawing shows the basic Zorro III PIC, with both Zorro III and the Amiga Video Slot fingers specified. All of the dimensions are in millimeters. Please consult the *A500/A2000 Technical Reference Manual* for the form factor specification of a video-only card that will fit both Amiga 2000 and Amiga 3000 computers.

CHAPTER 8

AUTOCONFIG®

"The goal of all inanimate objects is to resist man and ultimately defeat him."
-Russell Baker

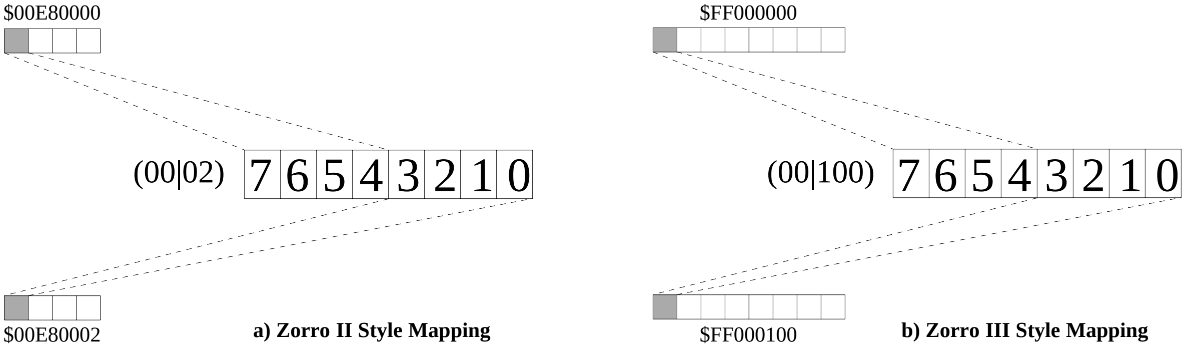
8.1 The AUTOCONFIG® Mechanism

The AUTOCONFIG® mechanism used for the Zorro III bus is an extension of the original Zorro II configuration mechanism. The main reason for this is that the Zorro II mechanism works so well, there was little need to change anything. The changes are simply support for new hardware features on the Zorro III bus.

Amiga autoconfiguration is surprisingly simple. When an Amiga powers up or resets, every card in the system goes to its unconfigured state. At this point, the most important signals in the system are /CFGIN$_N$ and /CFGOUT$_N$. As long as a card's /CFGIN$_N$ line is negated, that card sits quietly and does nothing on the bus (though memory cards should continue to refresh even through reset, and any local board activities that don't concern the bus may take place after /RESET is negated). As part of the unconfigured state, /CFGOUT$_N$ is negated by the PIC immediately on reset.

The configuration process begins when a card's /CFGIN$_N$ line is asserted, either by the backplane, if it's the first slot, or via the configuration chain, if it's a later card. The configuration chain simply ensures that only one unconfigured card will see an asserted /CFGIN$_N$ at one time. An unconfigured card that sees its /CFGIN$_N$ line asserted will respond to a block of memory called *configuration space*. In this block, the PIC will assert a set of read-only registers, followed by a set of write-only registers (the read-only registers are also known as AUTOCONFIG® ROM). Starting at the base of this block, the read registers describe the device's size, type, and other requirements. The operating system reads these, and based on them, decides what should be written to the board. Some write information is optional, but a board will always be assigned a base address or be told to shut up. The act of writing the final bit of base address, or writing anything to a shutup address, will cause the PIC to assert its /CFGOUT$_N$, enabling the next board in the configuration chain.

The Zorro II configuration space is the 64K memory block $00E8xxxx, which of course is driven with 16 bit Zorro II cycles; all Zorro II cards configure there. The Zorro III configuration space is the 64K memory block beginning at $FF00xxxx, which is always driven with 32 bit Zorro III cycles (PICs need only decode A_{31}-A_{24} during configuration). A Zorro III PIC can configure in Zorro II or Zorro III configuration space, at the designer's discretion, but not both at once. All read registers physically return only the top 4 bits of data, on D_{31}-D_{28} for either bus mode. Write registers are written to support nybble, byte, and word registers for the same register, again based on what works best in hardware. This design attempts to map into real hardware as simply as possible. Every AUTOCONFIG® register is logically considered to be 8 bits wide; the 8 bits actually being nybbles from two paired addresses.



a) Zorro II Style Mapping

b) Zorro III Style Mapping

Figure 8-1: Configuration Register Mapping

The register mappings for the two different blocks are shown in *Figure 8-1*. All the bit patterns mentioned in the following sections are logical values. To avoid ambiguity, all registers are referred to by the number of the first register in the pair, since the first pair member is the same for both mapping schemes. In the actual implementation of these registers, all read registers except for the 00 register are physically complemented; eg, the logical value of register 3C is always 0, which means in hardware, the upper nybbles of locations $00E8003C and $00E8003E, or $FF00003C and $FF00013C, both return all 1's.

8.2 Register Bit Assignments

The actual register assignments are below. Most of the registers are the same as for the Zorro II bus, but are included here anyway for completeness. The Amiga OS software names for these registers in the ExpansionRom or ExpansionControl structures are included.

Reg Z2 Z3 Bit

00 02 100 7,6 These bits encode the PIC type:
(er_Type)

00	Reserved
01	Reserved
10	Zorro III
11	Zorro II

5 If this bit is set, the PIC's memory will be linked into the system free pool. The Zorro III register 08 may modify the size of the linked memory.

4 Setting this bit tells the OS to read an autoboot ROM.

3 This bit is set to indicate that the next board is related to this one; often logically separate PICs are physically located on the same card.

2-0 These bits indicate the configuration size of the PIC. This size can be modified for the Zorro III cards by the size extension bit, which is the new meaning of bit 5 in register 08.

Bits	Unextended	Extended
000	8 megabytes	16 megabytes
001	64 kilobytes	32 megabytes
010	128 kilobytes	64 megabytes
011	256 kilobytes	128 megabytes
100	512 kilobytes	256 megabytes
101	1 megabyte	512 megabytes
110	2 megabytes	1 gigabyte
111	4 megabytes	RESERVED

04 06 104 7-0 The device's product number, which is completely up to the manufacturer. This is generally unique between different products, to help in identification of system cards, and it must be unique between devices using the automatic driver binding features.
(er_Product)

08 0A 108 7 This was originally an indicator to place the card in the 8 megabyte Zorro II space, when set, or anywhere it'll fit, if cleared. Under the Zorro III spec, this is set to indicate that the board is basically a memory device, cleared to indicate that the board is basically an I/O device.
(er_Flags)

6 This bit is set to indicate that the board can't be shut up by software, cleared to indicate that the board can be shut up.

5 This is the size extension bit. If cleared, the size bits in register 00 mean the same as under Zorro II, if set, the size bits indicate a new size. The

most common new Zorro III sizes are the smaller ones; all new sized cards get aligned on their natural boundaries.

4 Reserved, must be 1 for all Zorro III cards.

3-0 These bits indicate a board's sub-size; the amount of memory actually required by a PIC. For memory boards that auto-link, this is the actual amount of memory that will be linked into the system free memory pool. A memory card, with memory starting at the base address, can be automatically sized by the Operating System. This sub-size option is intended to support cards with variable setups without requiring variable physical configuration capability on such cards. It also may greatly simplify a Zorro III design, since 16 megabyte cards and up can be designed with a single latch and comparator for base address matching, while 8 megabyte and smaller PICs require large latch/comparator circuits not available in standard TTL packages.

Bits	Encoding
0000	Logical size matches physical size
0001	Automatically sized by the Operating System
0010	64 kilobytes
0011	128 kilobytes
0100	256 kilobytes
0101	512 kilobytes
0110	1 megabyte
0111	2 megabytes
1000	4 megabytes
1001	6 megabytes
1010	8 megabytes
1011	10 megabytes
1100	12 megabytes
1101	14 megabytes
1110	Reserved
1111	Reserved

For boards that wish to be automatically sized by the operating system, a few rules apply. The memory is sized in 512K increments, and grows from the base address upward. Memory wraps are detected, but the design must insure that its data bus doesn't float when the sizing routine addresses memory locations that aren't physically present on the board; data bus pullups or pulldowns are recommended. This feature is designed to allow boards to be easily upgraded with additional or increased density memoried without the need for memory configuration jumpers.

Reg	Z2	Z3	Bit	
0C	0E **(er_Reserved03)**	10C	7-0	Reserved, must be 0.
10	12	110	7-0	Manufacturer's number, high byte.
14	16 **(er_Manufacturer)**	114	7-0	Manufacturer's number, low bytes. These are unique, and can only be assigned by Commodore.
18	1A	118	7-0	Optional serial number, byte 0 (msb)
1C	1E	11C	7-0	Optional serial number, byte 1
20	22	120	7-0	Optional serial number, byte 2
24	26 **(er_SerialNumber)**	124	7-0	Optional serial number, byte 3 (lsb) This is for the manufacturer's use and can contain anything at all. The main intent is to allow a manufacturer to uniquely identify individual cards, but it can certainly be used for revision information or other data.
28	2A	128	7-0	Optional ROM vector, high byte.
2C	2E **(er_InitDiagVec)**	12C	7-0	Optional ROM vector, low byte. If the ROM address valid bit (bit 4 of register (00|02)) is set, these two registers provide the sixteen bit offset from the board's base at which the start of the ROM code is located. If the ROM address valid bit is cleared, these registers are ignored.
30	32 **(er_Reserved0c)**	130	7-0	Reserved, must be 0. Unsupported base register reset register under Zorro II*.
34	36 **(er_Reserved0d)**	134	7-0	Reserved, must be 0.
38	3A **(er_Reserved0e)**	138	7-0	Reserved, must be 0.
3C	3E **(er_Reserved0f)**	13C	7-0	Reserved, must be 0.
40	42 **(ec_Interrupt)**	140	7-0	Reserved, must be 0. Unsupported control state register under Zorro II*.
44	46	144	7-0	High order base address register, write only.
48	4A **(ec_Z3_HighByte)** **(ec_BaseAddress)**	148	7-0	Low order base address register, write only. The high order register takes bits 31-24 of the board's configured address, the low ordered resgister takes bits 23-16. For Zorro III boards configured in the Zorro II space, the configuration address is written both nybble and byte wide, with the ordering:

* The original Zorro specifications called for a few registers, like these, that remained active after configuration. Support for this is impossible, since the configuration registers generally disappear when a board is configured, and absolutely must move out of the $00E8xxxx space. So since these couldn't really be implemented in hardware, system software has never supported them. They're included here for historical purposes.

Reg Z2 Z3 Bit

Reg	Nybble	Byte
46	A_{27}-A_{24}	N/A
44	A_{31}-A_{28}	A_{31}-A_{24}
4A	A_{19}-A_{16}	N/A
48	**A_{23}-A_{20}**	**A_{23}-A_{16}**

Note that writing to register 48 actually configures the board for both Zorro II and Zorro III boards in the Zorro II configuration block. For Zorro III PICs in the Zorro III configuration block, the action is slightly different. The software will actually write the configuration as byte and word wide accesses:

Reg	Byte	Word
48	A_{23}-A_{16}	N/A
44	**A_{31}-A_{24}**	**A_{31}-A_{16}**

The actual configuration takes place when register 44 is written, thus supporting any physical size of configuration register.

4C	4E **(ec_Shutup)**	14C	7-0	Shut up register, write only. Anything written to 4C will cause a board that supports shut-up to completely disappear until the next reset.
50	52	150	7-0	Reserved, must be 0.
54	56	154	7-0	Reserved, must be 0.
58	5A	158	7-0	Reserved, must be 0.
5C	5E	15C	7-0	Reserved, must be 0.
60	62	160	7-0	Reserved, must be 0.
64	66	164	7-0	Reserved, must be 0.
68	6A	168	7-0	Reserved, must be 0.
6C	6E	16C	7-0	Reserved, must be 0.
70	72	170	7-0	Reserved, must be 0.
74	76	174	7-0	Reserved, must be 0.
78	7A	178	7-0	Reserved, must be 0.
7C	7E	17C	7-0	Reserved, must be 0.

APPENDICES

"I have been given the freedom to do as I see fit."
-REM

A.1 Physical and Logical Signal Names

The Amiga 3000 Bus signals vary based on the particular bus mode in effect. This table lists each physical pin by physical name, and then by the logical names for Zorro II mode, Zorro III mode, address phase, and Zorro III data mode, data phase.

PIN NO.	Physical Name	Zorro II Name	Zorro III Address Phase	Zorro III Data Phase
1	Ground	Ground	Ground	Ground
2	Ground	Ground	Ground	Ground
3	Ground	Ground	Ground	Ground
4	Ground	Ground	Ground	Ground
5	+5VDC	+5VDC	+5VDC	+5VDC
6	+5VDC	+5VDC	+5VDC	+5VDC
7	/OWN	/OWN	/OWN	/OWN
8	-5VDC	-5VDC	-5VDC	-5VDC
9	/SLAVE$_N$	/SLAVE$_N$	/SLAVE$_N$	/SLAVE$_N$
10	+12VDC	+12VDC	+12VDC	+12VDC
11	/CFGOUT$_N$	/CFGOUT$_N$	/CFGOUT$_N$	/CFGOUT$_N$
12	/CFGIN$_N$	/CFGIN$_N$	/CFGIN$_N$	/CFGIN$_N$
13	Ground	Ground	Ground	Ground
14	/C3	/C3 Clock	/C3 Clock	/C3 Clock
15	CDAC	CDAC Clock	CDAC Clock	CDAC Clock
16	/C1	/C1 Clock	/C1 Clock	/C1 Clock
17	/CINH	/OVR	/CINH	/CINH
18	/MTCR	XRDY	/MTCR	/MTCR
19	/INT$_2$	/INT$_2$	/INT$_2$	/INT$_2$
20	-12VDC	-12VDC	-12VDC	-12VDC
21	A$_5$	A$_5$	A$_5$	A$_5$
22	/INT$_6$	/INT$_6$	/INT$_6$	/INT$_6$
23	A$_6$	A$_6$	A$_6$	A$_6$
24	A$_4$	A$_4$	A$_4$	A$_4$
25	Ground	Ground	Ground	Ground
26	A$_3$	A$_3$	A$_3$	A$_3$
27	A$_2$	A$_2$	A$_2$	A$_2$
28	A$_7$	A$_7$	A$_7$	A$_7$
29	/LOCK	A$_1$	/LOCK	/LOCK
30	AD$_8$	A$_8$	A$_8$	D$_0$
31	FC$_0$	FC$_0$	FC$_0$	FC$_0$
32	AD$_9$	A$_9$	A$_9$	D$_1$
33	FC$_1$	FC$_1$	FC$_1$	FC$_1$
34	AD$_{10}$	A$_{10}$	A$_{10}$	D$_2$
35	FC$_2$	FC$_2$	FC$_2$	FC$_2$
36	AD$_{11}$	A$_{11}$	A$_{11}$	D$_3$
37	Ground	Ground	Ground	Ground
38	AD$_{12}$	A$_{12}$	A$_{12}$	D$_4$
39	AD$_{13}$	A$_{13}$	A$_{13}$	D$_5$
40	Reserved	(/EINT$_7$)	Reserved	Reserved
41	AD$_{14}$	A$_{14}$	A$_{14}$	D$_6$
42	Reserved	(/EINT$_5$)	Reserved	Reserved

PIN	Physical	Zorro II	Zorro III	Zorro III
NO.	Name	Name	Address Phase	Data Phase
43	AD$_{15}$	A$_{15}$	A$_{15}$	D$_{7}$
44	Reserved	(/EINT$_{4}$)	Reserved	Reserved
45	AD$_{16}$	A$_{16}$	A$_{16}$	D$_{8}$
46	/BERR	/BERR	/BERR	/BERR
47	AD$_{17}$	A$_{17}$	A$_{17}$	D$_{9}$
48	/MTACK	(/VPA)	/MTACK	/MTACK
49	Ground	Ground	Ground	Ground
50	E Clock	E Clock	E Clock	E Clock
51	/DS$_{0}$	(/VMA)	/DS$_{0}$	/DS$_{0}$
52	AD$_{18}$	A$_{18}$	A$_{18}$	D$_{10}$
53	/RESET	/RST	/RESET	/RESET
54	AD$_{19}$	A$_{19}$	A$_{19}$	D$_{11}$
55	/HLT	/HLT	/HLT	/HLT
56	AD$_{20}$	A$_{20}$	A$_{20}$	D$_{12}$
57	AD$_{22}$	A$_{22}$	A$_{22}$	D$_{14}$
58	AD$_{21}$	A$_{21}$	A$_{21}$	D$_{13}$
59	AD$_{23}$	A$_{23}$	A$_{23}$	D$_{15}$
60	/BR$_{N}$	/BR$_{N}$	/BR$_{N}$	/BR$_{N}$
61	Ground	Ground	Ground	Ground
62	/BGACK	/BGACK	/BGACK	/BGACK
63	AD$_{31}$	D$_{15}$	A$_{31}$	D$_{31}$
64	/BG$_{N}$	/BG$_{N}$	/BG$_{N}$	/BG$_{N}$
65	AD$_{30}$	D$_{14}$	A$_{30}$	D$_{30}$
66	/DTACK	/DTACK	/DTACK	/DTACK
67	AD$_{29}$	D$_{13}$	A$_{29}$	D$_{29}$
68	READ	READ	READ	READ
69	AD$_{28}$	D$_{12}$	A$_{28}$	D$_{28}$
70	/DS$_{2}$	/LDS	/DS$_{2}$	/DS$_{2}$
71	AD$_{27}$	D$_{11}$	A$_{27}$	D$_{27}$
72	/DS$_{3}$	/UDS	/DS$_{3}$	/DS$_{3}$
73	Ground	Ground	Ground	Ground
74	/CCS	/AS	/CCS	/CCS
75	SD$_{0}$	D$_{0}$	Reserved	D$_{16}$
76	AD$_{26}$	D$_{10}$	A$_{26}$	D$_{26}$
77	SD$_{1}$	D$_{1}$	Reserved	D$_{17}$
78	AD$_{25}$	D$_{9}$	A$_{25}$	D$_{25}$
79	SD$_{2}$	D$_{2}$	Reserved	D$_{18}$
80	AD$_{24}$	D$_{8}$	A$_{24}$	D$_{24}$
81	SD$_{3}$	D$_{3}$	Reserved	D$_{19}$
82	SD$_{7}$	D$_{7}$	Reserved	D$_{23}$
83	SD$_{4}$	D$_{4}$	Reserved	D$_{20}$
84	SD$_{6}$	D$_{6}$	Reserved	D$_{22}$

PIN	Physical	Zorro II	Zorro III	Zorro III
NO.	Name	Name	Address Phase	Data Phase
85	Ground	Ground	Ground	Ground
86	SD$_5$	D$_5$	Reserved	D$_{21}$
87	Ground	Ground	Ground	Ground
88	Ground	Ground	Ground	Ground
89	Ground	Ground	Ground	Ground
90	Ground	Ground	Ground	Ground
91	SenseZ$_3$	Ground	SenseZ$_3$	SenseZ$_3$
92	7M	E7M	7M	7M
93	DOE	DOE	DOE	DOE
94	/IORST	/BUSRST	/IORST	/IORST
95	/BCLR	/GBG	/BCLR	/BCLR
96	Reserved	(/EINT$_1$)	Reserved	Reserved
97	/FCS	No Connect	/FCS	/FCS
98	/DS$_1$	No Connect	/DS$_1$	/DS$_1$
99	Ground	Ground	Ground	Ground
100	Ground	Ground	Ground	Ground

A.2 A Glossary of Terms

The reader may be unfamiliar with a number of terms used in this document. Every effort has been made to include all such terms here.

address A byte-numbered memory location. The Zorro II bus is based on a 24 bit address, the Zorro III bus on a 32 bit address.

arbitration The unambiguous selection of one request out of a number of possible simultaneous requests for a resource. There are two kinds of arbitration in a Zorro III system; bus arbitration and quick interrupt arbitration.

asserted The active state of a state, regardless of its logic sense.

atomic cycle A cycle or set of cycles that are uninterruptable, and thus treated as a unit; both Multiple Transfer and LOCKed cycles are considered atomic under the Zorro III bus.

AUTOCONFIG® From "automatic configuration", the Zorro bus specification for how software and hardware cooperate to permit PIC addresses to be set by software and PIC type information to be determined by software. This is explained in Chapter 8, and in the *A500/A2000 Technical Reference Manual*, available from Commodore-Amiga.

backplane The cage or motherboard subsection into which PICs are inserted. The Amiga 2000 and Amiga 3000 computers have integral backplanes, the Amiga 500 and Amiga 1000 computers require add-on backplane cages for Zorro II compatibility.

burst A short name for Multiple Transfer Cycle mode. Essentially, within one full Zorro III cycle there can be any number of Multiple Transfer Cycles. Each full cycle has a complete 32 bit address supplied and a complete 32 bit datum transferred. Each burst cycle supplies only the 8 bit page address, but transfers a complete 32 bit datum faster than the standard full cycle would allow.

bus cycle One complete bus transaction, indicated by the assertion of least one cycle strobe For any single bus cycle, there is one address, one data value, one data direction, and one cycle type in effect.

bus hogging When a bus master takes over the bus for an undue amount of time. The Zorro II bus leaves it completely up to the individual PIC to avoid bus hogging; the Zorro III bus schedules PICs with the bus controller to evenly distribute the bus load.

bus starvation When a master can't get access to the bus, it is said to be starved. On the Zorro II bus, two busy masters can completely starve a third master. Complete starvation is impossible on the Zorro III bus, though a bus hogging Zorro II card can cause similar symptoms.

byte A collection of eight signals into a logical group, and the smallest independently addressable quantity on the Zorro bus.

clock A free running signal driven at a fixed frequency to the bus, used mainly for clocking state machines on Zorro II cards.

cool An unreachable goal for some, a way of life for others. The obvious example of this latter category being Dave Haynie, pictured at right.

cycle strobe A bus signal that defines the boundary of a bus cycle; the Zorro II and Zorro II modes on a Zorro III bus each have their own cycle strobes. The current bus master always asserts the cycle strobes.

data The contents of a memory location. The main purpose of a bus cycle is to transfer data between two locations. The Zorro II bus is based on a 16 bit data path, the Zorro III bus is based on a 32 bit data path.

DMA Direct Memory Access; devices that have direct access to Zorro III slaves are said to have DMA capability. These devices are also called masters.

DMA latency This is the time between a bus request and a bus grant as seen by a PIC wishing to become bus master.

device A PIC, eg, a Zorro bus master or bus slave.

grant The result of an arbitrated set of requests is a single grant; there are grants given for both the bus and quick interrupts.

Guinness Attitude adjustment tonic, from Ireland. Said by some to be vital for sanity, if not normal human life.

hidden cycles Cycles that occur on the local bus of a system, but can't be seen by devices on the expansion bus.

high A signal driven to a logical +5V state is said to be high.

interrupt An asynchronous line driven by a PIC to notify the CPU of some event, usually some hardware event governed by that PIC.

local bus The main system bus of an Amiga computer is called the local bus. In general, the main CPU, video chips, chip memory, and any other built-in resources are on the local bus. The bus controller sits on both the local and expansion buses and manages the communications between them.

longword Based on the Motorola conventions, a longword is equal to 4 bytes.

low A signal driven to a logical +0V state is said to be low.

master The device currently generating addresses for the expansion bus. There is only one master on the bus at a time, this being insured by the bus arbitration logic. The master also drives data on writes, the read, cycle, and data strobes, and several other signals.

motherboard The main system circuit board for any Amiga computer. Resources on the local bus of a machine are often called motherboard resources.

negated The inactive state of a signal, regardless of its logic sense.

nybble A collection of four bits; one half of a byte. AUTOCONFIG® ROMs are physically nybble-wide.

paragraph A sequence of closely related sentences, generally expressing and supporting one succinct idea. This term has no special computerese meaning in any rational, modern system.

PIC Plug In Card. Any Amiga expansion card is called a PIC for short.

request Asking for the use of some resource; the Zorro III bus has two kinds of requests, bus requests and quick interrupt requests.

slave The device currently responding to the address on the expansion bus. There is only one slave on the bus at a time; an error is signalled by the bus collision detect logic if multiple slaves respond to the same address. The slave also drives data on reads, the transfer acknowledge strobe, and several other signals.

slot A physical port on a Zorro backplane, which supplys independent /SLAVE$_N$, /BR$_N$, and /BG$_N$ lines, chained /CFGIN$_N$ and /CFGOUT$_N$ lines, and is mechanically manifested as a 100 pin single-piece connector.

termination Circuitry attached to a bus signal in order to minimize annoying analog things like ringing, reflections, crosstalk, and possibly random logic conditions which can arise when a bus is undriven.

timeout A bus cycle terminated by the bus controller instead of by a responding slave device. If no slave responds to a bus cycle within a reasonable time period, the bus controller will terminate the cycle to prevent lockup of the system.

tri-state A signal driven to a high impedence condition is said to be tri-stated.

word Based on the Motorola conventions, a word is equal to 2 bytes.

Zorro The name given to the Amiga bus specification. "Zorro I" refers to the original design for A1000 backplane boxes, "Zorro II" refers to the modification to this specification used for the A2000 and compatible backplanes, and "Zorro III" refers to the Zorro II compatible bus specification first used in the Amiga 3000 computer.

A.3 Zorro III Implementations

Functionally, there are two possible implementation levels in existance for the Zorro III bus. All of the features described in this focument are required for a full compliance Zorro III bus. However, the original Amiga 3000 computers were shipped with a bus controller that supported only a subset of the Zorro III specification published here. This is, however, upgradable.

The A3000 implementation of the Zorro III bus is driven by a custom controller chip called **Fat Buster**. The specification of this chip and the A3000 hardware are fully capable of supporting the complete Zorro III bus, but the initial silicon on Fat Buster, called the Level 1 Fat Buster, omits some features. Missing are:

- Support of Multiple Transfer Cycles.
- Support for Zorro III style bus arbitration.
- Support for Quick Interrupts.

The Level 2 version of Fat Buster has been in testing for some time at Commodore in West Chester, PA. Any developers who immediately intend to design PICs supporting these features are urged to contact Commodore Amiga Technical Support/Amiga Developer Support Europe for more information on obtaining samples of this part for use in A3000 systems. These parts are likely to be introduced into production, and available as part of an A3000 upgrade, very soon. All Buster chip revisions "13G" and earlier support the Level 1 features. Buster chip revisions "13H" and later support Level 2 features and improved Level 1 features as well.