

Z X - 4 M o n g o o s e M o t h e r b o a r d
H a r d w a r e S p e c i f i c a t i o n

Revi si on C
7/14/89
Mark D. Ni col

***** CONFIDENTIAL, FOR ZDS INTERNAL USE ONLY *****

THIS DOCUMENT CONTAINS PROPRIETARY AND CONFIDENTIAL INFORMATION. IT IS INTENDED ONLY FOR LIMITED DISTRIBUTION WITHIN ZENITH DATA SYSTEMS AND ZENITH ELECTRONICS CORPORATION.

1. Introduction

1.1 Scope

This document is intended to present the hardware specification for the ZX-4 series *Python/Mongoose/Mamba* motherboard. The motherboard circuitry consists of the CPU, numeric coprocessors, memory, bus controller, bus interface, cache controller, cache interface, and expansion slots.

1.2 System Overview

The ZX-4 motherboard is the primary component in a high performance, 32-bit, multi-tasking, multi-user AT compatible desktop personal computer. It's design employs the latest technology, including high speed microprocessors, custom gate arrays, and surface mount components. Key design features include an operating speed of 20, 25 or 33 megahertz, with a high speed cache memory and a 16 word deep write queue, which together combine to provide near zero wait state performance. System expansion is through an industry standard AT compatible bus which provides compatibility with existing 16-bit and 8-bit hardware and software. Additionally, four Zenith Data Systems proprietary 32-bit bus extensions are provided for high speed system and memory expansion.

1.3 Specifications

CPU:	Intel 80386
Clock Speed:	20 MHz (Python) 25 MHz (Mongoose) 33 MHz (Mamba)
Coprocessor Option:	Intel 80387 Weitek WTL 3167
Memory, Standard:	2 Megabytes (1M for Python)
Max. (1M SIMM):	8 Megabytes
Max. (4M SIMM):	32 Megabytes
Max Addressable:	64 Megabytes
Cache:	16K x 32 direct mapped (optional) 4K x 32 direct mapped (standard) 2K x 32 direct mapped (optional)
Cache hit rate:	95 per cent or greater
Write Queue:	16 locations deep x 66 bits wide
System Expansion:	3 PC-AT compatible slots 4 32-bit ZDS proprietary slots
Circuit Board:	6 layers, SMT and thru-hole

2. Design Constraints

The Central Processing Unit (CPU) used in the ZX-4 is capable of addressing 4,294,967,296 (4 gigabyte) memory locations and 65,535 (64 kilobyte) I/O locations. This is considerably more than the existing PC-AT industry standard of 16,777,216 (16 megabyte) memory and 1024 (1 kilobyte) I/O locations. The ZX-4 extends the existing limits to 67,108,864 (64 megabyte) memory and 4096 (4 kilobyte) I/O locations.

One design objective of the ZX-4 was to limit or eliminate the use of hardware jumpers or dip switches. This was accomplished through software programmability of the custom gate arrays.

The system chassis is identical to the existing Z-386 box and somewhat restricts the amount of on-board memory and expansion capability.

3. Hardware Functionality

The backplane/motherboard contains all of the circuitry for the ZX-4 system with the following exceptions:

- * I/O board
 - DMA controllers
 - Interrupt controllers
 - Refresh controller
 - Real Time Clock w/battery backup
 - System ROM
 - System Control Processor
 - Serial & parallel I/O
- * Disk controller board
- * Video system board
- * Cache sub-system (does not require a bus slot)

Additional components on the motherboard include a standard DIN connector for the detachable keyboard, an input power connector, an auxiliary power connector for a second fan, and six power supply status light emitting diodes for ACOK, DCOK, +12V, -12V, +5V, and -5V.

The 644-28 system/cache controller, which interfaces directly to the 644-29 bus/memory controller and the 644-25 posted write queue, along with an 80386 and a numeric coprocessor form the heart of this state of the art computing engine.

The following sections will describe in limited detail the individual subsections of the ZX-4 motherboard. Refer to figure 1.0 for a block diagram, and figure 2.0 for a component layout of the motherboard.

The BLOCK Diagram of the system goes here.

The Component Layout Diagram goes here.

3.1 CPU

The Central Processing Unit (CPU) of the ZX-4 is the Intel 80386. This is the latest generation of the 80x86 family and provides a true 32-bit internal and external architecture. The 80386 instruction set is a superset of the earlier 8086 and 80286 CPU's, thereby making it compatible with existing software, yet offering a significant performance increase. Additional advantages of the 80386 are dual addressing modes (real and protected), very large addressing space (4G physical, 64T virtual), on-chip memory management unit featuring virtual memory support, paging, and four levels of protection.

The CPU is capable of running in a *pipelined* mode to remove one wait-state in slower memory designs. This mode is not used in the ZX-4 since it complicates system timing, and the high speed cache memory design typically runs with zero wait states already.

A software programmable *slow mode* is provided in the hardware to allow the use of certain speed critical or copy protected software written for slower computers. This mode simply adds memory wait-states to slow the CPU to approximately the speed of an 8MHz 1 wait-state AT class computer.

For information on reference material for the 80386 CPU, see appendix D.

3.2 CoProcessors

To handle high speed numerics processing, the ZX-4 provides two co-processor options, the Intel 80387, and the WEITEK WTL 3167. Accommodation for either one of these chips is provided for in the *Extended Math Coprocessor* (EMC) socket, which is a WEITEK extension of the existing 80387 pinout. The 80387 provides for industry standard high speed numerics processing while the WEITEK provides for a higher performance system. The WEITEK can coexist in the same system as the 80387, provided that a simple daughterboard is added to physically connect both of them to the *EMC* socket.

3.2.1 80387

The Intel 80387 is the latest generation of the 80x87 family and provides a true 32-bit internal and external architecture. The 80387 instruction set is a superset of the earlier 8087 and 80287 NPU's, thereby making it compatible with existing software, yet offering a significant performance increase.

The 80386/80387 interface can be programmed to emulate the industry standard AT scheme, or the Intel direct mode.

For information on reference material for the 80387, see appendix D.

3.2.2 WEITEK

The WTL 3167 is a single chip CMOS implementation of the board level WTL 1167 numeric co-processor. It is not software compatible with the Intel 80x87 family, but does provide 3 to 4 times the performance of the 80387 for speed critical tasks such as CAE/CAD. Many compute intensive software packages already provide support for the chip.

The WTL 3167 coprocessor is a memory-mapped peripheral, meaning that to the 80386 and it's application software, the WTL 3167 appears to be a segment of memory at an upper address. Instructions are executed by performing memory moves to and from the coprocessor.

The WTL 3167 must run synchronously with the CPU, so no speed select option is made available.

For information on reference material for the WEITEK 3167, see appendix D.

3.3 Cache

To provide the highest level of performance available, the system CPU must run with as close to zero-wait states as possible. However, the ZX-4's fast clock speed presents severe design problems for a conventional memory design. For this reason a high speed cache memory has been implemented.

The function of a cache memory system is to provide fast local storage of frequently accessed code and data. The cache system intercepts 80386 memory references to see if the required data resides in the cache. If the data resides in the cache (a hit), it is returned to the CPU without incurring additional wait-states. If the data is not cached (a miss), the reference is forwarded to the system and the data is retrieved from main memory.

An efficient cache will yield a high "hit rate" (the ratio of cache hits to total 80386 accesses), such that the majority of accesses are serviced with zero wait states. The net result is that the wait states incurred by the relatively infrequent miss are averaged over a large number of accesses, resulting in an average of nearly zero wait states per access.

A good hit rate is an essential ingredient of a successful cache implementation, but it is not the only factor for performance consideration. Just as essential are sound cache management policies, including the handling of CPU writes and the preservation of cache coherency.

Another true measure of the success of a cache design is it's ability to meet an individual system cost/performance requirement. Some systems require only a moderate jump in performance while effecting only a minimal cost impact on the overall system. Other systems will want to squeeze every last percent of performance out of an architecture regardless of the cost. The ZX-4, with it's *adaptable architecture*, can easily be tailored to nearly any cost/performance specification.

The ZX-4 cache subsystem is controlled by a custom gate array which resides between the CPU and the rest of the system, and appears to be the CPU to the rest of the system.

The cache memory itself resides on a plug-in card which not only reduces the required motherboard area, but also accommodates the *adaptable architecture* philosophy. For example, the cache can be organized as a 64K direct mapped cache, which is the standard configuration, or by simply changing the plug-in card, a 16K two way set associative cache can be implemented.

3.3.1 Cache Controller

The ZX-4 system/cache controller is incorporated in a custom BiCMOS gate array (644-28). This was required due to the complexity of the design and the limited motherboard space available.

The main function of the system/cache controller is to interface the CPU with the memory subsystems, using either the high speed cache, or the lower speed system memory. However, as it's name implies, the controller contains more than just the cache control logic. It also contains bus arbitration logic, write queue interface control logic, bus interface control logic, bus snoop control logic and processor and coprocessor interface and reset logic.

The system/cache controller is designed to interface directly to the Intel 80386 CPU forming a *local* processor bus. The local bus interface includes the numeric coprocessor and the cache memory subsystem as well as the CPU itself.

The cache memory subsystem interfaces directly to the cache controller. This design makes extensive use of pipelining within each cache cycle to allow for speedups that would otherwise not be possible. The data buffers/latches shown between the processor and the cache memory subsystem are directly controlled by the cache controller. These 74F543 type devices with their latching as well as buffering capability allow the cycle pipelining to take place.

The system/cache controller interfaces to the remaining system hardware through its *system* bus. The system bus interface appears essentially as the 80386 to the system bus. The system bus interface on the system/cache controller includes direct interface to the memory/bus controller. The system bus also interfaces directly to the high performance posted write queue.

When the CPU performs a memory write cycle, a system memory access must occur, since a cache write alone will most likely cause the data to be lost if that location of the cache is subsequently overwritten. This system memory access would normally cause the CPU to slow down substantially since system memory cannot respond without adding additional wait-states.

The ZX-4 potentially eliminates this performance robbing situation by latching all CPU address, data, and control lines into the write queues, and allowing the CPU to continue on with the next instruction. This allows the bus/memory controller to complete the write to system memory concurrently and asynchronously. This *write posting* allows up to sixteen write cycles to be queued to the bus controller before any delay is passed on to the CPU. The posting of writes to the system bus allows them to occur in zero wait states on the local bus, even though they may take much longer to complete on the system bus.

When the CPU requests data, the cache controller first checks to see if the requested data is cached. If so, the data is immediately made available and the cycle terminated. If the data is not cached the controller must pass the request on to the bus/memory controller.

The cache controller will latch all the address, data, and control lines from the CPU into the write queue and assert DOCYCLE. Further processing by the CPU is suspended until the write queue is flushed of any previous write cycles which were still pending and the read cycle can be completed.

The bus/memory controller determines if the requested data resides on an 8-bit, 16-bit, or 32-bit device and issues the appropriate cycles to read a complete double word, since the cache's minimum transfer (line) size is 32-bits. When the 32-bit data is made available the cache controller latches it back into the write queues, delivers it to the CPU, updates the cache, and terminates the cycle.

The system bus arbitration is handled directly by the cache controller freeing the CPU from this task. Since the CPU does not control the system bus arbitration it never needs to enter the hold state. The performance benefit of this is that it allows parallel hardware operation between the local bus and the system bus.

This hardware parallelism is used several different ways. When the system bus has been given away for refresh, DMA or temporary MASTER accesses the CPU may continue running on the local bus, through read or write accesses to the cache, the cache controller registers, or the numeric coprocessor. The pipelining of queued writes to the queue is also one of the factors allowing the parallel bus operation. This parallel operation can continue on both busses up to the point when the CPU must make a system bus read access, or a posted write operation occurs when the write queue is already full. At that time the parallel operation stops and the CPU waits, idling on the local bus until the system bus is returned to it. This parallelism allows *pseudo hidden* refresh, DMA and temporary master accesses.

The parallel operation of the system bus and the local bus means that CPU accesses on the local bus can occur at the same time as temporary master accesses on the system bus. This parallel operation greatly enhances performance since the two separate busses seldom have to wait for each other to complete. There is a problem with the parallel operating busses however. The problem is *cache coherency*.

When both busses are operating simultaneously and the system bus is performing a write to a main memory location that is currently contained in the cache, the cache must be updated in order to make it's data match with main memory. The cache controller directly controls this function through an operation called *bus snooping*.

Bus snooping is the means by which the system/cache controller watches the bus activity at all times, regardless of what other tasks that it or the CPU are involved in. Bus snooping functions by watching the system bus write lines. Whenever a system bus write occurs the cache controller controls the latching of the current bus address and data values into external 74F573 type devices. CPU accesses to the cache are then interrupted, if they are occurring, and the address of the system bus write is checked in the cache to see if that entry is in the cache. If the snoop check finds that the entry was not in the cache than normal operation resumes. If the snoop check finds that the entry was in the cache, then that cache data entry is updated and normal operation can resume. The updating of the cache data entries through bus snooping maintains cache coherency and boosts system performance by allowing the parallel operation of the system and local bus to continue.

The cache controller also controls the reset to both the CPU and the numeric coprocessor. The numeric coprocessor is only reset at power up time. The CPU is reset at power up time and can also be reset individually at any time through the use of the RC input to the cache controller. The reset pulse to the CPU from the cache controller will always be a minimum of 16 cycles of CLK2. The length of the numeric coprocessor reset will be as long as the power up reset pulse.

The local bus, or processor ready (-PRDY) signal is controlled solely by the system/cache controller. It integrates the ready signals from all possible cycles. These include accesses to the coprocessor, to the internal cache controller registers, to the cache data RAM or accesses to the system bus.

3.3.2 Cache Controller Programmability

The system/cache controller is software programmable to configure or report on various aspects of the cache system. The I/O ports are defined as follows:

- CMCR** - *Cacheable Memory Control Register*
Used to program the areas of cacheable system memory.
- CSSR** - *Cacheable System Status Register*
Used to program miscellaneous status inputs.
- CTTR** - *Cache TAG Test Register*
Used to read the status of the cache tag match bits.
- CIDR** - *Cache Identification Register*
Used to read a four bit cache ID code containing the cache size and organization.

These ports are decoded externally of the controller itself to allow flexibility when used in other systems. The actual addresses are detailed in section 7.

Accommodations are provided to extend the address range of the cache controller or achieve finer granularity than that provided by the programmable registers. This is done by populating an address decoder PAL and driving the -NCC0 and/or the -CC0 lines.

See appendix D for additional information on the system/cache controller.

3.3.3 Cache Timing

The advantage of a cache memory system would be severely compromised if the the cache system was unable to respond without incurring additional memory wait-states. The 80386 is capable of executing a memory access in as little as two processor clock cycles, which at 33MHz clock rates translates to a window of only 60 nanoseconds. Clearly, the timing of the cache subsystem is extremely critical to the overall performance of the entire system. The following diagrams illustrate the details of the cache system timing.

The cache timing diagrams go here.

3.3.4 Cache Interface

The ZX-4 cache interface consists of latches and buffers for holding and multiplexing address and data, controlled by the system/cache controller. The cache card is mounted on a dedicated 100 pin connector allowing for different sizes and configurations of cache to be easily connected. The cache connector contains a 4-bit cache ID which, when driven by the different versions of cache cards, signal to the cache controller which version of cache memory is installed.

Section 3.3.5 defines the 4-bit ID codes and figure 4.0 specifies the cache connector pinout.

3.3.5 Cache Options

The plug-in configuration of the cache memory system and the *adaptable architecture* of the system/cache controller allow extreme flexibility in the structure of the cache subsystem. Different memory sizes and organizations allow the cache subsystem to be tailored to specific performance needs or applications. With these choices almost every cost/performance goal will be attainable.

The specific cache sizes and organizations specified to date are:

ID#	Size x 32	Organization	Total memory
F	2K	<i>direct mapped</i>	8K
D	4K	<i>direct mapped</i>	16K
9	16K	<i>direct mapped</i>	64K
0		<i>no cache installed</i>	

In addition to the direct mapped organization currently implemented, the cache controller can also support two way set associative caches. The two way set associative cache can be thought of as two direct mapped caches in parallel. The performance advantage over a direct mapped cache is that all identical page offsets map to two cache locations instead of one, reducing the potential for "*thrashing*".

Thrashing occurs when two or more data items want to occupy the same location in cache, and they repeatedly overwrite each other causing a hit rate of zero.

The two way set associative cache can be more efficient and result in a slightly higher hit rate in some situations. For instance, this option may become more attractive in a *UNIX* type environment.

The cache connector pinout goes here.

3.4 Write Queue

The write queue is implemented as a matched pair of custom CMOS gate arrays (644-25). It is fundamentally a 66 bit wide, 16 location deep, transparent latch/buffer between the cached processor and the system/bus controller. It will bi-directionally buffer address, control signals, and data between these devices, and provide externally controlled cached processor write command posting. Control lines provide the ability to latch and enable data, addresses, and commands separately in either direction, and to force all byte enables active.

The write queue is a logical extension of the single *posted write buffer* scheme used by competitive cache controllers. With any *write through* cache architecture, such as the ZX-4, all processor writes (cache hit or miss) must be broadcast to the system. Not only is the CPU slowed to the system bus speed during each write, but the system bus is also engaged by the CPU during each write.

An empty write queue will store up to 16 write commands (address, control, and data) issued by the cached processor without adding wait states, in a first-in, first-out (FIFO) format. When the queue is full, the processor has to wait until the first command that was posted has finished.

Studies have shown that software typically exhibits roughly 15 to 20 percent total write cycles, and that many writes occur in bursts of more than one. In a typical system, most of the consecutive writes will occur during block moves and stack manipulations. Block instructions, such as programmed disk IO, are usually infrequent but rather lengthy. These operations will quickly fill the queue, causing the CPU to wait. Stack manipulations occur during hardware and software initiated interrupts, and during high level language procedural calls. These operations will generally cause at least two to three words to be pushed to the program stack. Assuming that these locations have already been cached, the queue should never fill, allowing the CPU to continue processing while the memory writes are handled concurrently by the bus/memory controller. As long as all the queued writes can be processed by the bus/memory controller before any cache misses occur, the CPU will incur no additional wait-states.

If a cache miss should occur immediately following a queued write, the benefit of having posted the write is voided, since the queue must be flushed and the CPU halted until the cache miss can be serviced. This is necessary since the queue contents may affect the data to be retrieved. In addition, the bus can not be granted to a temporary MASTER if the queue is currently in the middle of a locked cycle. This will usually occur during a semaphore Read-Modify-Write (RMW) sequence when the location is a cache hit.

3.5 Memory

The system memory of the ZX-4 is organized as 8 banks of either 1 or 4 megabyte *Single In-line Memory Modules* (SIMM's). These modules include 32 bits of data and 4 parity bits, one for each of the 4 bytes. The banks are all socketed and are easily field upgradeable.

The bus/memory controller gate array provides all the necessary control signals to read, write, refresh, and error check the system memory. The design of the controller allows the memory to be intermixed with 1 megabyte and 4 megabyte SIMM's in any combination. Any additional memory, up to the maximum of 64 megabytes, must reside in any of the three memory expansion bus slots.

Programmable registers within the bus/memory controller allow each bank of memory to be dynamically sized and mapped. The first bank of memory on the motherboard can also be programmed to respond to *extended* or *EMS* accesses.

3.5.1 Memory Controller

All of the logic and control for the ZX-4 memory subsystem is contained in one custom BiCMOS gate array. This array, which is ZDS part number 644-29, also controls the system expansion bus and is packaged in a 136 pin quad flat pack.

The bus/memory controller contains programmable registers to configure the motherboard memory. Each bank has a register which allows it's starting address and depth to be programmed. These registers allow 1 megabyte banks to be placed on any 1M boundary, and 4 megabyte banks to be placed on a 4M boundary.

On power-up, the first bank will be hardware programmed with a depth of 1 megabyte starting at 000000H, and all other banks will be disabled. The ROM will then program the next bank's starting address and attempt to size any existing memory. If memory is found in the second bank it's depth is programmed and the process is repeated on the next bank, until all contiguous memory has been found.

The configuration of the first 1 megabyte is programmable as to how *base*, *EMS*, and *extended* memory is allocated. *Base* memory can be isolated as the first 256K, 512K, 640K or disabled completely. The reserved memory space from 0A0000H to 0DFFFFH, the *slushware* space from 0E0000 to 0FFFFFFF, and any base memory not used, can be mapped to *EMS* board 0, mapped to *extended*, or disabled. See section 6 for details of the system memory map, and appendix D for more information on the bus/memory controller.

System memory cycles are started and terminated under the direct control of the system/cache controller. It issues a DOCYCLE signal to indicate the start of a cycle to the system bus.

When the bus/memory controller determines that the addressed data resides in the local memory, it immediately starts a standard 32-bit memory cycle. The specific memory bank requested is encoded in three bank-enable lines and externally decoded to generate the RAS and CAS signals, which together with the multiplexed address lines determine which location in memory is accessed.

If the memory is located in one of the system expansion slots, the controller must issue the appropriate cycle or cycles needed to accomplish the memory access. Section 3.6.1 discusses this in more detail.

The termination of a system memory cycle occurs when the bus/memory controller issues a RDYBUS signal. This signal must be issued in the proper phase of the cycle with adequate setup and hold time from the rising CLK2 edge that represents the end of the cycle.

3.5.2 Memory Controller Programmability

The bus/memory controller is software programmable to configure or report on various aspects of the system. The I/O ports are all protected with PRIVY and defined as follows:

- SLOW** - *SLOW Mode Register*
Programs the speed to roughly 8MHz/1WS AT speed.
- SPRE** - *ScratchPad Ram Enable (PRIVY)*
Enables access to protected I/O ports and memory.
- FMMR** - *First Megabyte Mapping Register*
Contains the starting address of any extra memory which is being mapped to extended.
- FMCR** - *First Megabyte Configuration Register*
Contains the amount of the first megabyte used as base, mapped to EMS, or mapped to extended.
- PEAR** - *Parity Error Address Register*
Contains information concerning the address of the last parity error experienced.
- 32SP** - *32-bit Speed Register*
- 16SP** - *16-bit Speed Register*
- 8SP** - *8-bit Speed Register*
Holds the number of wait-states (0-15) which will be added to the basic I/O and memory cycles.
- RISP** - *Refresh/INTA Speed Register*
Holds the number of wait-states (0-15) which will be added to refresh and interrupt acknowledge cycles.
- REVR** - *Revision Register*
Contains the revision number of the gate array.
- EMSO** - *EMS Page 0 Register*
Contains the EMS mapping register for EMS page 0.
- OBCR** - *Bank 0 Configuration Register*
- 1BCR** - *Bank 1 Configuration Register*
- 2BCR** - *Bank 2 Configuration Register*
- 3BCR** - *Bank 3 Configuration Register*
- 4BCR** - *Bank 4 Configuration Register*
- 5BCR** - *Bank 5 Configuration Register*
- 6BCR** - *Bank 6 Configuration Register*
- 7BCR** - *Bank 7 Configuration Register*
Contains the starting address and memory depth for each of the 8 banks of motherboard memory.

3. 5. 3 Memory Cycle Timing

The local system memory located on the motherboard can be accessed with the addition of one 32-bit wait-state programmed into the bus/memory controller, for a total of five processor clock cycles from a valid DOCYCLE.

See figures 5.0 and 5.1 for details of the 32-bit memory timing.

3. 5. 4 RAM modules

Single In-line Memory Modules (SIMM) are employed to house all 32-bit memory in the ZX-4 system since they are both convenient and economical of board space. The SIMM s are organized as 32-bit wide and either 256 kilobyte or 1 megabyte deep. This organization translates to either 1 megabyte or 4 megabytes of memory per bank. In addition each SIMM carries 4 bits of parity data, 1 bit for each of the 4 bytes. Use of the SIMM s allows for easy field upgrades or replacements.

The 1 megabyte SIMM s are packaged as multi-layer, surface mount PCB' s containing eight 256K x 4 DRAMs and four 256K x 1 DRAMs. The 4 megabyte SIMM s are similar, with the 256K parts being replaced by 1 megabyte parts. Interconnection to the motherboard is through a 72 pin card edge connector. The outline of the SIMM is shown in figure 5.0, however the component layout may differ.

The SIMM module diagram goes here.

3.5.5 Error Detection

All the local system memory on the ZX-4 motherboard is parity checked to detect single bit errors. For every byte written to memory, a bit is generated to produce odd parity, and is stored along with the data. When that byte is subsequently read from memory, it's parity bit is also retrieved. If upon examination, odd parity has not been maintained, a parity error is generated.

To test the parity circuitry, parity errors can be forced to occur by enabling the DISPG (disable parity) line. This causes even parity to be generated when writing to memory. Since odd parity is still used to check data read from memory, a parity error is generated, provided that the parity circuitry is performing correctly.

To report the error, the bus/memory controller contains an I/O port which will contain the six high order address lines, and the two low order address lines of the offending byte. The high order address lines decode which bank of memory contains the error, and the two low order address lines, decode which byte is at fault.

Since the system memory can be dynamically mapped and sized, the system can be made relatively fault tolerant. If an error is detected during the power up sequence, that bank of memory can be disabled, and the next available bank mapped into it's place. This allows the user to continue on until that bank of memory can be repaired or replaced.

3.6 Expansion Bus

The expansion bus for the ZX-4 consists of three industry standard AT compatible slots for the interconnection of conventional circuit cards. In addition, four 32-bit Zenith Data Systems proprietary slots are provided for the inclusion of extra memory beyond that provided for on the motherboard, and the interconnection of high speed peripherals such as disk controllers and video boards.

The Zenith 32-bit slots consist of the standard 62-pin PC bus card edge connector, the standard 36-pin AT bus card edge connector, and another 62-pin card edge connector containing the upper 16 data bits, four byte enable signals for addressing, and other signals required for system control.

The slot assignment is as follows: slots 1 through 3 are AT slots, and 4 through 7 are 32 bit slots. Additionally, slots 5 through 7 are defined as 32-bit memory card slots since they alone contain an additional signal line for memory decoding.

Refer to figure 2.0 for specific slot assignments and locations.

3.6.1 Bus Controller

All of the logic and control for the ZX-4 bus is contained in one custom BiCMOS gate array. This array, which is ZDS part number 644-29, also controls the memory subsystem and is packaged in a 136 pin quad flat pack.

The primary function performed by the bus controller is the access and management of all data, either memory or I/O, which reside on the system expansion bus. This requires the generation of all command and control signals necessary to access data in any combination of 8, 16, 24, or 32 bits. The controller must generate the correct signals to build additional cycles and swap or reroute data paths when there is a data path width mismatch.

The monitoring of source and destination data path widths is accomplished using the AT signals M16, I16, HOLDA and MASTER, and the ZDS signals M32, I32, and MSTR32. If, for example, the CPU performs a 32-bit read of an 8-bit PC style memory card, the bus controller will respond by building four consecutive PC type bus cycles, and routing the bytes from the lower 8-bit bus up to the appropriate 32-bit byte. This procedure does require substantially more time to perform than a 32-bit access to a 32-bit card, but is necessary to maintain compatibility with existing hardware.

3. 6. 2 Bus Timing

The timing for the expansion bus is designed to be fully compatible with the existing AT bus timing for all 8-bit and 16-bit cycles. Since the bus controller gate array could be used in other systems, with different clock speeds, the number of wait states used is programmable for all 8-bit, 16-bit, 32-bit, refresh, and interrupt acknowledge cycles. For details on the memory controller programmability see section 3.5.2 .

The 32-bit timing is used for all local memory accesses or whenever the source device is recognized as 32-bit and the destination device responds with either M32 or I32. The 32-bit bus can run as fast as possible since no additional cycles are needed, no byte re-routing is necessary, and no established standard must be adhered to. The actual bus speed is much slower than the optimal two cycles generated by the CPU, due to delays caused by slow peripherals, bus buffering, and synchronization of the bus controller to other system components.

The diagrams on the following pages illustrate the basic bus timing for the AT cycles and the 32-bit cycles.

The 32-bit timing diagrams go here.

The 8/16-bit timing diagrams go here.

3.6.3 Signal Definitions

Signal definitions for the AT portions of the bus are identical to the industry standard. These signals and the ZDS proprietary signals appear in figure 7 and are defined below.

3.6.3.1 PC AT Definitions (ISA)

sA0-19 These bi-directional signals address memory or I/O devices within the system. They form the low order 20 bits (1 megabyte) of the 26 bit system. These lines are enabled onto the bus while sALE is high and are latched on the falling edge of sALE. These signals are driven by the CPU or DMA controller and may also be driven by an expansion bus MASTER.

sLA17-23 These bi-directional signals (*Latchable Address*) decode memory within the system which must respond with 0 or 1 wait state, or which is addressed above 1 megabyte. They give the system up to 16 megabytes of addressability. These lines are enabled onto the bus while sALE is high but are *not* latched by sALE. These signals are driven by the CPU or DMA controller and may also be driven by an expansion bus MASTER.

sD0-D7 These bi-directional signals are the low 8 bits of the 16-bit system data bus. They should be used exclusively by all 8-bit devices to transfer data. These signals are driven by the CPU and may also be driven by an expansion bus MASTER. An expansion card should have no more than two Low Power Schottky loads (0.8mA low, 400uA high, 50pF) on this bus.

sD8-D15 These bi-directional signals are the high 8 bits of the 16-bit system data bus. They should be used by all 16-bit devices to transfer the high byte of the data word when -sBHE is asserted. These signals are driven by the CPU and may also be driven by an expansion bus MASTER. An expansion card should have no more than two Low Power Schottky loads (0.8mA low, 400uA high, 50pF) on this bus.

-sBHE When low, the *Byte High Enable* output signal indicates that the high byte of the sD0-15 data bus should transfer data on boards that support the 16-bit data bus.

- sM16 This bi-directional signal (*Memory is 16 bits wide*), notifies the system that the addressed memory device is capable of transferring 16 bits of data at once. When this line is pulled low at the start of a memory read or write, the standard 1 wait state 16-bit memory cycle is run, avoiding the 16-bit to 8-bit conversion cycle. This signal should be driven low by an open collector driver capable of sinking 20mA. During *Hold Acknowledge* this signal can be asserted for a ZX-4 local memory access.
- sI16 This input signal (*I/O is 16-bits wide*), notifies the system that the addressed I/O device is capable of transferring 16 bits of data at once. When this line is pulled active at the start of an I/O read or write, the standard 1 wait state 16-bit I/O cycle is run, avoiding the 16-bit to 8-bit conversion cycle. This signal should be driven low by an open collector driver capable of sinking 20mA.
- sIOCHCK The *I/O Channel Check* input signal is used to signal the CPU about parity or other serious errors on boards plugged into the expansion bus. This signal should be driven low by an open collector driver capable of sinking 20mA.
- sIOCHRDY The *I/O Channel Ready* input signal is pulled low (not ready) by a memory or I/O device which cannot respond quickly enough to a standard cycle. Bus cycles are extended by an integral number of cycles until sIOCHRDY is released. Any slow device using this line should drive it low immediately upon detecting it's valid address and a READ or WRITE command. This line should be held low for no longer than 2.5 microseconds. This signal should be driven low by an open collector driver capable of sinking 20mA.
- sOWS The *0 Wait State* signal informs the system that the addressed device can perform the required cycle without adding the standard wait states. The decode logic used to drive this line should include the device address and the relevant command. The ZX-4 -sOWS line is sampled at the rising edge of each sCLK after a valid command. This signal should be driven low by an open collector driver capable of sinking 20mA.

sALE	When high, the <i>Address Latch Enable</i> signal indicates that a valid address is present on the sLAXx address lines. The system sA0-19 lines are latched with the falling edge of sALE. The sLAXx address lines, or any decodes developed from them, should also be latched with the falling edge of sALE. This line is driven high (active) during all DMA or MASTER cycles.
-sMEMR	When low, the <i>MEMory Read</i> signal commands a memory device to send data to the data bus. This signal is active over the entire memory range, and may be driven by a temporary MASTER.
-sMEMW	When low, the <i>MEMory Write</i> signal commands a memory device to accept data from the data bus. This signal is active over the entire memory range, and may be driven by a temporary MASTER.
-sSMEMR	When low, the <i>Standard MEMory Read</i> signal commands a memory device to send data to the data bus. This signal is active only when an address in the range of 00000H to FFFFFH (low 1M) has been decoded. This signal is derived from -sMEMR.
-sSMEMW	When low, the <i>Standard MEMory Write</i> signal commands a memory device to accept data from the data bus. This signal is active only when an address in the range of 00000H to FFFFFH (low 1M) has been decoded. It is derived from -sMEMW.
-sIOR	When low, the <i>I/O Read</i> signal commands an I/O device to send data to the data bus. This signal may be driven by a temporary MASTER.
-sIOW	When low, the <i>I/O Write</i> signal commands an I/O device to accept data from the data bus. This signal may be driven by a temporary MASTER.
sRESET	This active high output signal resets the hardware during power-up or following a power failure.
sCLK	This output signal defines the 8MHz bus clock which is used for synchronization.
sOSC	This signal is a 14.31818MHz clock from the system

I/O board.

- sDRQ0-3, 5-7 The *DMA ReQuest* signals are asynchronous inputs used to request DMA service from the DMA subsystem, or to gain control of the system bus from the CPU. They are prioritized, with sDRQ0 having the highest priority and sDRQ7 the lowest. The request is made when a line goes from a low to a high state and remains there until the appropriate -sDACKx (DMA Acknowledge) line goes active. The channels sDRQ0 through sDRQ3 perform 8-bit DMA transfers, while sDRQ5 through sDRQ7 perform 16-bit transfers.
- sDACK0-3, 5-7 The *DMA ACKnowledge* signals are used to acknowledge DMA requests. These signals are active low.
- sAEN When active (high), the *Address ENable* signal indicates that the DMA controller has control of the system bus. When inactive, the CPU or other bus MASTER has control of the bus. This signal is often used to disable devices which must not respond during DMA cycles.
- sMASTER This input signal is used with an sDRQx line to gain control of the system bus. A temporary bus MASTER pulls this line low when the appropriate -DACKx line is made active, signaling that a MASTER request is granted. The system address, data, and control lines are floated, allowing the MASTER to begin controlling them one full sCLK period after -sMASTER becomes active. At least one more full sCLK period should be allowed after putting a valid address on the bus before activating any of the control lines. Upon release, the control lines should be driven inactive, then all lines should be floated. The -sMASTER and sDRQx lines can then be made inactive. This signal should be driven low by an open collector driver capable of sinking 20mA.
- sT/C When high, this output signal indicates that the *Terminal Count* of a DMA operation has been reached. It should be decoded with the appropriate -sDACKx line for proper operation.

sIRQx

These *Interrupt ReQuest* signals interrupt the CPU asynchronously to request some service. The sIRQx lines are prioritized, in the following decreasing order:

sIRQ0
sIRQ1
sIRQ9, sIRQ10, sIRQ11,
sIRQ12, sIRQ14, sIRQ15
sIRQ3
sIRQ4
sIRQ5
sIRQ6
sIRQ7

The interrupt is recognized when a line goes from a low to a high state and remains there until the appropriate interrupt service routine is executed.

3. 6. 3. 2 ZDS Definitions

- sD16-D31** These bi-directional signals are the high 16 bits of the 32-bit system data bus. They should be used by all 32-bit devices to transfer the high half of the double word when -sM32 is asserted. These signals are driven by the CPU and may also be driven by an expansion bus 32-bit MASTER. An expansion card should have no more than two Low Power Schottky loads (0.8mA low, 400uA high, 50pF) on this bus.
- sLA24-25** These output signals (*Latchable Address*) decode memory within the system which is addressed above 16 megabyte. They give the system up to 64 megabytes of addressability. These lines are enabled onto the bus while sALE is high but are *not* latched by sALE. These signals are driven by the CPU or DMA controller and may also be driven by an expansion bus 32-bit MASTER.
- sLA31** Extended address line used to decode the ROM space which exists at the top of the four gigabyte memory space.
- sBYTEN0-3** The *Byte Enable* bi-directional signals directly indicate which byte of the 32-bit data bus are involved with the current transfer. These signals are driven by the CPU and may also be driven by an expansion bus 32-bit MASTER.
- sMPAGE** This 32-bit memory control output signal informs when the *Row Address Strobe* (RAS) should be asserted on ZX-4 memory boards.
- sMCYCLE** This 32-bit memory control output signal informs when the *Column Address Strobe* (CAS) should be asserted on ZX-4 memory boards.
- sMEMR32** When low, the *32-bit MEMORY Read* signal commands a 32-bit memory device to send data to the data bus. This signal is active over the entire memory range, and may be driven by a 32-bit MASTER.
- sMEMW32** When low, the *32-bit MEMORY Write* signal commands a 32-bit memory device to accept data from the data bus. This signal is active over the entire memory range, and may be driven by a 32-bit MASTER.

- sMSTR32 The *MaSTeR32* signal is generated by any temporary bus MASTER which is capable of handling a full 32-bits of data. This signal tells the bus controller that a 32-bit MASTER will be driving the AT and the 32-bit bus lines.

- sMB2 This bi-directional signal (*Memory is 32 bits wide*), notifies the system that the addressed memory device is capable of transferring 32 bits of data at once. When this line is pulled low at the start of a memory read or write, the standard 32-bit memory cycle is run, avoiding the 32-bit to 16/8-bit conversion cycle. During *Hold Acknowledge* this signal can be asserted for a ZX-4 local memory access. This signal should be driven low by an open collector driver capable of sinking 20mA.

- sI32 This input signal (*I/O is 32 bits wide*), notifies the system that the addressed I/O device is capable of transferring 32 bits of data at once. When this line is pulled active at the start of an I/O read or write, the standard 32-bit I/O cycle is run, avoiding the 32-bit to 16/8-bit conversion cycle. This signal should be driven low by an open collector driver capable of sinking 20mA.

- sFASTRDY The *FAST ReaDY* input signal is pulled low (not ready) by a memory or I/O device which cannot respond quickly enough to a standard 32-bit cycle. Bus cycles are extended by an integral number of cycles until sFASTRDY is released. Any slow device using this line should drive it low immediately upon detecting it's valid address and a READ or WRITE command. This signal should be driven low by an open collector driver capable of sinking 20mA.

- sHOLD The *HOLD* input signal, which is generated by the system I/O card, indicates a request to gain control of the system bus.
- sHOLDA When asserted, the *HOLD Acknowledge* output signal indicates that the CPU has relinquished control of the system bus to a temporary master.
- sMEMDECx The *MEMory DECode* signals supply a unique address line to each of three 32-bit slots on the backplane. They are used to provide distinctive I/O address locations for otherwise identical ZDS memory boards.
- NOTE: *The absence of this line in slot 4 disqualifies it as a memory expansion slot.*
- sKBCLOCK The *KeyBoard CLOCK* line routes the signal from the backplane keyboard connector to the system command processor (SCP) on the system I/O board where it is processed.
- sKBDATA The *KeyBoard DATA* line routes the signal from the backplane keyboard connector to the system command processor (SCP) on the system I/O board where it is processed.
- sIRQ8 The *Interrupt ReQuest 8* line carries the signal from the *Real Time Clock* interrupt. This signal is not used on the ZX-4.
- sIRQ13 The *Interrupt ReQuest 13* line routes the signal from the *Numeric Coprocessor Error* interrupt on the motherboard to the interrupt controller on the I/O board.
- sINTR The *INTerrupt Request* input signal, which is generated by the system interrupt controller on the I/O card, indicates to the CPU, a request for interrupt service.
- sINTA When asserted, the *Interrupt Acknowledge* output signal indicates that the CPU has acknowledged the interrupt request.
- sNMI The *Non-Maskable Interrupt* input indicates to the CPU a request for interrupt service which cannot be masked by software. This is generally used for memory parity or other fatal system error.

sRC	The <i>Reset Cpu</i> signal is an input from the system I/O board which causes the 80386 CPU to be reset without disturbing the rest of the system hardware.
sA20GATE	The <i>Address line 20 GATE</i> signal is an input from the system I/O board which when inactive, causes the 80386 CPU to force address line A20 inactive. This is necessary in order to make the system look like an 8088 based system in which memory accesses beyond FFFFFH caused the processor to wrap back around to 00000H.
-sDISPG	The <i>DISable Parity Generation</i> signal is used for testing the memory parity hardware. It disables the generation of the correct parity bit during a WRITE, causing the successive READ to generate a parity error.
-sSLUSH	This input signal controls the write protection of the <i>Slushware</i> memory and is controlled by the system I/O board. When this signal is asserted (low), the slushware RAM is writeable.
sSPRAMEN	The <i>Scratch Pad RAM ENable</i> signal controls the write protection of the scratchpad RAM, and is controlled by the system I/O board. When this signal is asserted (high), the scratchpad RAM is writeable.
sPCLK	The <i>Processor CLoCK</i> signal is a CPU speed reference clock signal used for synchronization of peripherals.
sDCOK	This signal becomes invalid when the power supply DC voltage becomes unstable.
sACOK	This signal becomes invalid when the power supply AC voltage becomes unstable.
-sWRT	Not used on ZX-4, reserved for future use.
-sWRDIS	Not used on ZX-4, reserved for future use.
sRESx	Not used on ZX-4, reserved for future use.

The Bus Pinout Diagram goes here

4. Mechanical

The ZX-4 motherboard is physically the same size as the existing Z-386 backplane, and resides in the same chassis.

5. Jumper Options

The ZX-4 was designed with the intention of reducing or eliminating the number of jumpers and switches needed to program the hardware. While this was considered successful, one unpopulated jumper location still exists which can be used to hide an 80387 coprocessor from the 80386 CPU. This jumper (J101) is primarily used for testing purposes only, to disable the 80387, without having to physically remove it.

6. Memory Map

The CPU can address 64 megabytes of memory in its protected address mode. In real-address mode, it can address 1 megabyte of conventional (PC-compatible) memory. The first 640K of this range contains dynamic user memory. The remaining address range (up to 1 megabyte) is reserved for specific controllers (video, hard disk, etc.) and a window through which access to *expanded memory* (EMS) occurs. Access to memory beyond the CPU's 1 megabyte address range limit can only be accomplished through the 80386's protected address mode or the *expanded memory specification* (EMS) option.

The BIOS ROM on the I/O card is an evolution of the current Zenith AT ROM. The ROM in the ZX-4 copies itself into RAM at boot-up time, then disables writes to that area of RAM via the SCP. This is called *slushing* the BIOS or *slushware*. *Slushware* allows faster execution of BIOS calls, since the RAM can be accessed faster than the ROM. Also, because the 8-bit wide ROM is copied into the 32-bit wide cacheable system memory, only one ROM is required instead of four, resulting in space, power and cost savings. If there is an OEM ROM present, its code will also be *slushed*.

During coprocessor accesses, the system/cache controller will not pass the cycle out to the system, to avoid any unwanted response from the ROM or other device.

The following provides an overall map of the system's addressable memory. All addresses are in hexadecimal.

Address Range	Size	Description
C0000000-C1FFFFFF	32M	optional WEITEK coprocessor
*80000000-8003FFFF	256K	system ROM
:	:	:
<i>top of addressable system memory</i>		
00100000-03FFFFFF	63M	<i>extended memory</i> (1M - 64M)
:	:	:
<i>top of first megabyte of system memory</i>		
000F1000-000FFFFF	60K	System BIOS ROM <i>slushware</i>
000F0000-000F0FFF	4K	Scratchpad RAM
000E0000-000EFFFF	64K	Optional ROM <i>slushware</i>
000D0000-000DFFFF	64K	EMS window
000CA000-000CFFFF	24K	Open
000C8000-000C9FFF	8K	Hard disk BIOS ROM
000C0000-000C7FFF	32K	EGA video BIOS ROM
000B8000-000BFFFF	32K	CGA video RAM
000B0000-000B7FFF	32K	MDA video RAM
000A0000-000AFFFF	64K	EGA video RAM
00080000-0009FFFF	128K	System base memory (640K)
00040000-0007FFFF	256K	System base memory (512K)
00000000-0003FFFF	256K	System base memory (256K)

*duplicated every 256K to the 4 gigabyte top of memory except where the WEITEK coprocessor is decoded.

7. I/O Map

The following list defines the ZX-4 motherboard I/O port addresses, many of which are implemented in the bus/memory controller. When an extended I/O address (beyond the standard 10 bits) is recognized by the bus/memory controller or the system/cache controller, and it is decoded as an on-chip register, the cycle is not forwarded on to the rest of the system. This eliminates the possibility of a standard I/O device erroneously responding to an extended I/O request.

PORT	RW	ACTION	DESCRIPTION
00EC	W	Set	80387 interface Intel direct mode.
00ED	W	Clear	80387 interface Intel direct mode.
00F0	W	Clear	80387 coprocessor BUSY latch.
00F4	W	Set	CPU <i>slow</i> mode.
00F5	W	Clear	CPU <i>slow</i> mode.
00F9	W	Clear	Scratchpad RAM enable. (PRIVY)
00FB	W	Set	Scratchpad RAM enable. (PRIVY)
0258	RW		EMS page 0 register.
04F4	RW		First 1M mapping register.
08F4	RW		Bank 0 configuration register.
09F4	RW		Cache system status register.
0AF4	R		Cache TAG test register.
0BF4	RW		Cacheable memory control register.
0CF4	R		Cache identification register.
14F4	RW		First 1M configuration register.
18F4	RW		Bank 1 configuration register.
24F4	R		Parity error address register.
28F4	RW		Bank 2 configuration register.
34F4	RW		32-bit speed select register.
38F4	RW		Bank 3 configuration register.
4258	RW		EMS page 1 register. (Not used)
44F4	RW		16-bit speed select register.
48F4	RW		Bank 4 configuration register.
54F4	RW		8-bit speed select register.
58F4	RW		Bank 5 configuration register.
64F4	RW		Refresh/INTA speed select register.
68F4	RW		Bank 6 configuration register.
74F4	R		Bus/memory controller revision register.
78F4	RW		Bank 7 configuration register.
8258	RW		EMS page 2 register. (Not used)
C258	RW		EMS page 3 register. (Not used)

8. Power Requirements

The power requirement is yet to be determined, but should be similar to the existing Z-386 system and will in fact be powered by the same power supply.

Appendix A. PAL Equations

The following PAL equations are written in the *CUPL* device programming language. For information on it's syntax see appendix C.

Appendix A.1 Coprocessor Interface, 644-65

Appendix B. PAL Test Vectors

The following PAL test vectors were written in the *CUPL* device programming language. For information on it's syntax see appendix C.

Appendix B.1 Coprocessor Interface, 644-65

Appendix C. CUPL Language Reference

For a complete definition of the CUPL language see the CUPL user's manual. The following is a brief definition of the major operators.

Priority	CUPL syntax	Definition	Example
1	!	NOT	!A
2	&	AND	A & B
3	#	OR	A # B
4	\$	XOR	A \$ B

NOT		AND			OR			XOR		
A	!A	A	B	A&B	A	B	A#B	A	B	ASB
0	1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1	0	1	1
		1	0	0	1	0	1	1	0	1
		1	1	1	1	1	0	1	1	0

Signal Name Extensions

Extension	Description
.D	D Input of D-type flip-flop
.J	J Input of JK-type flip-flop
.K	K Input of JK-type flip-flop
.S	S Input of SR-type flip-flop
.R	R Input of SR-type flip-flop
.DQ	Q Output of D-type flip-flop
.AP	Asynchronous Preset of flip-flop
.AR	Asynchronous Reset of flip-flop
.SP	Asynchronous Preset of flip-flop
.SR	Asynchronous Reset of flip-flop
.CK	Programmable Clock of flip-flop
.OE	Programmable Output Enable
.CA	Complement Array

Appendix D. Reference Material

Advanced Work Station Technical Reference (595-xxxx)

This manual contains information on computer operation and programming methods, as well as a hardware introduction for the service technician.

Publisher: Zenith Data Systems

MS-DOS Programmer's Utility Pack (CB-3163-30)

This software packing produced by Microsoft and ZDS, contains information and software for writing and assembling MS-DOS and native code 8086, 80286, and 80386 assembly language programs.

Publisher: Zenith Data Systems

644-25 Posted Write Array

Internal ZDS document describing the hardware specification of the write queue gate arrays.

Publisher: Zenith Data Systems

644-28 High Performance 32/16-Bit Cache Memory Controller

Internal ZDS document describing the hardware specification of the system/cache controller gate array.

Publisher: Zenith Data Systems

644-29 32-Bit AT Bus/Memory Controller

Internal ZDS document describing the hardware specification of the bus/memory controller gate array.

Publisher: Zenith Data Systems

WTL3167 Floating Point Coprocessor

Manufacturer's data sheet which provides detailed information on hardware interfacing, electrical data, and programming.

Publisher: Weitek Corporation

Introduction to the 80386

This manual provides an overview and introduction to the 80386 microprocessor.

Publisher: Intel Corporation order number 231252

Microprocessor and Peripheral Handbook, Volumes 1 & 2

These manuals provide information on the architecture and programming procedures for the 80386 and 80387, and other support circuitry.

Publisher: Intel Corporation order number 230843-005.

80386 Programmer's Reference Manual

This manual contains detailed information on the architecture, instruction set, and programming procedures required for the 80386 microprocessor.

Publisher: Intel Corporation order number 230985-001

80386 Hardware Reference Manual

This manual contains detailed descriptions of the architecture and operation of the 80386 microprocessor. Device timing diagrams and pinouts are also included in this manual.

Publisher: Intel Corporation order number 231732-002.

80387 Programmer's Reference Manual

This manual contains detailed descriptions of the architecture, instruction set, and programming procedures required for the 80387 numeric coprocessor.

Publisher: Intel Corporation order number 231917-001.

LIM Expanded Memory Device Interface Specification

This specification details the software and hardware interface for EMS memory systems. Intel Technical Support provides a toll-free number 1-800-538-3373, that you may call to obtain this specification.

Publisher: Intel Corporation

Intel publications may be obtained by writing to the following address:

*Intel Literature Sales
P. O. Box 58130
Santa Clara, California 95052-8130*

or dial toll free: 1-800-548-4725

Weitek publications may be obtained by writing to the following address:

*WEITEK Corporation
1060 East Arques Avenue
Sunnyvale, California 94086-BRM-9759
ATTN: Ed Masuda*