Advanced
Micro
Devices

AmZ8001/ AmZ8002 Processor Interface

Advanced Micro Devices

AmZ8000 Family Reference Manual

Principles of Operation AmZ8001/AmZ8002 Processor Interface

AM-PUB089

CHAPTER 1
INTERFACING FUNDAMENTALS

Introduction

The AmZ8001 and AmZ8002 are initial members of the AmZ8000 microprocessor family. This chapter discusses the CPU interface signals and suggested circuit implementation for clock generation, CPU initialization (reset), wait state generation, signal buffering and single stepping.

Interface Signal Description

Figure 1.1 shows the CPU logic symbols and the following is a description of the signals.

Vcc: +5V Power Supply

Vss: Ground

AD∅-AD15: Address/Data Bus (Bidirectional, 3-state)

This 16 bit multiplexed address/data bus is used for all I/O and memory transactions. HIGH on the bus corresponds to 1 and LOW corresponds to ∅. AD∅ is the least significant bit position and so on and AD15 is the most significant. The $\overline{\text{AS}}$ output and $\overline{\text{DS}}$ output will indicate whether the bus is used for address or data. The status output lines ST∅-ST3 will indicate whether address information on the bus is intended for memory or I/O.

$\overline{\text{AS}}$: Address Strobe (output, 3-state)

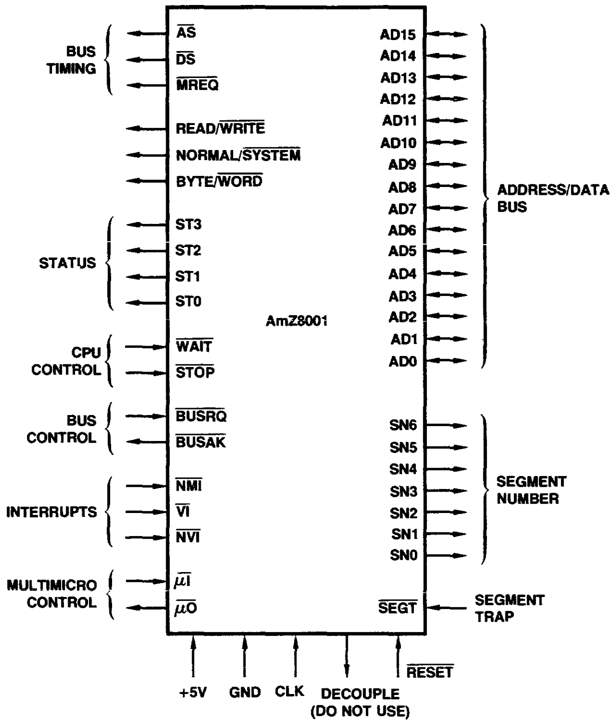
LOW on this output indicates that the AD∅-AD15 bus contains address information. The address information is stable by the time of the LOW to HIGH transition of the $\overline{\text{AS}}$ output. The status outputs ST∅-ST3 will indicate whether the bus contains a memory address or I/O address.

$\overline{\text{DS}}$: Data Strobe (output, 3-state)

LOW on this output indicates that the AD∅-AD15 bus is being used for data transfer. The R/$\overline{\text{W}}$ output indicates the direction of data transfer - read (or in) means data into the CPU and write (or out) means data from the CPU. During a read operation, data can be gated on to the bus when $\overline{\text{DS}}$ goes LOW. A LOW to HIGH transition on the $\overline{\text{DS}}$ output indicates that the CPU has accepted the data. During a write operation, LOW on the $\overline{\text{DS}}$ output indicates that data is setup on the bus. Data will be removed sometime after the LOW to HIGH transition of the $\overline{\text{DS}}$ output.

R/$\overline{\text{W}}$: Read/Write (output, 3-state)

This output indicates the direction of data flow on the AD∅-AD15 bus. HIGH indicates a read operation, i.e. data into the CPU and LOW indicates write operation, i.e. data from the CPU. This output is activated at the same time as $\overline{\text{AS}}$ going LOW and remains stable for the duration of the whole transaction.



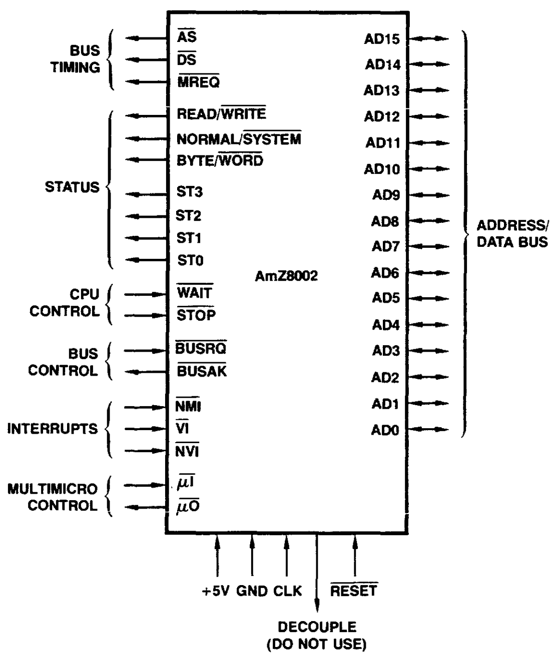


Figure 1-1. CPU Logic Symbols.

ST∅-ST3: Status (outputs, 3-state)

These four outputs contain information regarding the current transaction in a coded form. The status line codes are shown below:

ST3	ST2	ST1	ST∅	
L	L	L	L	Internal Operation
L	L	L	H	Memory Refresh
L	L	H	L	Normal I/O Transaction
L	L	H	H	Special I/O Transaction
L	H	L	L	Segment Trap Acknowledge in AmZ8001 Reserved in AmZ8002
L	H	L	H	Non-maskable Interrupt Acknowledge
L	H	H	L	Non-vectored Interrupt Acknowledge
L	H	H	H	Vectored Interrupt Acknowledge
H	L	L	L	Memory Transaction for Operand
H	L	L	H	Memory Transaction for stack
H	L	H	L	Reserved
H	L	H	H	Reserved
H	H	L	L	Memory Transaction for Instruction fetch (Subsequent word)
H	H	L	H	Memory Transaction for Instruction fetch (First word)
H	H	H	L	Reserved
H	H	H	H	Reserved

$B/\overline{W}$: Byte/word (output, 3-state)

This output indicates the type of data transferred on the ADØ-AD15 bus. HIGH indicates byte (8-bit) and LOW indicates word (16-bit) transfer. This output is activated at the same time as $\overline{AS}$ going LOW and remains valid for the duration of the whole transaction. The address generated by the CPU is always a byte address. However, the memory is organized as 16-bit words. All instructions and word operands are word aligned and are addressed by even addresses. Thus, for all word transactions with the memory the least significant address bit will be zero. When addressing the memory for byte transactions, the least significant address bit determines which byte of the memory word is needed; even address specifies the most significant byte and odd address specifies the least significant byte. In the case of I/O transactions, the address information on the ADØ-AD15 bus refers to an I/O port and $B/\overline{W}$ determines whether a data word or data byte will be transacted. During I/O byte transactions, the least significant address bit A0 determines which half of the ADØ-AD15 bus will be used for the I/O transactions. The STØ-ST3 outputs will indicate whether the current transaction is for memory, normal I/O or special I/O.

$\overline{VI}$: Vectored Interrupt (Input)

LOW on this input constitutes vectored interrupt request. Vectored interrupt is next lower to the non-maskable interrupt in priority. The VIE bit in the Flag and Control Word register must be 1 for the vectored interrupt to be honored. The CPU will respond with Vectored Interrupt Acknowledge code on the STØ-ST3 outputs and will begin the interrupt sequence. The $\overline{VI}$ input can be driven LOW any time and is customarily held LOW until acknowledged.

$\overline{NVI}$: Non-Vectored Interrupt (Input)

LOW on this input constitutes non-vectored interrupt request. Non-vectored has the lowest priority of the three types of interrupts. The NVIE bit in the Flag and Control Word register must be 1 for this request to be honored. The CPU will respond with Non-Vectored Interrupt Acknowledge code on the STØ-ST3 outputs and will begin the interrupt sequence. The $\overline{NVI}$ input can be driven LOW anytime and is customarily held LOW until acknowledged.

$\overline{\mu I}$: Micro-In (Input)

This input participates in the resource request daisy chain. See the section on multi-microprocessor support facilities in this document.

$\overline{\mu O}$: Micro-Out (Output)

This output participates in the resource request daisy chain. See the section on multi-microprocessor support facilities in this document.

$\overline{\text{RESET}}$: Reset (Input)

LOW on this input initiates a reset sequence in the CPU. See the section on Initialization for details on reset sequence.

$\overline{\text{BUSRQ}}$: Bus Request (Input)

LOW on this input indicates to the CPU that another device (such as DMA) is requesting to take control of the bus. The $\overline{\text{BUSRQ}}$ input can be driven LOW anytime. The CPU synchronizes this input internally. The CPU responds by activating $\overline{\text{BUSAK}}$ output LOW to indicate that bus has been relinquished. Relinquishing the bus means that the ADØ-AD15, $\overline{\text{AS}}$, $\overline{\text{DS}}$, B/$\overline{\text{W}}$, R/$\overline{\text{W}}$, N/$\overline{\text{S}}$, STO-ST3 and $\overline{\text{MREQ}}$ outputs will go to high impedance state. The requesting device should control these lines in an identical fashion to the CPU to accomplish transactions. The $\overline{\text{BUSRQ}}$ input must remain LOW as long as needed to perform all the transactions and the CPU will keep the $\overline{\text{BUSAK}}$ output LOW. After completing the transactions, the device must disable its ADØ-AD15, $\overline{\text{AS}}$, $\overline{\text{DS}}$, B/$\overline{\text{W}}$, R/$\overline{\text{W}}$, N/$\overline{\text{S}}$, STØ-ST3 and $\overline{\text{MREQ}}$ outputs into high impedance state and stop driving the $\overline{\text{BUSRQ}}$ input LOW. The CPU will make $\overline{\text{BUSAK}}$ output HIGH sometime later and take the bus control back.

$\overline{\text{BUSAK}}$: Bus Acknowledge (Output)

LOW on this output indicates that the CPU has relinquished the bus in response to a bus request.

$\overline{\mathrm{NMI}}$: Non-Maskable Interrupt (Input)

HIGH to LOW transition on this input constitutes non-maskable interrupt request. The CPU will respond with the non-maskable Interrupt Acknowledge on the ST0-ST3 outputs and will enter an interrupt sequence. The transition on the $\overline{\mathrm{NMI}}$ can occur anytime. Of the three kinds of interrupts available, the non-maskable interrupt has the highest priority.

$\overline{\mathrm{WAIT}}$: Wait (Input)

LOW on this input indicates to the CPU that memory or I/O is not ready for the data transfer and hence the current transaction should be stretched. The $\overline{\mathrm{WAIT}}$ input is sampled by the CPU at certain instances during the transaction. If $\overline{\mathrm{WAIT}}$ input is LOW at these instances, the CPU will go into wait state to prolong the transaction. The wait state will repeat until the $\overline{\mathrm{WAIT}}$ input is HIGH at the sampling instant.

$\mathrm{N}/\overline{\mathrm{S}}$: Normal/System Mode (Output, 3-state)

HIGH on this output indicates that the CPU is operating in Normal Mode and LOW indicates operation in System Mode. This output is derived from the Flag Control Word (FCW) register. The FCW register is described under the program status information section of this document.

$\overline{\mathrm{MREQ}}$:

LOW on this output indicates that a CPU transaction with memory is taking place.

CLK: Clock (Input)

All CPU operations are controlled from the signal fed into this input.

DECOUPLE: Output from the on-chip substrate bias generator. Presently not connected.

$\overline{\mathrm{STOP}}$: Stop (Input)

This active LOW input facilitates one instruction at a time operation. See the section on single stepping.

The following signals exist in AmZ8001 only.

SNØ-SN6: Segment number (Outputs, 3-state)

These seven outputs contain the segment number part of a segmented memory address. SN6 is the most significant and SNØ the least significant bit. HIGH corresponds to 1 and LOW corresponds to Ø.

$\overline{\text{SEGT}}$: Segment Trap (Input)

LOW on this input constitutes a segmentation trap. This line is asserted by the memory management unit when an access violation has occured. The CPU will respond with the segment trap acknowledge code on the status line, and commence the trap sequence. The $\overline{\text{SEGT}}$ input can be driven low at any time and is customarily held LOW until acknowledged.

CLOCK GENERATION

The CPU requires a single phase clock for its operation. Figure 1.2 shows a suggested circuit. The oscillator consists of an inverter biased into the linear region by the 390 ohm resistor. The frequency of oscillation is fixed by the 8MHz crystal. The oscillator output is divided by two in the toggling flip-flop to generate a square wave at 4MHz. The flip-flop output is buffered by a pair of complementary transistors as shown to obtain the CPU clock signal. The buffering circuit shown ensures that clock signal amplitude satisfies the required specifications of the CPU device. In some applications, buffering the flip-flop output with a suitable 3-state buffer may be satisfactory.

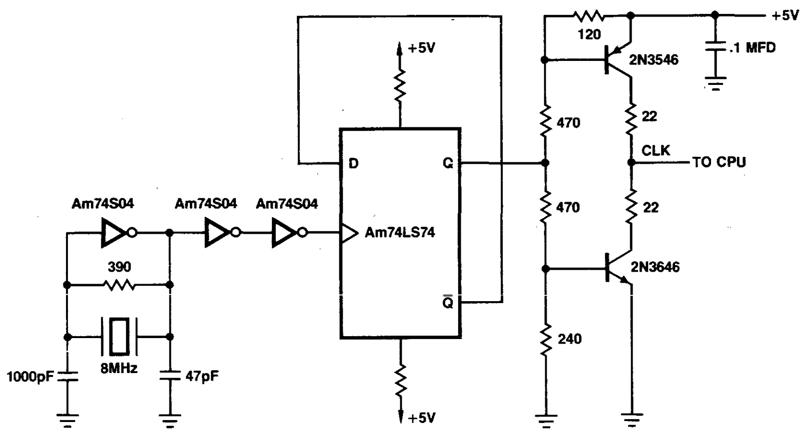


Figure 1-2. CPU Clock Generation.

CPU Initialization

The CPU will be initialized when its $\overline{RESET}$ input is LOW for a minimum of 5 clock periods. Figure 1.3 is a suggested initialization circuit.

SW1 is a single-pole-double-throw momentary contact switch debounced by the flip-flop formed by the cross coupled NAND gates. Depressing and releasing the switch will generate a debounced HIGH pulse at the output of the flip-flop. This output is connected to the LOAD input of the 74LS163 synchronous binary counter. When the LOAD input is HIGH the counter begins to count at the CPU clock rate since the CP input of the counter is driven by the CPU clock. The count starts from the initially loaded value of 15 and will go through 0 up to 8. At Count 8, the ENP input of the counter will be LOW because of the decoding by the two NAND gates monitoring the QA and QD outputs. The LOW level on the ENP input disables counting and the counter holds the value 8. When SW1 is released, the LOAD input of the counter becomes LOW. This results in re-loading the initial value of 15 from the parallel data inputs. The QD output of the counter is the RESET signal for the CPU.

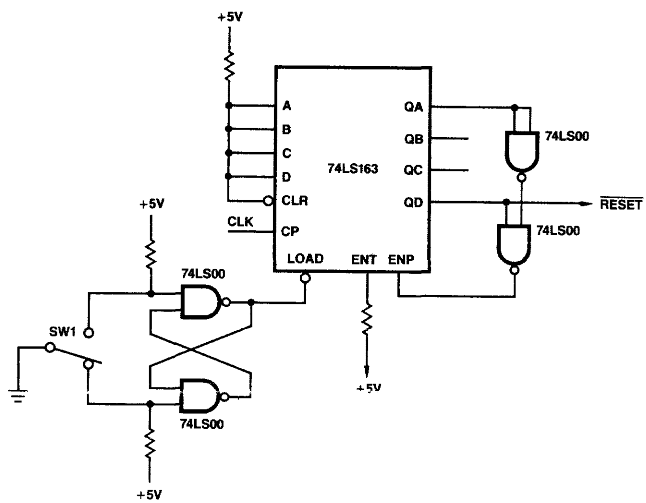


Figure 1-3. CPU Initialization Circuit.

CPU SIGNAL BUFFERING

In general, signal buffering is required for two reasons: Capacitive load and fan out. Driving capacitive loads directly from a CPU output will degrade the signal waveform due to the high impedance nature of the MOS outputs. Buffering will isolate the load capacitance from the CPU output. The CPU outputs can sink 2mA current: corresponding to a fan out of 4 low-power Schottky loads. Higher fan out will require buffering. The CPU signals fall into two categories - bidirectional and unidirectional. The AmZ8104 is an octal buffer intended for buffering bidirectional signals while AmZ8144 is another octal buffer intended for unidirectional signals.

Figure 1.4 is a bidirectional buffering scheme using the AmZ8104s. When the CD input is HIGH, the chip is disabled and both AØ-A7 and BØ-B7 signals of the AmZ8104 will be in the high impedance state. When $T/\overline{R}$ input is HIGH AØ-A7 signals of the AmZ8104 receive data and transmit it to the corresponding BØ-B7 output. Thus in transmit mode information from AØ will appear on the BØ output and so on. On the other hand, if the $T/\overline{R}$ input is LOW, BØ-B7 signals of the AmZ8104 will transfer information to the corresponding AØ-A7 output. In Figure 1.4 the $T/\overline{R}$ input is derived from the $R/\overline{W}$ and $\overline{DS}$ outputs of the CPU. The ADØ-AD15 outputs of the CPU are connected to the A-side of the AmZ8104 while the B-side is the buffered bus.

When address information is present on the ADØ-AD15 bus, the $\overline{DS}$ output from the CPU is HIGH. Thus the $T/\overline{R}$ input of the AmZ8104 is HIGH. Hence address information from the ADØ-AD15 will appear on the buffered bus. If the CPU is performing a write operation, the $R/\overline{W}$ output from the CPU will be LOW. After removing the address information on the ADØ-AD15 outputs, the CPU will establish data on these outputs and activate the $\overline{DS}$ output LOW. Because the $R/\overline{W}$ is still LOW, the AmZ8104 will transmit data from the CPU side to the buffered side. On the other hand, if the CPU is performing a read operation, the R/$\overline{W}$ will be HIGH. After removing the address information from the ADØ-AD15, the CPU will activate the $\overline{DS}$ output LOW. The resulting LOW on the T/$\overline{R}$ input of the AmZ8104 will transfer information from buffered side to the CPU side.

It should be noted that the CD input of the AmZ8014 is driven by the inverted $\overline{BUSAK}$ output from the CPU. When CPU has relinquished the bus to a DMA device the $\overline{BUSAK}$ will be LOW, thus disabling the AmZ8104. One advantage of the scheme in Figure 1.4 should be pointed out. During write operation, $\overline{DS}$ output going HIGH signifies impending termination of the write cycle. The data is held stable on the ADØ-AD15 outputs for a fixed time after $\overline{DS}$ going HIGH. This provides data hold time when $\overline{DS}$ is used to generate write enable signal for the memory devices. Using the scheme shown in Figure 1.5 transfers the data hold time benefit to the buffered side also.

Figure 1.5 shows unidirectional buffering using AmZ8144. Normally $\overline{BUSAK}$ is HIGH making $\overline{1G}$ and $\overline{2G}$ inputs of the AmZ8144 LOW. This enables the chip and inputs 1A1, 1A2 etc will be transferred to the corresponding output 1Y1, 1Y2 etc. During DMA operations, $\overline{BUSAK}$ will be LOW and will disable the AmZ8144 into high impedance state.

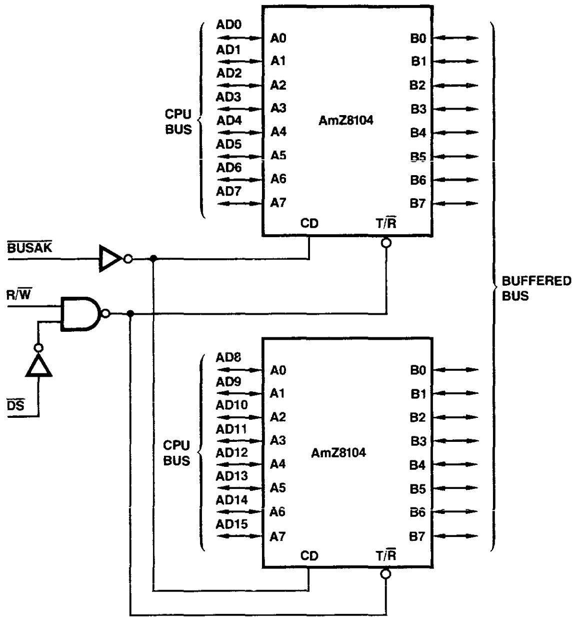


Figure 1-4. Bidirectional CPU Buffering.

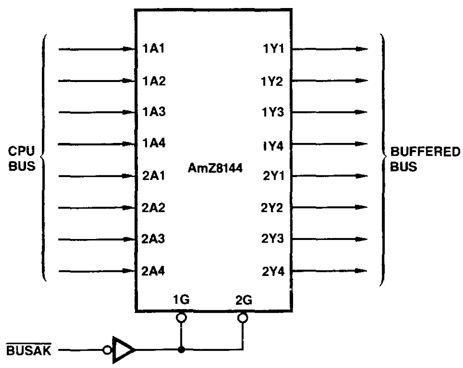


Figure 1-5. CPU Unidirectional Buffering.

ADDRESS LATCHING

The ADØ-AD15 bus from the CPU is time multiplexed for address/data and is bidirectional in nature. The address information on this bus is valid only during the T1 state of a machine cycle. In many applications, the address must be latched externally so that it will remain stable for the whole transaction. The AmZ8173 octal latches are intended for this purpose. Figure 1.6 is a suggested circuit for address latching. Normally, $\overline{\text{BUSAK}}$ is HIGH; thus the $\overline{\text{OE}}$ input of the AmZ8173 is LOW enabling the internal 3-state buffers. The G input of the AmZ8173 is driven by the $\overline{\text{AS}}$ through an inverter. When $\overline{\text{AS}}$ is LOW, the latches are enabled, hence the latch outputs YØ, Y1 etc. will follow the corresponding inputs DØ, D1 etc. Thus, the address provided by the CPU will appear at the latch outputs. After the address has become stable, the $\overline{\text{AS}}$ goes HIGH thus disabling the latches. The address that was present prior to $\overline{\text{AS}}$ going HIGH is stored in the latches. The latch outputs will be disabled into the high impedance state by LOW on the $\overline{\text{BUSAK}}$. If such disabling is not required, the $\overline{\text{OE}}$ input of the AmZ8173 should be grounded.

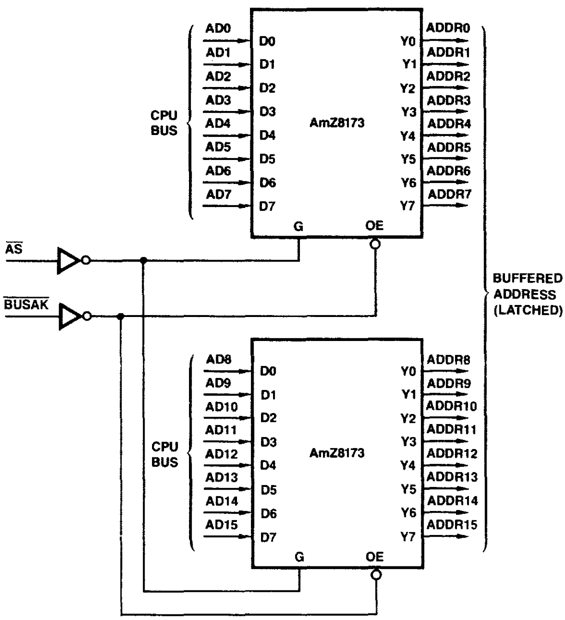


Figure 1-6. CPU Address Latching.

Single Stepping

The $\overline{STOP}$ input is used to accomplish single instruction stepping. The CPU samples the $\overline{STOP}$ input during the last machine cycle of an instruction execution. If the $\overline{STOP}$ input is LOW, the CPU completes fetching the next instruction. Instead of executing this fetched instruction, a series of memory refresh cycles are performed. The $\overline{STOP}$ input is repeatedly sampled by the CPU during these refresh cycles. If the $\overline{STOP}$ is found HIGH, one more final refresh cycle is performed and the CPU resumes execution of the instruction. Thus, by selectively activating and deactivating the $\overline{STOP}$ input, single instruction stepping can be accomplished.

Figure 1.7 shows a suggested single step circuit. It uses two switches; SW1 is a single pole single throw switch and SW2 is a single pole double throw momentary contact switch. SW1 in the RUN position, allows the CPU to operate normally. In the HALT position, it causes the CPU to stop and execute repetitive refresh cycles. SW1 must be set to halt position for single stepping. One instruction will be executed for each activation of SW2.

With SW1 in the RUN position, the D-input of the flip-flop D2 is LOW. Thus, the LOW to HIGH transition of $\overline{AS}$ repeatedly clears the flip-flop. Thus its $\overline{Q}$ output will be HIGH making the $\overline{STOP}$ input of the CPU HIGH. When SW1 is moved to HALT position, $\overline{Q}$ output of D2 goes LOW on the next $\overline{AS}$ transition.

When SW2 is activated, the clock input of D1 flip-flop is connected the $\overline{AS}$. Thus the Q output of D1 goes HIGH on the LOW to HIGH transition of $\overline{AS}$. On the following $\overline{AS}$ transition, $\overline{Q}$ of D2 goes HIGH, thus deactivating the $\overline{STOP}$. Once D1 flip-flop is set, it remains set. The subsequent $\overline{AS}$ transition will set D2 again, establishing LOW again on the $\overline{STOP}$. Thus $\overline{STOP}$ was made HIGH for one machine cycle following activation of SW2. This allows the CPU to execute one instruction.

WAIT STATE GENERATION

Any I/O device or memory interfaced to the CPU must activate the $\overline{\text{WAIT}}$ input LOW to request stretching of a machine cycle. Such stretching is needed if a device requires more time to complete a CPU transaction than is normally allowed in the CPU timing.

A slow memory, when accessed by the CPU must request insertion of one or more wait states. The actual number of wait states required is known beforehand and will not vary from transaction to transaction. Such a situation is called fixed wait requirement. An I/O device may also require extra time to complete a CPU transaction. In the case of I/O, the number of wait states to be inserted depends upon when the CPU attempts an access to the I/O device in relation to the latter's operating cycle. The device may be busy internally and cannot respond to the CPU access until it completes the internal operation. This is an illustration of what is called demand wait requirements.

Figure 1.8 shows a suggested circuit for fixed wait operations. It uses a 74LS195A, a 4-bit parallel Load Shift Shift register. The shift register is clocked by the CPU clock connected to its CP input. The data inputs A, B and C are connected to jumpers as shown and Input D is grounded. Normally the $\overline{\text{MREQ}}$ output from the CPU is HIGH and goes LOW during memory transaction cycles. When $\overline{\text{MREQ}}$ is HIGH, the S/$\overline{\text{L}}$ input of the shift register is LOW. Thus, information present on the data inputs of the shift register will appear at its output. Hence the $\overline{\text{QD}}$ output will be HIGH. This output is connected to the $\overline{\text{WAIT}}$ input of the CPU. When the CPU is going to perform a memory transaction, the $\overline{\text{MREQ}}$ will be LOW and hence the S/$\overline{\text{L}}$ input of the shift register will be HIGH i.e. shift mode. The next LOW to HIGH transition of the CP input will then shift the register one place to the right. If the jumper at the C input is not present, the QC output would have been HIGH prior to this shifting. Hence the $\overline{\text{QD}}$ will become LOW after the shift driving the $\overline{\text{WAIT}}$ input of the CPU LOW. The CPU recognizes this and inserts a wait state. The next CP transition will shift the register again as before. If there is no jumper at the B input, the $\overline{\text{QD}}$ will be still LOW and the CPU will insert a second wait state. Similarly, if there is no jumper at the A input also, the CPU will insert a total of 3 wait states. As the register is

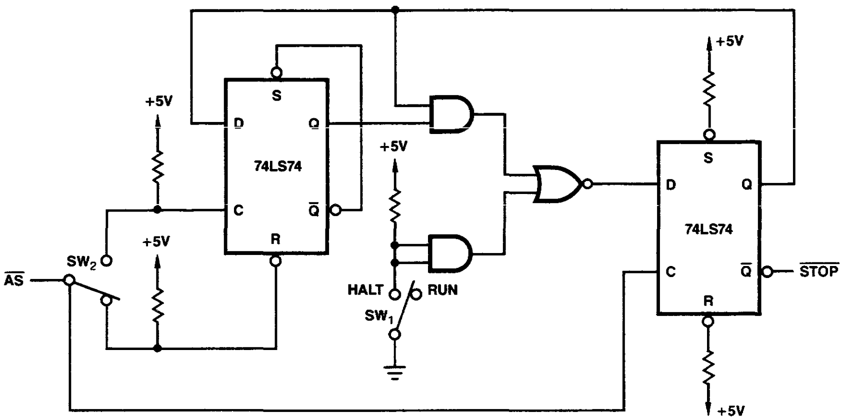


Figure 1-7. CPU Single Step Circuit.

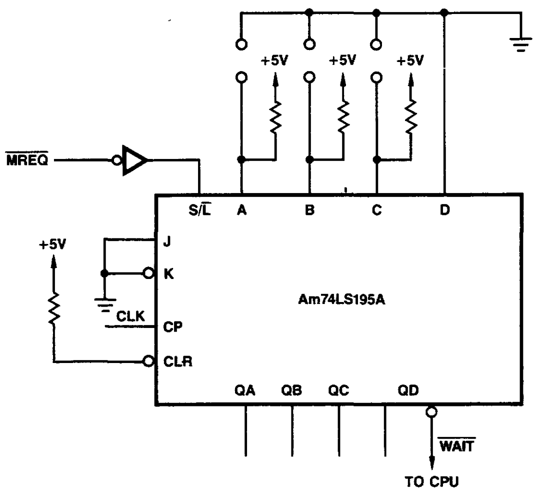


Figure 1-8. Fixed Wait Implementation.

shifting right, a 0 is being shifted into the register from J-K input of the register, thus on the 4th CP transition $\overline{QD}$ will go HIGH signalling the CPU to terminate wait state insertions and proceed with normal operations. In summary, the circuit shown in Figure 1.7 can be used for a programmable fixed wait generation. In Figure 1.7 $\overline{MREQ}$ output from the CPU is used to trigger the $\overline{WAIT}$ input. This implies that wait states will be introduced for every memory access irrespective of the memory address. It is possible to use an appropriately decoded value of the address to trigger the wait state generator when necessary rather than the $\overline{MREQ}$ signal.

Demand Wait Implementation

The fixed wait implementation described above is suitable wherever a fixed number of wait states are required. For example, an access to an EPROM memory may require the unconditional insertion of two wait states in every access. In some circumstances, however, the responding device may assert its own wait requirement to the CPU. If the device has no knowledge of the CPU clock, the $\overline{WAIT}$ request may be asynchronous. Thus some form of synchronization - leading to further delay in the wait path may be required. The time required for a responding device to assert its wait request to the CPU, may be too long for the request to be honored by the CPU since the latter samples the $\overline{WAIT}$ input at a certain point in the transaction. Also this time delay may be aggravated by the synchronization requirements making the device wait request too late in the CPU's machine cycle. Figure 1.9 shows an implementation of demand wait which overcomes these problems using one TTL package. As in the fixed wait generation, a 74LS195 Shift Register is used. The CPU $\overline{AS}$ signal, and the inverted CPU Clock ($\overline{CLK}$)cause a programmed number of wait states to be implemented. In this example, however, a LOW on the QD output of the register signifies a WAIT request. The J and K inputs of the shift register are both driven HIGH to cause the shifting of HIGHs into the register. Thus a fixed number of wait states are programmable on register inputs A-D. The $\overline{PAUSE}$ or $\overline{WAIT}$ output from the responding device drives the CLR input of the register. Thus a LOW (= WAIT Required) on $\overline{PAUSE}$

causes the QD register output ($\overline{\text{WAIT}}$) to go LOW which in turn generates CPU wait states. When the responding device drives its $\overline{\text{PAUSE}}$ output HIGH, the register returns to the shift mode and subsequently shifts a HIGH into QD, releasing the CPU from the wait states. The shift register inputs A-D should be programmed to hold the CPU in the WAIT STATES until the responding device is able to assert its $\overline{\text{PAUSE}}$ output. In this manner, most devices with a $\overline{\text{PAUSE}}$ output will be given sufficient time to make a wait/no wait decision, since the CPU will always be delayed by the fixed wait states.

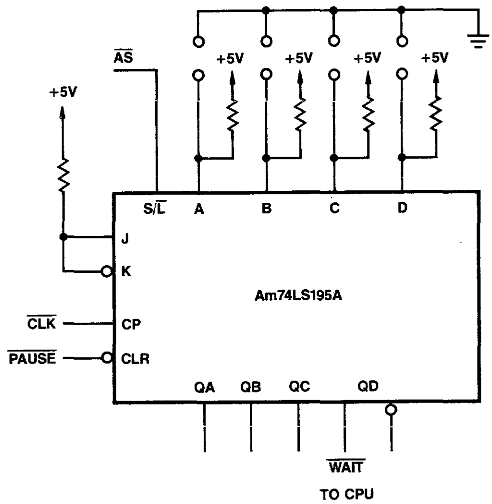


Figure 1-9. Demand Wait Implementation.

CHAPTER 2
MEMORY INTERFACING TECHNIQUES

Introduction

This chapter describes interfacing memory to the CPU. The CPU's have certain memory requirements, in terms of both timing and data format. These requirements are discussed in detail and this is followed by some examples of memory connection. Three types of memory are shown connected to the CPU. The first example shows the connection of a 16K byte memory employing Am9124 1K X 4 static RAMs. The second design interfaces the Am9016 16K X 1 dynamic RAMs providing a 64K byte memory. The final example describes the connection of the Am9716 16K X 1 EPROM which can be used to implement start-up facilities.

Memory Addressing

The AmZ8001 and AmZ8002 CPU's have different memory addressing capabilities. The AmZ8002 address memory with a 16-bit address which is valid on ADØ-AD15 during the first half of a memory cycle. The address designates a byte in memory and thus up to 64Kb of memory can be directly accessed. The AmZ8001, on the other hand, addresses memory with a 23-bit segmented address. Seven bits of this address designate a segment number and are valid on CPU outputs SNØ-SN6 during the memory transaction. The remaining 16 bits of address specify a byte offset within the segment. Thus each segment may be up to 64Kb in size, and up to 128 segments can be specified. Thus the AmZ8001 has an addressing range of 8Mb. The addressing range can be extended by incorporating the CPU STØ-ST3 lines in the memory address decode. As described in Chapter 1, the STØ-ST3 lines indicate the type of CPU transactions. Three memory spaces are defined; code, data and stack. If the CPU $N/\overline{S}$ line is included in this decode, six memory spaces are defined; code (system), data (system), stack (system), code (normal), data (normal) and stack (normal). Thus up to a six-fold increase in addressing range can be gained, by separating the memory spaces. In practice, there are several advantages to a partial decode of the STØ-ST3 lines in which data and stack memory are not separated. Since stacks are addressed using general purpose registers, addressing modes such as "Top of Stack + n" are available. The STØ-ST3 lines indicate "data" in this case, but since the data and stack spaces are common, this is insignificant.

The CPU address designates a byte location. The majority of CPU memory accesses, however, can be 16-bits wide and must be aligned on even byte boundaries. Instruction fetches from memory are all 16-bits wide, and operand accesses may be 8 or 16 bits wide. Thus the memory system should be 16 bits wide with a byte access capability. A conceptual memory system is shown in Figure 2.1. The memory consists of two byte banks; one bank contains all the even addressed bytes in memory and the other bank contains all the odd addressed bytes. When accessing a word, the memory address should always be even. See AMPUB086 - "The AmZ8001/2 Processor Instruction Set" for details of memory addressing.

The two byte banks of memory have separate enables. (The nature of the enable may vary dependent upon the type of memory device used). Each enable is driven from a logical "OR" of the CPU $R/\overline{W}$ line, the $B/\overline{W}$ line, inverted, and the least significant memory address bit AØ. The even bank enable is driven from the inverse of AØ. During a word transaction, the CPU $B/\overline{W}$ line is LOW. Thus both banks will be enabled. For a memory read, the CPU inputs 16-bits of data from the even and odd banks which are connected to the upper and lower halves of the bus respectively. For a memory write, the CPU outputs data onto both halves of the bus. Both banks of memory are enabled and the 16-bit data is written into the memory.

If a byte transaction is being executed, the $B/\overline{W}$ line is HIGH. For a memory read, the $R/\overline{W}$ line will be HIGH. Thus both banks of memory will be read. The required byte may be on the upper or lower half of the bus. The CPU steers either the even or odd byte to the byte destination, dependent upon the least significant address bit AØ. If AØ is LOW, the upper (even) half of the bus will be read. If AØ is HIGH, the lower (odd) half of the bus will be read. Thus, during a byte read the memory need only respond with a 16-bit word containing the byte data. During a byte write, $R/\overline{W}$ will be LOW. Thus either the odd or even byte bank of memory will be enabled, dependent upon the value of the least significant address bit AØ. If AØ is LOW, the even byte bank will be enabled. If AØ is HIGH, the odd byte bank will be enabled. When writing a byte to memory, the CPU duplicates the byte data on both halves of the 16-bit data bus. Hence the memory can pick off the byte from either half of the bus, by enabling only the relevent (even or odd) bank.

Memory transaction timing

The transactions between CPU and memory are referenced to the $\overline{AS}$ and $\overline{DS}$ CPU outputs. Figure 2.2 shows the memory transaction timing. The memory transaction commences with $\overline{AS}$ going LOW. The CPU status information (STØ-ST3, B/$\overline{W}$, R/$\overline{W}$) becomes valid at least 40ns before the trailing edge of $\overline{AS}$, and indicate one of the several possible types of memory transaction discussed in the previous section, together with the size and direction of the transaction. The 16-bit address becomes valid on ADØ-AD15 at least 55ns before the trailing edge of $\overline{AS}$. Whereas the STØ-ST3 outputs are valid for the whole transaction, the address is not valid 60ns after the trailing edge of $\overline{AS}$. This is due to the address bus being shared with the data bus. In many applications, the address will be required for the whole transaction. This can be implemented using external latches. Thus the address will latch on the trailing edge of $\overline{AS}$, and remain valid until the next transaction. The SNØ-SN6 segment number output, in the AmZ8001, becomes valid in the Clock cycle preceeding the start of the memory transaction. This is to facilitate the connection of a memory management unit to the AmZ8001. In a memory read transaction, the CPU reverses the direction of its ADØ-AD15 lines, in preparation for receiving the incoming data. This reversal is indicated by $\overline{DS}$ going LOW. Thus the memory can use $\overline{DS}$ LOW to enable its output buffers to drive the data onto the CPU ADØ-AD15 bus. The data is required by the CPU no later than 155ns after $\overline{DS}$ goes LOW.

In terms of $\overline{AS}$, the memory access time is 290ns while in terms of $\overline{MREQ}$, this figure is 330ns. Either $\overline{AS}$ or $\overline{MREQ}$ can be used to initiate the memory read cycle. In the dynamic memory example discussed below, $\overline{MREQ}$ is used to generate the row address strobe thus defining the access time at 330ns.

In the memory write case, the address/data bus is not reversed in direction but the valid memory address is replaced by valid data. $\overline{DS}$ does not go LOW for a minimum of 55ns following valid data, to allow data to be set up at the memory inputs. $\overline{DS}$ remains LOW for a minimum of 160ns. Data is guaranteed valid 80ns after the rising edge of $\overline{DS}$, to enable hold times to be met where required. If the memory system cannot meet the response times required by the CPU, Wait States can be inserted in the CPU's memory transaction. See Chapter 1 for wait state generation circuit examples.

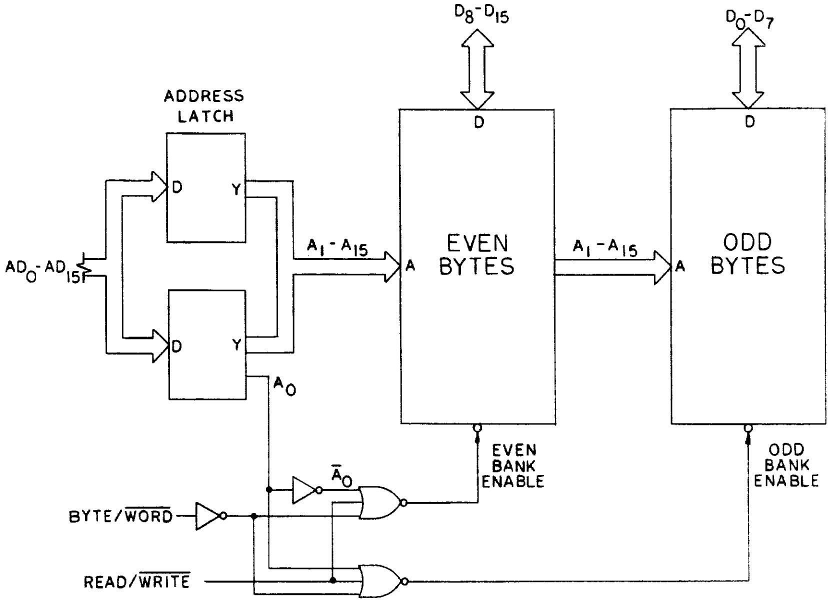


Figure 2-1.

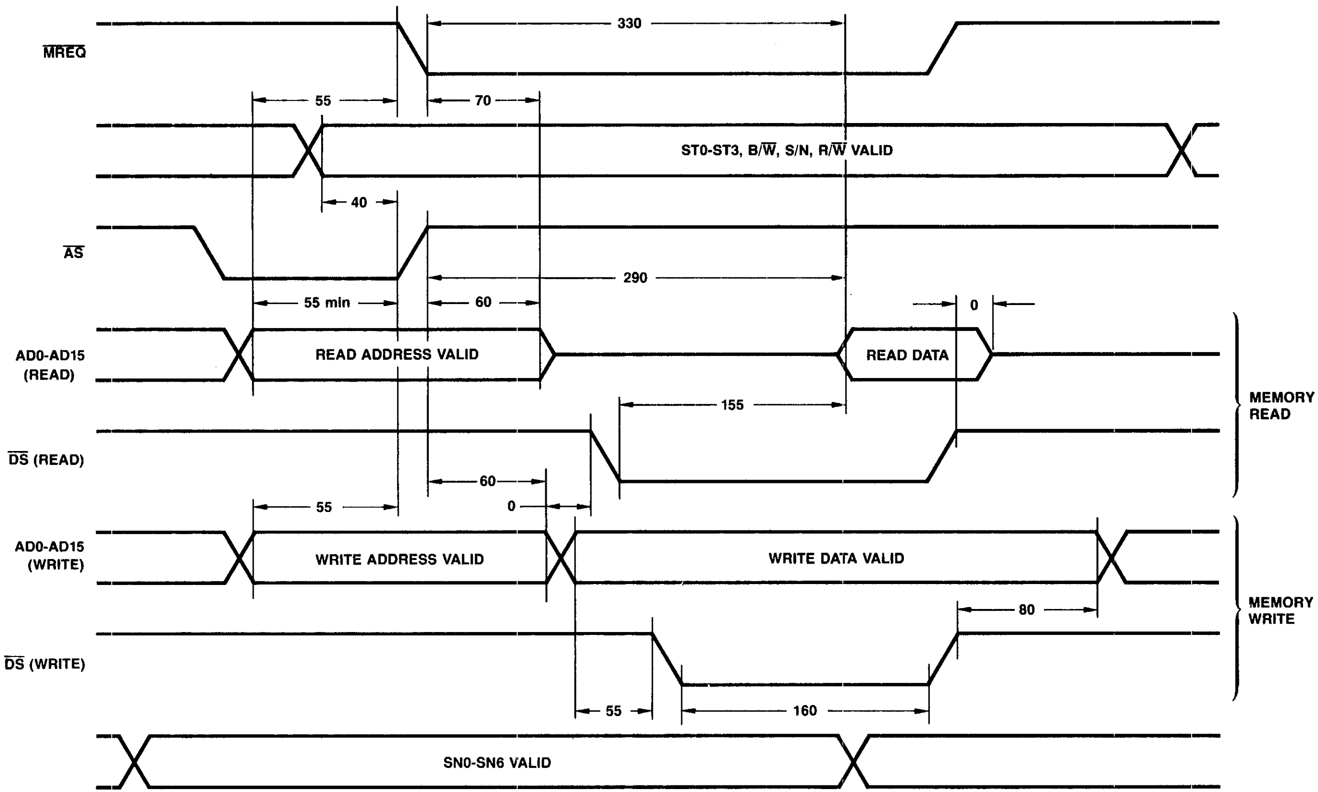


Figure 2-2. Memory Transaction Timing.

Am9124 Static Memory Design

A static memory implementation employing the Am9124 1K X 4 static RAM is shown in Figure 2.3. The memory has a 16 Kilobyte capacity and uses 32 Am9124's in an 8 X 4 array. The implementation makes use of the Am9124 power down feature. Since the CPU can access either bytes or words from memory, the array is organized into two separate banks, each 8-bits wide. The byte banks can be accessed in parallel or separately. One bank is considered as the upper (even) bank and the other is referenced as the lower (odd) bank. Bank selection is determined by two 25S138 one-of-eight decoders. Each decoder output is used to enable one row of 1K X 8 bits. Two decoders are used, not only to select between the two banks, but to power down the inactive bank. The decoders use latched address bits A11 to A13 to select which 1K X 8 row is enabled. The ADØ-AD15 bus is separated into address and data buses by 2 AmZ8104 octal 3-state transceivers and 2 AmZ8173 octal 3-state latches. The AmZ8104 buffer the data bus to the memory system and the AmZ8173's hold the address stable until the next transaction.

During byte writes the least significant address bit AØ (latched) determines which bank of memory (even or odd) is written to. If AØ is LOW, the upper or even byte (D8-D15) of the 16-bit memory word is addressed. The lower or odd byte (DØ-D7) is addressed if AØ is HIGH. The signals controlling the write enables are HIGH Byte Write ($\overline{HBW}$) and LOW Byte Write ($\overline{LBW}$). The table below shows their relationship to the B/$\overline{W}$ line and address bit Ø.

B/$\overline{W}$	AØ	$\overline{HBW}$	$\overline{LBW}$
H	L	L	H
H	H	H	L
L	L	L	L
L	H	L	L

Note that in normal circumstances, the bottom entry in the table shall not occur, since all memory word accesses should use even addresses i.e. AØ = LOW.

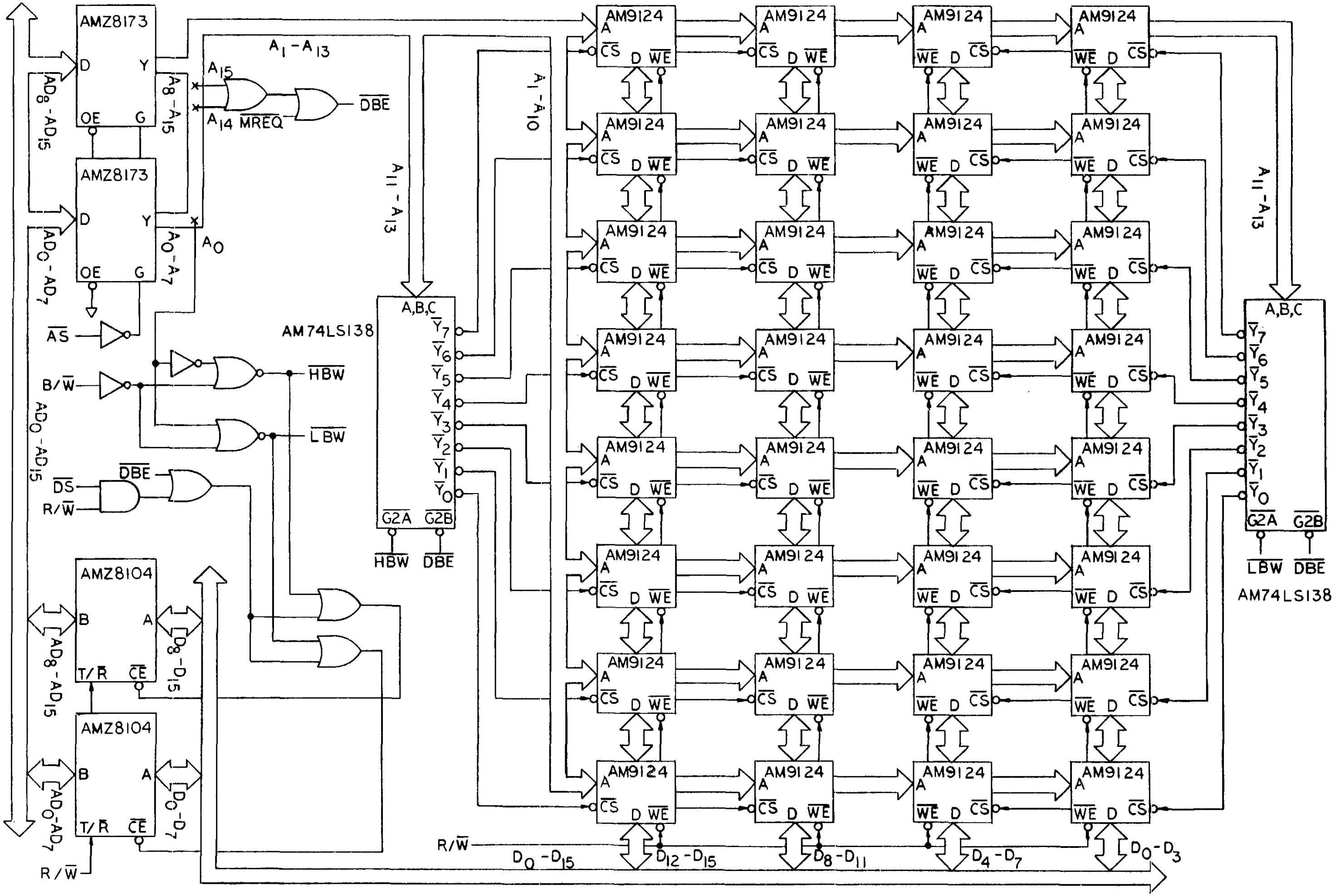


Figure 2-3. AmZ8000 8K x 16 Memory with Am9124.

The two bank select signals are also used in enabling the appropriate bidirectional tranceiver(s) to operate in conjunction with the enabled bank(s). The bank select signals are conditioned by the $\overline{\text{Data Strobe}}$ ($\overline{\text{DS}}$) and the Read/$\overline{\text{Write}}$ (R/$\overline{\text{W}}$) signals. This produces the appropriate gating of the buffers and the data timing for the CPU.

During a Read Cycle, the $\overline{\text{DS}}$ signal is used to produce the timing window for the data (refer to Figure 2.3). The enabling of the transceiver is actually determined by $\overline{\text{DS}}$ and R/$\overline{\text{W}}$ during Read Cycles and by R/$\overline{\text{W}}$ during Write Cycles. This method is used to prevent a bus contention problem that could exist between the memory and the transceivers. $\overline{\text{MREQ}}$ is used to indicate when a memory type operation is in process. After $\overline{\text{MREQ}}$ becomes active, data must be available to the CPU within 330ns. However, the 330ns should not be thought of as the total memory access time needed. Decoding and other logic times must be taken into account within the 330ns allowed. For example, the 25LS138 decoder needs approximately 40ns for worst propagation delay. While the only other critical logic is the OR gate that produces the Data Board Enable ($\overline{\text{DBE}}$) signal at an extra 10ns delay. The $\overline{\text{DBE}}$ is used to enable the G2A input of the two 25LS138 decoders. As for the bank select logic, the logic delay coincides within the $\overline{\text{MREQ}}$ high time and therefore can be neglected. Therefore, when using low-power Schottky, a 50ns delay should be subtracted from the time (330ns) data is required when $\overline{\text{MREQ}}$ goes low. Thus only 250ns is left for the memory $\overline{\text{CS}}$ low to data out valid. Both the Am9124C and Am9124E memory parts meet this parameter. Schottky 74S138 decoders may be used but the speed increase in this particular design does not enable the selection of the slower Am9124B part which requires 420ns for data access. During the write cycles, the data tranceivers are enabled by R/$\overline{\text{W}}$ being LOW. As in the read cycle, the direction of the data flow is controlled by R/W. Since the tranceivers never drive the CPU bus during a write cycle, there is no requirement to condition the tranceiver enable with $\overline{\text{DS}}$, as in the read case.

The memories are still deselected until $\overline{MREQ}$ goes LOW thus avoiding the contention problem. The memory write cycle begins when both $\overline{CS}$ and $\overline{WE}$ overlap and terminates when one goes high. Within this design, $\overline{CS}$ is shorter than $\overline{WE}$. Therefore $\overline{CS}$ conditionally starts and terminates the memory write cycle. Important memory parameters are the "Data In Valid to $\overline{CS}$ HIGH" and "$\overline{CS}$ LOW Enable" time. If slower memory devices must be used, then a wait state can be inserted in the memory transaction. The generation of fixed wait states was discussed generally in Chapter 1. In the case of a memory transaction requiring one wait state, however, the implementation shown in Figure 2.3 can be used. It consists of a 74LS74 D-type flip-flop and an OR gate. The flip-flop is normally set to produce a logical one at the $\overline{Q}$ output. When $\overline{AS}$ becomes low, the 74LS74 $\overline{Q}$ goes low. Therefore, $\overline{Q}$ only becomes low once during a complete machine cycle. And if the board is enabled ($\overline{DBE}$), a memory $\overline{WAIT}$ request is sent to the CPU for one extra clock cycle. The flip-flop is clocked with the inverted system clock. Thus the negative clock edge of the clock causes the 74LS74 $\overline{Q}$ output to reset. At the same edge, the CPU also samples the $\overline{WAIT}$ signal. Thus the $\overline{WAIT}$ flag is sampled and cleared in the same period and causes a 250ns wait state, at the nominal CPU clock frequency.

Am9016 Dynamic Memory Design

The design of a 64kb dynamic memory is shown in Figure 2.4. The implementation employs Am9016E 16K dynamic RAMs. Thirty-two devices are used, organized in a 16 X 2 array. Thus the memory has a width of 16-bits and a depth of 32K. The memory consists of two 8-bit wide banks. One bank contains all the even addressed bytes and the other contains all the odd bytes.

The CPU address/data bus is buffered using two AmZ8104 octal tranceivers. The CPU memory address is captured from the address/data bus using two AmZ8173 octal 3-state latches. The latches are strobed with $\overline{\text{AS}}$ inverted. Thus the address valid during the first part of the memory transaction is held valid until the start of the next transaction. The Am9106 RAMs are operated in the "early write" mode. That is write enable is applied to the device early in the transaction (if appropriate). The actual write into the device is timed from the column address strobe ($\overline{\text{CAS}}$) going LOW. The write enable generation is implemented with a 74LS153 4 input multiplexor. The multiplexor select lines are driven from B/$\overline{\text{W}}$ and R/$\overline{\text{W}}$. Thus the four possible transactions are;

R/$\overline{\text{W}}$	B/$\overline{\text{W}}$	$\overline{\text{ODSEL}}$	$\overline{\text{EVSEL}}$	
L	L	L	L	word write
L	H	$\overline{\text{AØ}}$	AØ	byte write
H	L	H	H	word read
H	H	H	H	byte read

$\overline{\text{ODSEL}}$ drives the lower (odd) bank write enables, and $\overline{\text{EVSEL}}$ drives the upper (even) bank write enables. The generation of a write enable is dependent upon the CPU accessing memory. This is defined by the STØ-ST3 lines. A 74LS139 2 to 4 decoder is enabled with ST1 and has the most significant select line driven by ST3. The least significant select line is grounded. Thus any memory access (data, stack or code) will generate a LOW on the third decoder output, Labelled $\overline{\text{MEMSEL}}$. In this implementation, there is no separation of memory spaces for data, stack and code.

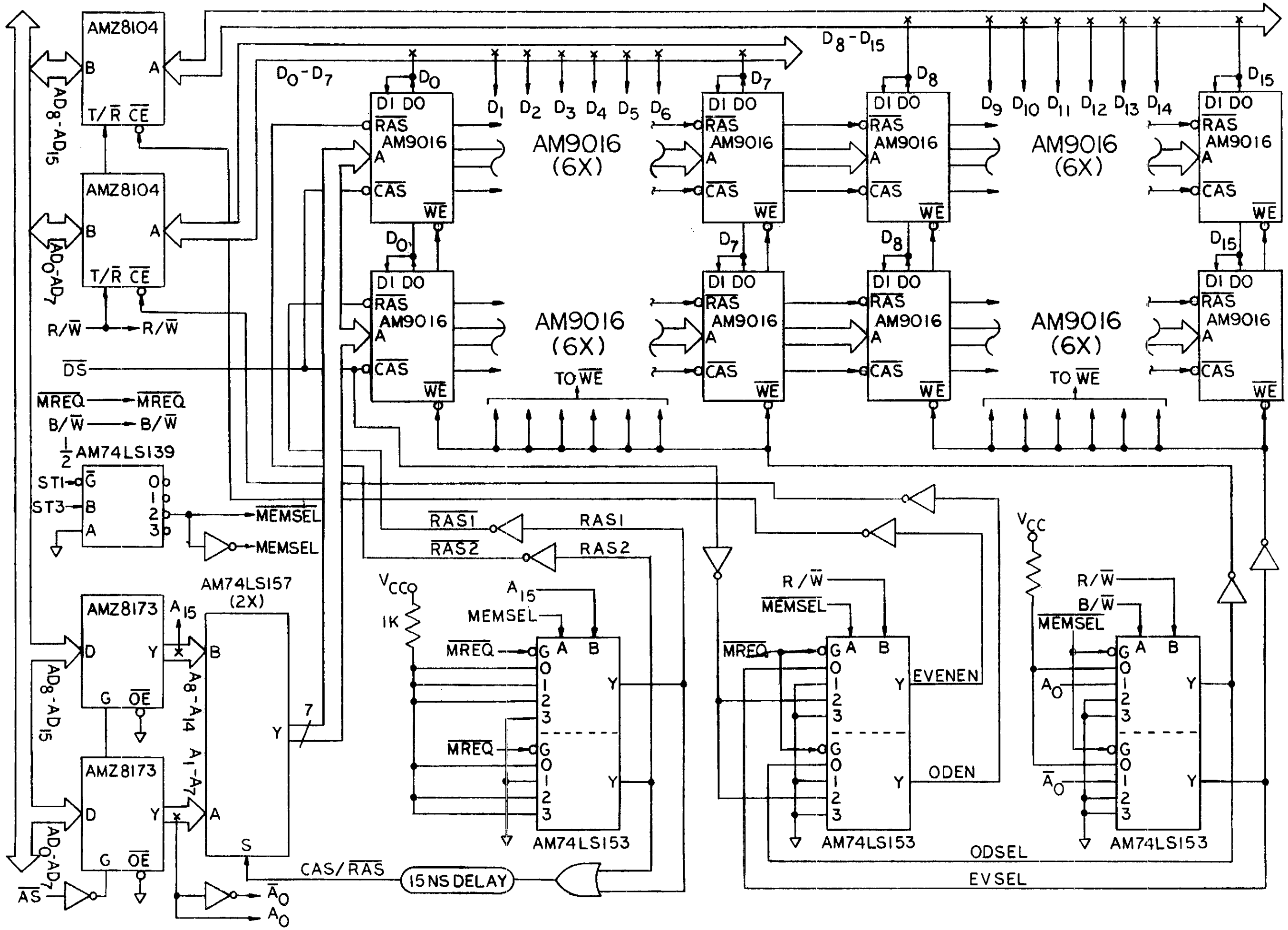


Figure 2-4. AmZ8000 32K x 16 Memory with Am9016.

The data tranceivers must be controlled to avoid bus contention, both between the CPU and the tranceiver and the tranceiver/memory interface: The tranceiver $T/\overline{R}$ line is driven from $R/\overline{W}$. When $R/\overline{W}$ is HIGH, the tranceiver drives from memory to CPU. The tranceiver enables are derived from a 74LS153 4 input multiplexor. The multiplexor select lines are driven from the $R/\overline{W}$ line and $\overline{MEMSEL}$ indicating a memory transaction. The four possible combinations are given below for $\overline{MREQ}$ driving the multiplexor enable LOW.

R/W	$\overline{MEMSEL}$	$\overline{EVENEN}$	$\overline{ODEN}$
L	L	$\overline{EVSEL}$	$\overline{ODSEL}$
L	H	H	H
H	L	$\overline{DS}$	$\overline{DS}$
H	H	H	H

If $\overline{MEMSEL}$ is HIGH, then the memory is not being accessed. Thus both tranceiver enables are inactive (HIGH). If $\overline{MEMSEL}$ is LOW, and the memory is being read, then both tranceivers are enabled if $\overline{DS}$ is LOW. If the memory is being written, then either one or both tranceivers are enabled, by the write enable signals that are applied to the memory devices. Thus, during a byte write, only one tranceiver is enabled. This avoids contention between the other tranceiver and the bank of memory not being written to.

The row and column address strobes are driven LOW sequentially, to strobe a 14-bit address into each device. One of two row address strobes are generated. $\overline{RAS1}$ is applied to one of the two rows of devices, or $\overline{RAS2}$ is applied to the other. Thus the row address strobe selects either the upper 16K words of memory or the lower 16K words of memory. A 74LS153 four input multiplexor generates $\overline{RAS1}$ and $\overline{RAS2}$. The multiplexor select lines are driven from the most significant address bit (latched) ADDR15, and MEMSEL. The four possible combinations (subject to $\overline{MREQ}$ enabling the multiplexor) are;

ADDR15	MEMSEL	$\overline{RAS1}$	$\overline{RAS2}$
L	L	L	L
L	H	L	H
H	L	L	L
H	H	H	L

If MEMSEL is LOW, then the transaction is not a memory read or memory write. However, if $\overline{\text{MREQ}}$ is LOW, the transaction is a refresh cycle initiated from the CPU. Under these circumstances, each device is read. Thus both $\overline{\text{RAS1}}$ and $\overline{\text{RAS2}}$ are driven LOW. If MEMSEL is HIGH, then a memory read or write is being executed. Dependent upon ADDR15 either the upper or lower half of memory is accessed.

Following $\overline{\text{RAS1}}$ or $\overline{\text{RAS2}}$ a column address strobe ($\overline{\text{CAS}}$) is generated from $\overline{\text{DS}}$. $\overline{\text{CAS}}$ is applied to all devices and strobes seven bits of column address into the device.

The device address lines must be driven with the current row and column addresses, synchronous with $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ generation. The latched address bits ADDR1-ADDR14 are split into a row address (ADDR8-ADDR14) and column address (ADDR1-ADDR7). The row and column addresses are applied sequentially to the memory devices through two 25LS157 2 input multiplexors. The multiplexor select line is driven by a logical OR of $\overline{\text{RAS1}}$ and $\overline{\text{RAS2}}$, delayed by 15ns. The 15ns delay ensures that the necessary row address hold time with respect to $\overline{\text{RAS}}$ is met.

Am9716 EPROM Interface

As described in Chapter 1, the CPU loads new program status from memory following initialization. Thus the requirement arises for a set of initialization parameters to be available in memory. The solution given in this example employes non-volatile memory which occupies part of the memory addressing space. Other system parameters may also be stored in this area. The implementation using 2 Am9716 2K X 8 EPROMs is shown in Figure 2.5. Since the EPROMs may only be read by the CPU, and not written, the memory can be designed for word reads only and does not require knowledge of the CPU B/$\overline{W}$ line. The access time of the Am9716 EPROM does not meet the CPU's requirements if the latter is executing a three clock-cycle memory read. This means that a wait state is required during the transaction to give the EPROMs sufficient time to respond with the read data.

The EPROM outputs are permanently enabled, by grounding the $\overline{CS}$ inputs, and are buffered from the CPU by 2 AmZ8144 octal 3-state drivers. The drivers are enabled if the current transaction is a memory read, the $\overline{DS}$ line is LOW and the CPU memory address is in the required range. The CPU memory address is latched using two AmZ8173 octal latches. The latch outputs are permanently enabled by grounding the output enable ($\overline{OE}$) lines. The latch is strobed with $\overline{AS}$, inverted and thus holds the address valid for the whole memory transaction. The EPROM is addressed by CPU addresses in the range 0000H - 0FFEH. The most significant four bits of latched address, together with ST3 and R/$\overline{W}$ are input to a 74LS138 3-8 decoder. If the most significant 4 address bits are LOW, ST3 is HIGH and R/$\overline{W}$ is HIGH, the 74LS138 generates a LOW on $\overline{PSEL}$, which enables the AmZ8144 buffers when $\overline{DS}$ goes LOW. The remaining latched CPU address bits, with the exception of address bit Ø, are applied directly to the EPROM devices to designate one of 2K word locations. The access time for the Am9716 EPROM is 450ns which exceeds the access time required by the CPU. The insertion of 1 wait state in the three clock cycle memory transaction, however, solves the problem. A 74LS74 flip-flop is set at the start of the memory transaction by the CPU $\overline{AS}$ signal going LOW. The flip-flop Q output is not driven LOW until the next falling clock edge which strobes a LOW from the D input to the Q output.

The $\overline{Q}$ output of the flip-flop drives the CPU $\overline{WAIT}$ line through a 74S02 enabled with $\overline{PSEL}$ and an inverter. If $\overline{PSEL}$ is LOW, indicating an EPROM access, the CPU $\overline{WAIT}$ line will be driven LOW until the falling clock edge in T2. This ensures the insertion of one 250ns wait state in the CPU/EPROM transaction.

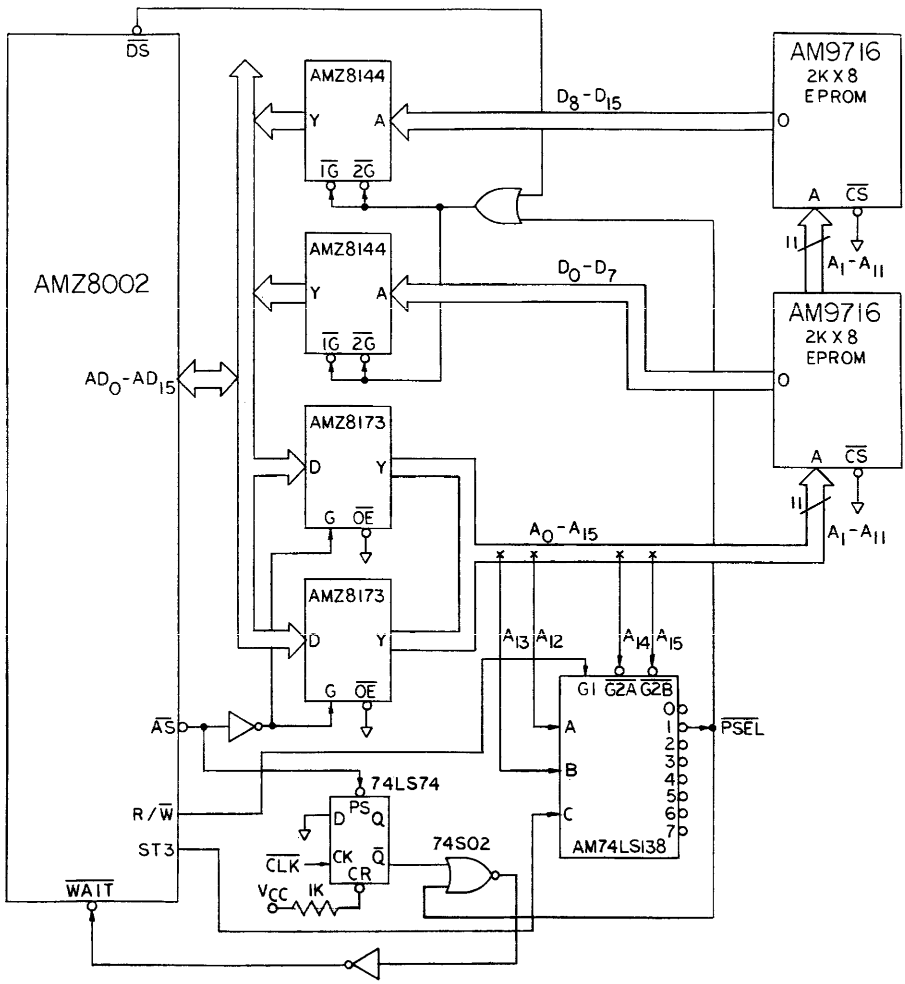


Figure 2-5. AmZ8000 to Am9716 Interface.

CHAPTER 3
I/O INTERFACING TECHNIQUES

Introduction

This chapter discusses some techniques involved in interfacing I/O devices to the AmZ8000 system. The initial discussions explore the nature of the AmZ8000 I/O transaction, in terms of data format and timing. This is followed by a description of some interfacing techniques for three levels of I/O connection; programmed I/O; programmed I/O with interrupts and programmed I/O with interrupts and DMA. The chapter concludes with some examples of I/O interfacing, which demonstrates how non-AmZ8000 peripherals can be connected to the AmZ8000 system.

Types of I/O Interface

In general, an I/O device may interface to an AmZ8000 system with varying levels of privilege. The least privileged interface is the programmed I/O connection. The device may only communicate with the CPU when the latter initiates a transaction. The transactions are restricted to CPU reads and writes using addressable locations within the I/O device.

The interface is shown in Figure 3.1. The address decoding determines when the I/O device is being accessed. The CPU port address and/or the STØ-ST3 lines generate the necessary chip select requirements for the device. The interface buffers data between the ADØ-AD15 bus and the I/O device. In some instances this may be a straight connection between the I/O device and the address/data bus. In other applications it may be bidirectional TTL buffers. The control section of the interface generates the required I/O commands from the CPU controls signals.

The next level of I/O interface gives the I/O device more privilege. As well as communicating with the CPU by means of programmed I/O transaction, the I/O device can now interrupt the CPU to request a transaction. The interface is shown in Figure 3.2. It comprises the programmed I/O section, described above, together with a section to handle interrupts. The interface must detect an interrupt requirement within the I/O device and generate an interrupt request to the CPU. The interface must also recognize an interrupt acknowledge from the CPU, encoded on the STØ-ST3 lines and cause the necessary response from the I/O device. The response required from the I/O device may vary dependent upon the type of interrupt involved. A device generating a vectored interrupt, for example, must return information to the CPU during an acknowledge cycle. This information is interpreted as a vector by the CPU.

The third and highest privileged I/O connection to the AmZ8000 system provides the I/O device with a direct memory access (DMA) capability, in addition to the facilities described above. Figure 3.3 shows the block diagram.

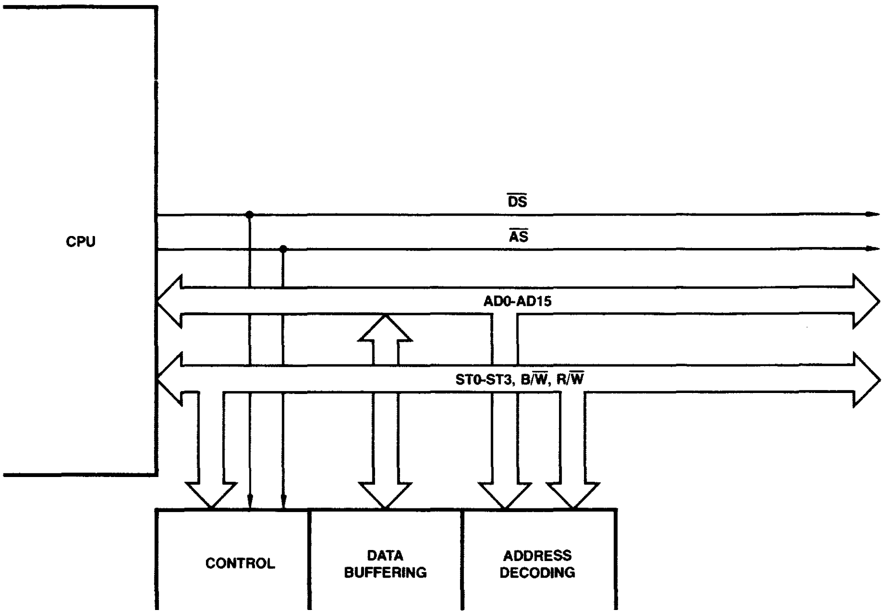


Figure 3-1. Programmed I/O Interface.

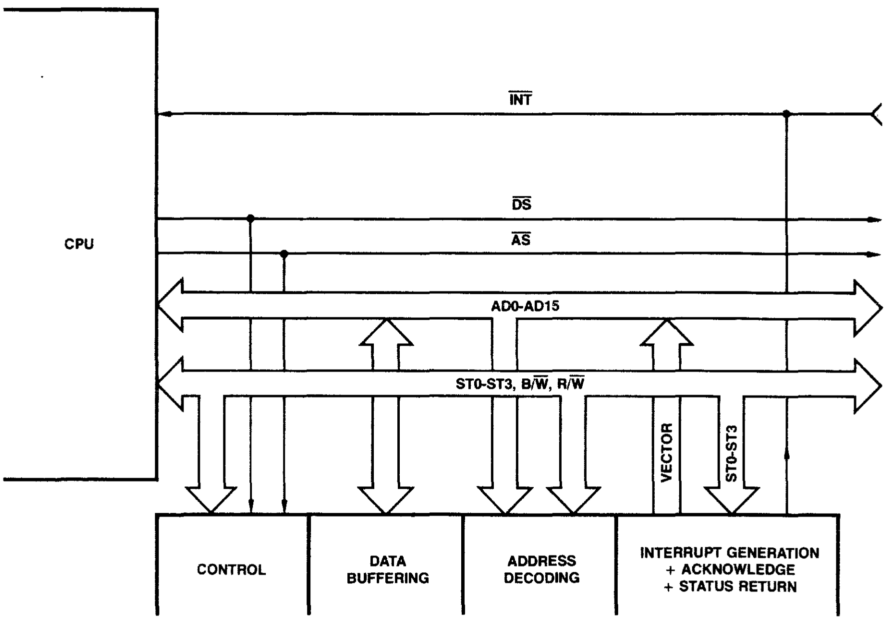


Figure 3-2. Programmed I/O and Interrupts.

The DMA interface must be capable of controlling transfers directly between the I/O device and memory. Facilities exist within the AmZ8000 CPU's to inhibit the CPU from controlling the bus during the period that the DMA interface is conducting a transaction. The DMA interface gains control of the system, while causing the CPU to enter an inactive state. This is achieved using the bus arbitration signals described later in this chapter. The other portion of the DMA interface supports the generation of the necessary control signals that would normally be generated by the CPU, to enable the I/O device to perform direct transactions with memory.

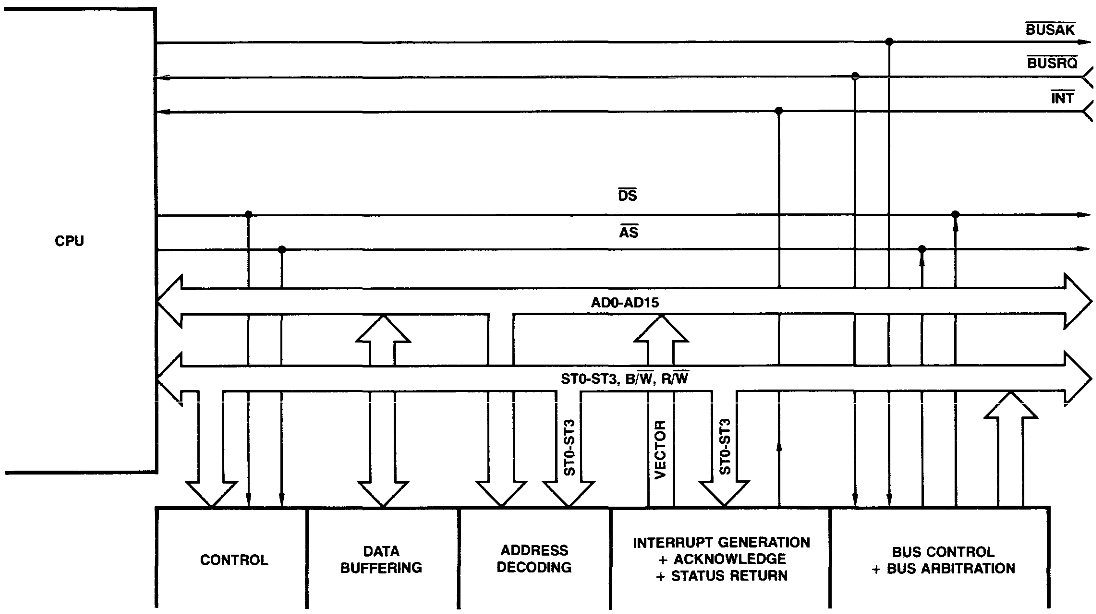


Figure 3-3. Programmed I/O and Interrupts and DMA.

Programmed I/O Address Formats

During every programmed I/O transaction, the CPU outputs an I/O address on the ADØ-AD15 bus. The I/O or port address is 16 bits wide. The least significant address bit determines which half of the data bus will be used for the transaction.

Programmed I/O Status Encoding

Two types of programmed I/O transaction are possible in the AmZ8000 system: Normal I/O and Special I/O. These are identical in operation but cause different values to appear on the status lines. A Normal I/O transaction, which takes place whenever the CPU is executing a normal I/O instruction, is indicated by LLHL on the ST3-STØ lines respectively. A Special I/O transaction which takes place whenever a special I/O instruction is being executed is indicated by LLHH on the ST3-STØ lines respectively.

Programmed I/O Data Formats

The CPU may communicate with an I/O device on either a word or byte basis, dependent upon the type of instruction being executed. The data formats on the address/data bus are shown in Figure 3.4. Word transactions for both input and output instructions take place on the ADØ-AD15 bus. Byte transactions, however, are a little more complex. When inputting a byte, the CPU reads either the upper or lower half of the address/data bus dependent upon the least significant port address bit. If the least significant bit is LOW (the port address is even), the CPU reads the upper half of the ADØ-AD15 bus. If the least significant bit is HIGH (the port address is odd), the CPU reads the lower half of the ADØ-AD15 bus.

When outputting a byte, CPU operation is the same for both the odd or even port address; the byte output data is duplicated onto both the upper and lower halves of the ADØ-AD15 bus. The designer must use the least significant port address bit and the $B/\overline{W}$ line to determine whether a device on the upper or lower half of the bus should be written to during a byte output instruction. To remain consistent with the Input operation, the designer should ensure that an even port address causes a byte write to a

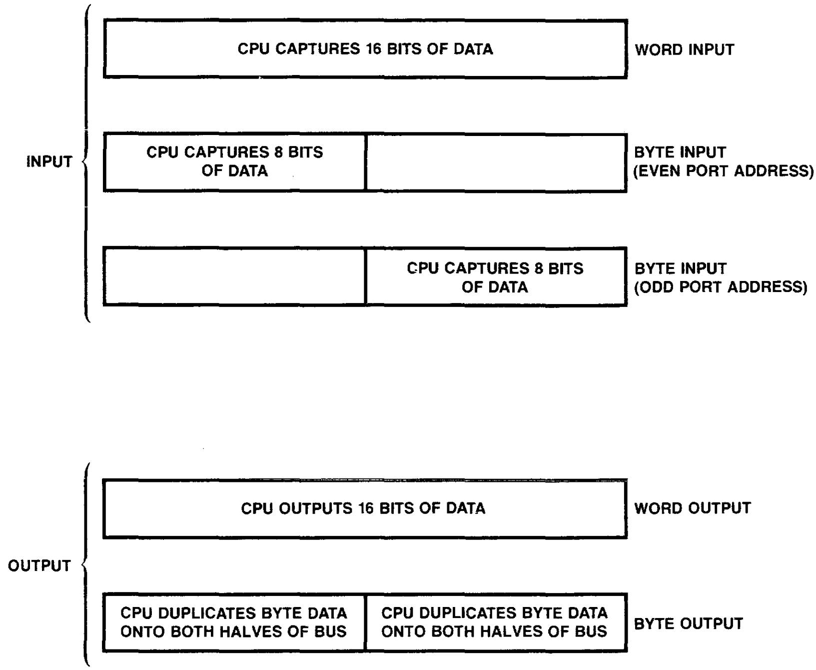


Figure 3-4. Input/Output Data Formats.

device on the upper half of the address/data bus, and that an odd port address causes a byte write to a device on the lower half of the address/data bus.

Programmed I/O Cycle Timing

I/O Cycle timing in the AmZ8000 system can be described in terms of the two CPU bus timing signals, $\overline{AS}$ and $\overline{DS}$. As with all CPU transactions, the $\overline{AS}$ signal has two functions. Firstly, it signifies the start of a new transaction and secondly, the trailing edge of $\overline{AS}$ indicates that an address has been set up on the ADØ-AD15 bus for a minimum of 55nS, as shown in Figure 3.5. The valid address is held on the bus for a minimum of 60ns following the trailing edge of $\overline{AS}$ which also indicates that CPU signals STØ-ST3, B/$\overline{W}$ and R/$\overline{W}$ have been set up for 40ns minimum. These CPU output signals remain valid for the whole I/O transaction. In the input transaction, $\overline{DS}$ low indicates that the responding I/O device may drive the address/data bus with the input data. (In practical terms, $\overline{DS}$ will be used to turn on the responding devices output buffers.) The trailing edge of $\overline{DS}$ indicates that the CPU has captured the input data which need not be held valid by the responding device for any further time.

The CPU requires a certain response time from the I/O device. Input data must be available on the address/data bus 315ns after the leading edge of $\overline{DS}$, at the latest. In terms of the response time in relation to $\overline{AS}$, data must be available 540ns after the trailing edge of $\overline{AS}$, at the latest. If a responding device cannot meet this requirement, the CPU may be delayed from completing the I/O transaction by the insertion of wait cycles. Each wait cycle delays the completion of the transaction by 250ns (assuming a 4MHz CPU clock). Thus the responding unit is given extra time, in increments of 250ns, to complete the transaction. The CPU will execute a wait cycle when its $\overline{WAIT}$ line is driven LOW no later than 340ns after the trailing edge of $\overline{AS}$.

Output transactions are similar to input transactions, but $\overline{DS}$ is interpreted differently in this case. The leading edge of $\overline{DS}$ now indicates that the CPU has removed the port address off the address/data bus, and replaced it with valid data. The data is set up on the address/data bus for at least 55ns prior to the leading edge of $\overline{DS}$, as shown in Figure 3.5. In practical terms, this edge can be used to strobe data into the I/O device, provided the required device set-up times are met. The trailing edge of $\overline{DS}$ indicates that data will be held valid on the address/data bus for a further 80ns minimum.

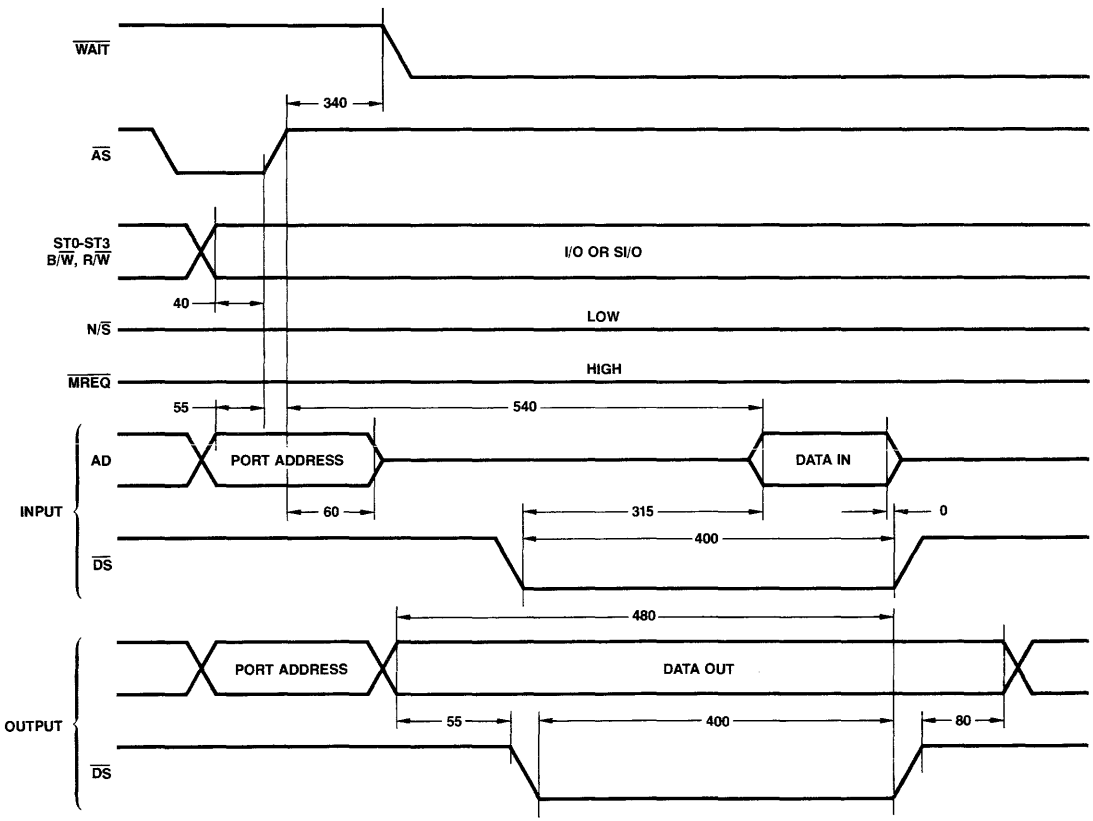


Figure 3-5. Programmed I/O Timing.

Interrupt Protocol and Prioritization

The AmZ8000 family features three types of interrupts. In order of decreasing CPU priority, these are Non-Maskable Interrupt, Vectored Interrupt and Non-Vectored Interrupt. Each type of interrupt, however, may have multiple sources and therefore requires prioritization which is implemented by means of a daisy chain external to the CPU i.e. permission to interrupt is passed from higher priority peripherals down to lower priority peripherals on the daisy chain.

Figure 3.6 shows an example of an interrupt scheme on an AmZ8000 system. Eight devices are shown and each device is capable of interrupting the CPU. Devices A, B and C issue Non-Maskable Interrupts, devices D and E issue Vectored Interrupts, and devices F, G and H issue Non-Vectored interrupts. As mentioned above, the CPU prioritizes the three interrupt types. Within one type of interrupt, however, a further prioritization takes place among the devices capable of issuing an interrupt of that type. This prioritization is implemented without CPU involvement, using a daisy chain. In the Figure shown, device D will take priority over device E when a vectored interrupt is issued. Devices A, B and C are similarly prioritized with respect to each other using another daisy chain and so on.

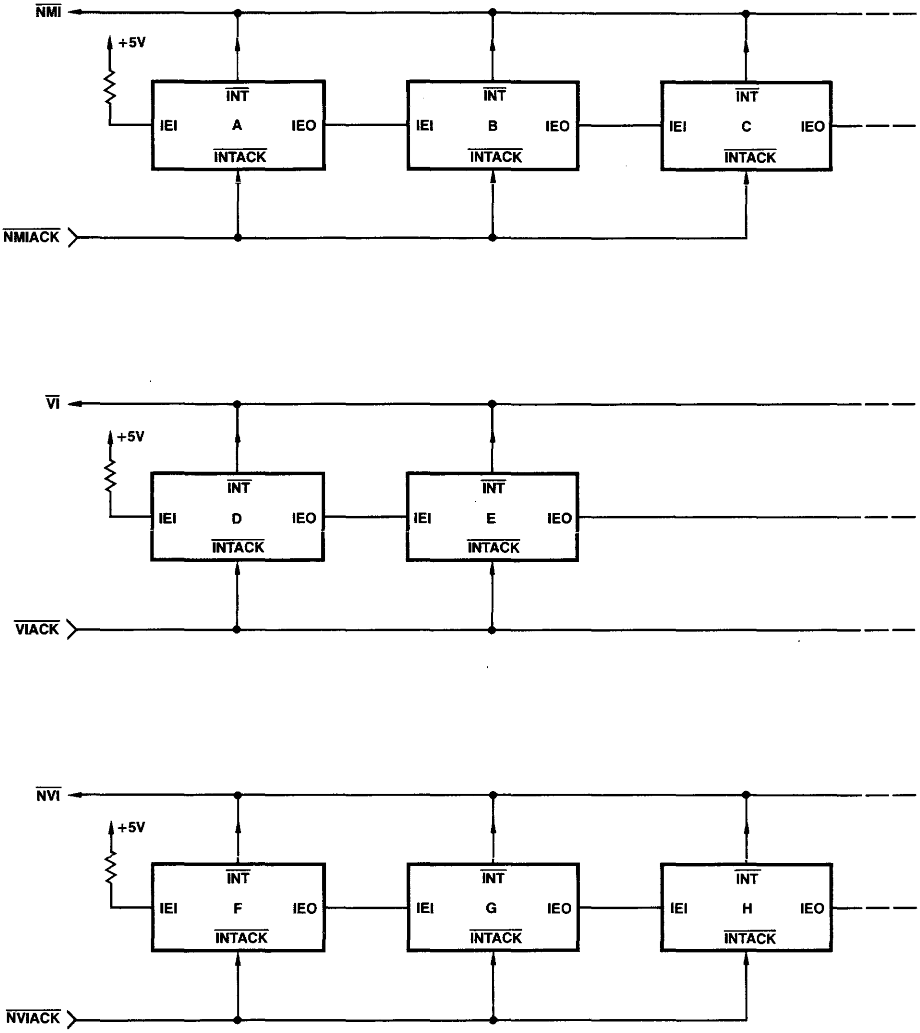


Figure 3-6. Interrupt Scheme.

To implement the priority scheme, four signals are defined for each of the interrupt types mentioned previously.

Interrupt Request (VI/NVI/NMI)	One of these inputs to the CPU is driven LOW by a device requesting an interrupt.
Interrupt Acknowledge ($\overline{\text{INTACK}}$)	This output (encoded in STØ-ST3 on AmZ8001/2) is used to acknowledge the interrupt request.
Interrupt Enable In/Interrupt Enable Out (IEI/IEO)	These input and output signals are used to implement the daisy chain for each device. IEI is an input to a device which grants the device permission to interrupt, and IEO is the permission grant output by a device to a lower priority device on the daisy chain. The IEI of a lower priority device is driven by the IEO of the next higher priority device on the chain. Under quiescent circumstances (no interrupts) IEO output by a device follows the IEI input to the device.

Each device has a number of control bits which are also involved in the interrupt protocol.

Interrupt Under Service (IUS)	This control bit when HIGH indicates that a device is currently having an interrupt serviced. It is not reset until the service is complete. The IUS bit provides the mechanism for inhibiting lower priority interrupts during a service routine. This bit is writeable by programmed I/O.
Interrupt Enable (IE)	This control bit inhibits the device from requesting an interrupt when LOW. It is writeable by programmed I/O.
Interrupt Pending (IP)	This bit records the devices interrupt requirement. Under suitable conditions, IP HIGH generates an interrupt request. This bit is readable allowing the device to be polled by the system. The bit is also writeable for debugging purposes.

Disable Lower Chain (DLC)	This is a writeable bit which enables the device to unconditionally force its IEO line LOW, disabling all lower priority interrupts. HIGH = force IEO LOW
No Vector (NV)	When HIGH this bit inhibits the device from returning any form of status during an Interrupt Acknowledge cycle.
Vector Includes Status (VIS)	A LOW on this bit sets the status returned by the device to a value pre-loaded into the device. A HIGH on this bit allows the status returned to be modified by the device's internal status.

DEVICE PROTOCOL

Interrupt generation from any AmZ8000 device must commence with the setting of the IP bit within the device. The IP bit signals the interrupt requirement to the device's interrupt logic. The IP bit being set will cause an interrupt request ($\overline{INT}$ = LOW) if the following conditions are met.

IEI = HIGH; IEI is the permission to interrupt, handed down via the daisy chain from a higher priority device. Any device currently being serviced denies permission to interrupt to devices lower on the daisy chain.

IUS = LOW; If the device is presently undergoing an interrupt service (IUS = HIGH), the interrupt request is delayed until service completion.

$\overline{INTACK}$=HIGH; During an interrupt acknowledge cycle, the daisy chain is shielded from any further interrupt requests, to allow the Chain to settle. Thus a device requiring an interrupt during an acknowledge cycle must wait until the completion of the acknowledge before issuing the request.

IE = HIGH; The device may have an internal Interrupt Enable control bit, writeable from the CPU. This is a convenient feature for inhibiting a device from interrupting.

When one or more devices have met the above conditions, an interrupt request is made to the CPU. The latter, after some delay, acknowledges the request by executing an interrupt acknowledge cycle. During the interrupt acknowledge cycle the CPU STØ-ST3 signals signify one of the three types of interrupt acknowledgements. The interrupt acknowledge cycle achieves two results. Firstly, it freezes the daisy chain, allowing one device to have priority over the rest, and secondly it captures any status returned by the device for interrupt identification.

The device prioritization is achieved independently of the $\overline{\text{INT}}$ line that signified the interrupt request to the CPU. Instead, the IP bit is used in the prioritization. An interrupt device is given priority over all other devices on the chain (IUS=HIGH) if the following conditions are met.

IEI = HIGH During the interrupt acknowledge cycle, permission is given from a higher priority device to a lower priority device to allow its IUS bit to be set. This permission is denied (IEI = LOW) if a higher priority device has IP HIGH and IUS LOW, indicating that it requires service from the CPU. Note that the conditions driving IEI LOW are different during the $\overline{\text{INTACK}}$ cycle, than at all other times in the system operation.

IUS = LOW; A device is only given priority if it does not have an interrupt currently being serviced.

IE = HIGH; The same condition applies as that discussed above.

Interrupt Sequencing and Acknowledgement

The AmZ8000 CPU's respond to an interrupt request with an interrupt acknowledge cycle. The interrupt acknowledge cycle must accomplish two tasks. It must freeze the daisy chain from further stimuli (interrupts) and capture an identifier from the interrupting device that has highest priority on the daisy chain , defined by its IUS bit being set HIGH.

Any device requesting a non-vectored or vectored interrupt should maintain the interrupt request until the acknowledge cycle. The non-maskable interrupt is edge detected by the CPU and therefore need not obey this requirement. The Interrupt Acknowledge is indicated on the ST3-STØ outputs: Three values on ST3-STØ correspond to the three types of interrupt acknowledgement. ST3-STØ outputs of LHLH respectively indicate a Non-Maskable Interrupt Acknowledge. ST3-STØ outputs of LHHL indicate a Non-Vectored Interrupt Acknowledge and ST3-STØ outputs of LHHH indicate a Vectored Interrupt Acknowledge. Figure 3.7 shows the timing of an acknowledge cycle.

The start of the acknowledge cycle is indicated by a LOW on the $\overline{AS}$ output. At this time, no further interrupts will be allowed ensuring that the daisy chain will settle. The next event in the acknowledge cycle is timed from the leading edge of $\overline{DS}$. The interrupting device that has the highest priority on the daisy chain sets its IUS bit on the leading edge of $\overline{DS}$. Thus the interval from the start of the acknowledge cycle to the leading edge of $\overline{DS}$ must be large enough to ensure that any rippling through the daisy chain has settled. At the nominal clock frequency this interval is 960ns. If more time is required, the CPU $\overline{WAIT}$ line can be driven LOW at the appropriate time to cause the CPU to enter a wait state. Each wait state increases the interval specified above by 250ns (at the nominal clock frequency).

Following the setting of IUS, the device can return an identifier to the CPU. This data must be available on the CPU address/data bus 420ns after the leading edge of $\overline{DS}$ at the latest. If the device requires more time to return the identifier, the CPU $\overline{WAIT}$ line can be driven LOW at the appropriate time. The CPU indicates data capture by driving $\overline{DS}$ HIGH. The data need not be held valid by the device for any further time.

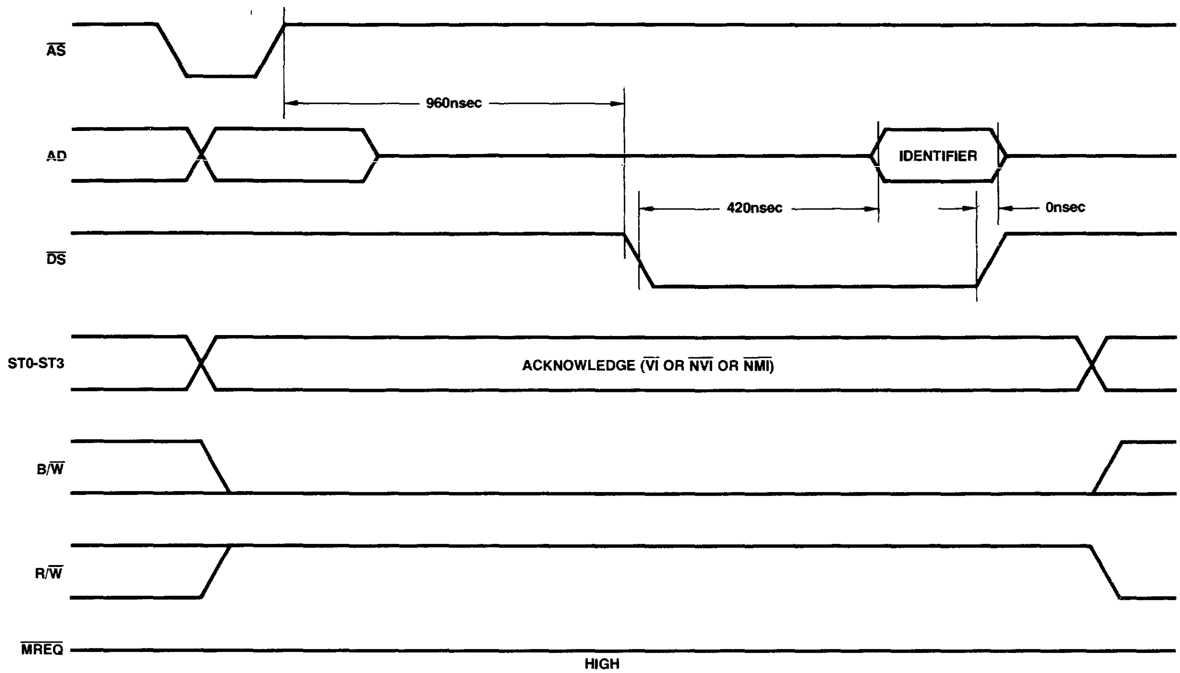


Figure 3-7. Interrupt Acknowledge Cycle.

DMA Operation

The third type of I/O connection is the direct memory access (DMA) interface. If a peripheral device requires attention, it does not interrupt the CPU, but requests service from a DMA Controller (DMAC). The DMAC then requests the bus from the CPU and performs the necessary system transaction with the CPU inactive. Following completion of the transaction, the DMAC returns the bus to the CPU. The advantages of such a scheme are two-fold. First, the DMAC may be able to perform transactions between I/O and memory more efficiently than the CPU. Second, the overhead involved in obtaining the bus from the CPU may be significantly less than the overhead involved in a peripheral device directly interrupting the CPU.

The main functions of a DMAC are bus arbitration to de-activate the CPU during a DMA transaction and bus manipulation, to effect the required transaction. In the bus environment, the CPU is the default bus master, and always regains control of the bus following DMA activity. The CPU cannot use the bus arbitration facilities to gain control of the bus, but can only give the bus away and passively wait for its return. The prioritization of devices requesting the bus from the CPU is implemented as a daisy chain. Figure 3.8 shows the daisy chain configuration, which is implemented as three signals.

$\overline{\text{BRQ}}$: Bus Request (Input). This CPU line is driven LOW by a device to gain control of the bus. It is also used as a status line by the peripheral device prior to the issuance of a request, to avoid clashes.

$\overline{\text{BUSAK}}$: Bus Acknowledge (Output). This signal is output by the CPU and when LOW informs the external devices that is has given up bus mastership.

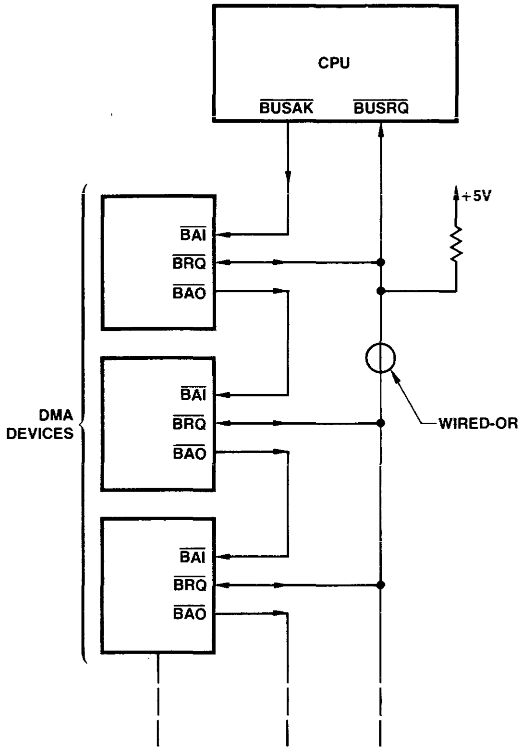


Figure 3-8. Bus Request Daisy Chain.

$\overline{\text{BAO}}$: Bus Acknowledge Out (Output). This signal is output by each peripheral device and is used in the implementation of the daisy chain. When LOW it signifies acknowledge.

$\overline{\text{BAI}}$: Bus Acknowledge In (Input). This peripheral input line is driven from the $\overline{\text{BAO}}$ of the next higher peripheral on the chain. When LOW it signifies Acknowledge.

The $\overline{\text{BRQ}}$ line is bidirectional, and is used as both a status line and the request line. The daisy chain is implemented by means of the bus acknowledge in and bus acknowledge out signals which pass down the the acknowledge from a higher priority device to a lower priority device.

The protocol each device must obey is shown in Figure 3.9. If a device detects a requirement to use the bus its first action is to examine the $\overline{\text{BRQ}}$ input. If this is active (LOW) then either another device is requesting or has control of the bus and a Bus request cannot be made. The device therefore requests at a later time. If the $\overline{\text{BRQ}}$ line is inactive, then the device is able to drive $\overline{\text{BRQ}}$ LOW, and request the bus. The bus acknowledge is given by the $\overline{\text{BUSAK}}$ signal LOW from the CPU. The acknowledge is given to the highest priority device on the bus and is passed down the daisy chain to the highest priority device that had requested the bus. A requesting device may only use the bus on receipt of the acknowledge. Until then the device remains with the request asserted LOW. After having used the bus, the device deactivates its request, releasing control of the bus and passing the acknowledge on down the daisy chain. In the initial contention for the bus, one or more lower priority devices may have contended unsuccessfully with the device that gained control of the bus. These devices continue to assert their request and will use the bus when it is released. The CPU does not regain the bus until all the devices involved in the contention have been granted the bus. Note that these multiple requests can only occur in the interval between a device inspecting the $\overline{\text{BRQ}}$ line and then asserting its $\overline{\text{BRQ}}$ output i.e. during this interval several devices can each assert their $\overline{\text{BRQ}}$ lines.

In addition to bus arbitration, the DMA interface must generate the control signals necessary for managing the transactions between the I/O device and memory. When the CPU is rendered inactive by the bus arbitration, ADØ-AD15, STØ-ST3, $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, B/$\overline{W}$, N/$\overline{S}$ and $\overline{MREQ}$ all enter the high impedance state. At this point these signals must now be generated by the DMA interface, which must emulate the CPU in its control of the system.

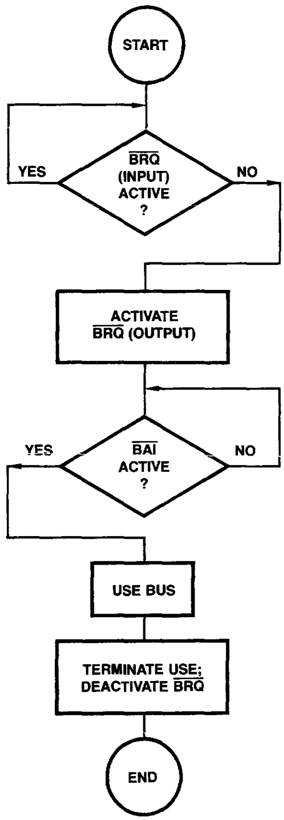


Figure 3-9. Bus Request Protocol.

Bus Request/Acknowledge Sequencing

A DMA interface requests control of the bus by driving the $\overline{BUSRQ}$ signal LOW. The CPU responds to the request by driving the $\overline{BUSAK}$ output LOW some time later, as shown in Figure 3.10. The latency involved here will most frequently be between 3 and 6 clock cycles, with a possible worst case of 20 clock cycles. After this latency period, the CPU drives its $\overline{BUSAK}$ output LOW. At this time the CPU bus signals ($\overline{AS}$, AD0-AD15, $\overline{MREQ}$, $\overline{DS}$, ST0-ST3, B/$\overline{W}$, R/$\overline{W}$ and N/$\overline{S}$) will all be in the high impedance state, allowing the DMA interface to take control of the bus.

When the DMA interface has finished using the bus, it must drive the CPU $\overline{BUSRQ}$ input HIGH. After 1-2 clock cycles, the CPU responds by driving the $\overline{BUSAK}$ output HIGH. By this time the DMA interface should have ceased driving the bus signals allowing the CPU to regain control of the bus and recommence execution.

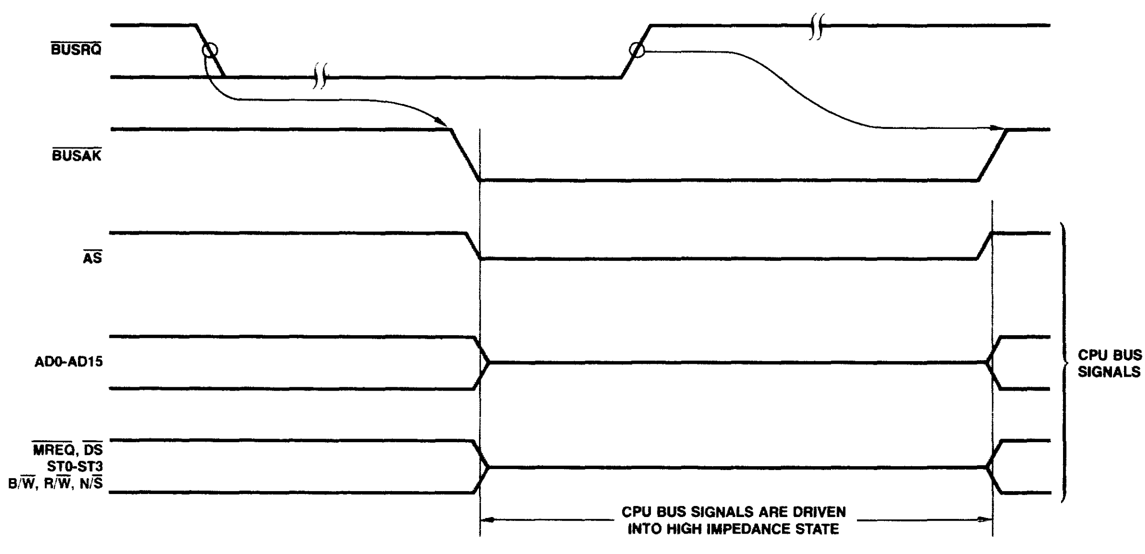


Figure 3-10. Bus Request/Acknowledge Timing.

Resource Request Operation

The AmZ8000 family offers facilities for resolving requests from multiple CPU's for a common shared resource, such as a memory or peripheral device. The prioritization of multiple requests is handled using a daisy chain as shown in Figure 3.11. Each of the boxes represents a CPU with some interface logic to be described below. The daisy chain is implemented using four signals.

$\overline{\mu RQ}$: Multimicro request (output). A LOW on this line constitutes a resource request and is asserted by a CPU that requires the resource.

$\overline{\mu ST}$: Multimicro Status (Input). A LOW on this line signifies resource busy. This line is inspected by a CPU prior to making a request. If the line is LOW, no request is made.

$\overline{\mu AI}$: Multimicro Accept In (Input). A LOW on this line which is passed serially through each of the daisy chained CPU's constitutes acceptance of a resource request. Acceptance is passed down through the daisy chain following a request.

$\overline{\mu AO}$: Multimicro Accept Out (Output). A LOW on this output indicates that a daisy chained CPU is passing the accept accept down to a lower CPU in the chain. The $\overline{\mu AO}$ line drives the $\overline{\mu AI}$ line of the next lower CPU on the chain.

A CPU, before being able to gain the resource, must obey a protocol which commences with an inspection of the $\overline{\mu ST}$ line. If this line is LOW the resource is busy and no requests can be made. If the line is HIGH, the resource is not busy and the CPU can generate a request by driving the $\overline{\mu RQ}$ line LOW. This has the effect of driving the $\overline{\mu ST}$ inputs of each

CPU LOW, inhibiting them from making further requests. The LOW on $\overline{\mu RQ}$ drives the $\overline{\mu AI}$ line of the highest priority CPU LOW. This indicates that the highest priority CPU may use the resource, if it has requested. If it has not requested, it drives the $\overline{\mu AO}$ line LOW, which in turn drives the $\overline{\mu AI}$ line of the next lower CPU LOW. In this manner, the accept is passed down the chain until is reaches a CPU which did make a request. That CPU interprets the LOW on $\overline{\mu AI}$ as the accept, confirming that it has gained the resource. The requesting CPU also blocks the accept from travelling any further down the daisy chain, by maintaining its $\overline{\mu AO}$ line HIGH.

In practical terms, the implementation of this four wire chain is wasteful of CPU pins. The AmZ8000 CPU's implement this prioritization with just two lines.

$\overline{\mu O}$: Multimicro Out (Output). A LOW on this CPU output signifies a multimicro request. It may be driven LOW by CPU instructions. (MREQ and MSET).

$\overline{\mu I}$: Multimicro In (Input). This input serves a dual purpose in the protocol. It is inspected by CPU instructions (MBIT, MREQ), to determine the state of the resource and whether a request is accepted.

The mapping of the CPU lines to the daisy chain is shown in Figure 3.12. The $\overline{\mu O}$ line from the CPU drives the $\overline{\mu RQ}$ line, through an open collector 'S07 buffer. This line also controls an 'S157 quad two input multiplexer which drives the CPU $\overline{\mu I}$ line. If a request has not been made by that CPU ($\overline{\mu O}$ = HIGH), $\overline{\mu I}$ is driven from the $\overline{\mu ST}$ line of the daisy chain. If a request has been made ($\overline{\mu O}$ = LOW), $\overline{\mu I}$ is driven from the $\overline{\mu AI}$ line of the daisy chain. This multiplexing function does not restrict the protocol described above since the requesting CPU only examines the $\overline{\mu ST}$ line prior to driving $\overline{\mu O}$ LOW and furthermore, only examines the $\overline{\mu AI}$ line after driving the $\overline{\mu O}$ line LOW. The 'S157 multiplexer is also used to gate the $\overline{\mu AI}$ line to the $\overline{\mu AO}$ line. If the $\overline{\mu O}$ line is HIGH, $\overline{\mu AO}$ follows $\overline{\mu AI}$. If the $\overline{\mu O}$ line is LOW, $\overline{\mu AO}$ is set unconditionally HIGH inhibiting the accept from futher propagation down the daisy chain.

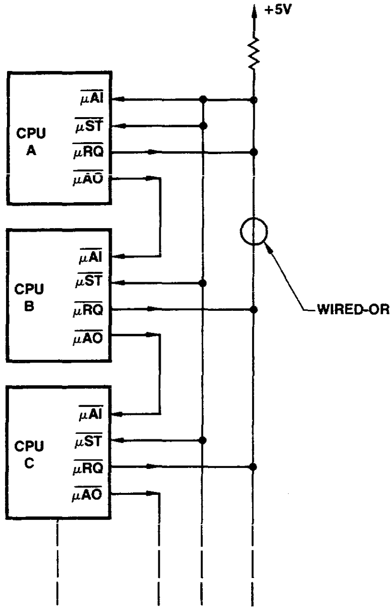


Figure 3-11. Multimicro Daisy Chain.

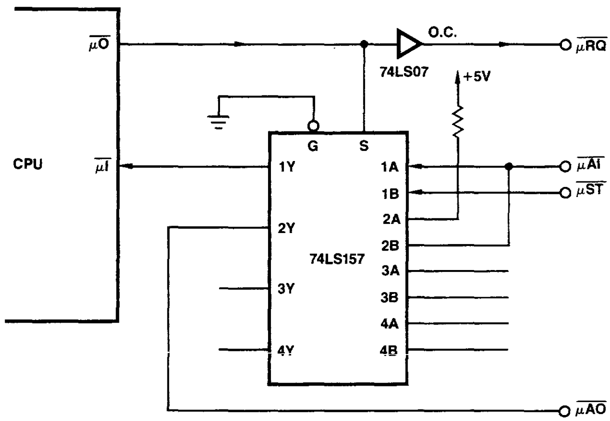


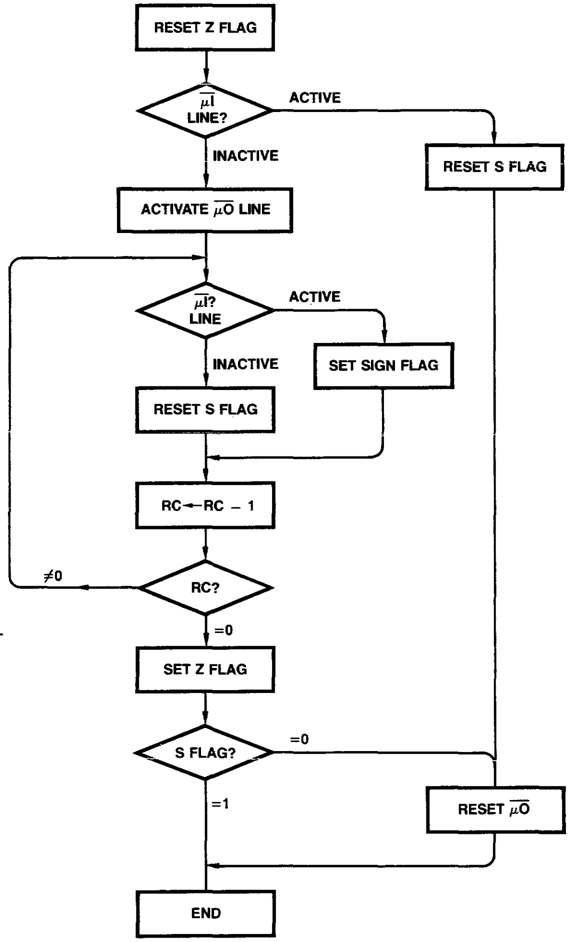
Figure 3-12. Multimicro Interface.

The internal CPU protocol that implements the request/accept behaviour makes the assumption that the interface described above is present in the system. The resource request is made with the MREQ instruction. The function of this instruction is shown in Figure 3.13. The instruction sets program flags dependent upon the outcome of the request. Initially, the $\overline{\mu I}$ line is tested. Since the $\overline{\mu 0}$ line is HIGH, $\overline{\mu I}$ reflects the state of the $\overline{\mu ST}$ daisy chain line. If $\overline{\mu I}$ is LOW, the resource is currently busy and the instruction is aborted. If $\overline{\mu I}$ is HIGH, the resource is not currently being used and $\overline{\mu 0}$ is driven LOW initiating the request. If $\overline{\mu 0}$ is LOW, $\overline{\mu I}$ now reflects the state of the $\overline{\mu AI}$ line.

The CPU enters a loop in which the $\overline{\mu I}$ line is repeatedly tested. It remains in the loop until a designated general register, which is being decremented reaches zero. This delay permits the daisy chain to settle following the resource request. During this time any multiple request conflicts will be resolved. The necessary delay can be calculated, for a given position in the chain by the number of gates in the $\overline{\mu AI}/\overline{\mu AO}$ chain. This delay is then implemented by programming the general register which is decremented at one seventh the clock rate.

If the $\overline{\mu AI}$ line is not LOW when it is tested for the last time in the loop, the CPU terminates the request, having failed to gain the resource. This result occurs when multiple requests occur and a lower priority requestor fails to gain the resource. The setting of the S and Z flags following the instruction indicates the outcome of the request.

Other instructions enable the manipulation and testing of $\overline{\mu 0}$ and $\overline{\mu I}$. The MSET instruction unconditionally sets $\overline{\mu 0}$, and the MBIT instruction its $\overline{\mu I}$ and sets program flags. The MRES instruction unconditionally resets the $\overline{\mu 0}$ line and is used by a CPU to signify that it has finished with the resource. Should the multi-micro facilities not be required, the MSET, MBIT and MRES instructions enable the $\overline{\mu I}$ and $\overline{\mu 0}$ lines to be used as a 1 bit dedicated I/O port.



FLAGS

S	Z	
0	0	RESOURCE NOT AVAILABLE
0	1	RESOURCE NOT AVAILABLE (NO ACKNOWLEDGE)
1	1	RESOURCE AVAILABLE & ACKNOWLEDGED

Figure 3-13. Multimicro Request Protocol.

Interface Examples

The preceeding sections of this chapter described the principles of CPU input output operation and the characteristics of the I/O transaction which enable efficient transfers between the CPU and a peripheral device. This efficiency is augmented by a set of block I/O instructions. The I/O characteristics also facilitate the connection of a wide spectrum of peripheral devices. Since the I/O instructions can transfer byte or word operands, both 8 and 16 bit peripherals can be interfaced to the CPU. The remainder of this chapter is devoted to typical peripheral interfaces that are commonly needed. Implementations will be discussed for the connection of parallel I/O, serial I/O, counter/timers, DMA and interrupt controllers.

Am9551 Interface

The Am9551 serial interface chip can be connected to the CPU at two levels. At the programmed I/O level, the Am9551 can be polled by the CPU using regular I/O instructions. However, the Am9551 can also be connected to the AmZ8000 system at the interrupt level. Under these circumstances, the CPU need not poll the Am9551 but will be interrupted when the latter requires service from the CPU. The interface is shown in Figure 3.14 together with the logic required to perform the interrupts.

The programmed I/O connection is implemented for a minimum system. This need not be the case in general, but for the purpose of this publication is illustrative.

A linear address decoding scheme is used which avoids the decoding of the port address. Instead the sense of individual address bits is used to select the device. The CPU address/data bus is split into separate buses using an AmZ8104 Octal Bus Tranceiver to buffer the lower half of the data bus and two AmZ8173 Octal 3-state Latches to hold the address valid for the whole I/O transaction. In this system the buffer is permanently enabled by grounding the output enable. The address latches are strobed with the CPU $\overline{AS}$ inverted and the latch outputs are permanently enabled by grounding the output enable. Thus an address is captured by the latches on the rising edge of $\overline{AS}$, and remains valid until the next time $\overline{AS}$ goes LOW. The transmit/receive input to the data buffer is controlled synchronously using the R/$\overline{W}$ line 'anded' with $\overline{DS}$. Thus the buffers are only able to drive the CPU address/data bus during the latter half of the read transaction and drive data out from the CPU at all other times. The data buffer control ($\overline{RD}$) is also used as the read command for the Am9551. The CPU R/$\overline{W}$ line and $\overline{DS}$ are also used to generate the write command to the Am9551 ($\overline{WR}$). Since a linear addressing scheme is being adopted, the Chip Select ($\overline{CS}$) line of the Am9551 can be driven from one bit of the I/O address. In fact, two bits are used to select either the command or data port of the Am9551. Since the device is driven by the lower half of the CPU data bus, the I/O address should be odd to ensure correct CPU access. The two port addresses are 0005H for data and 0007H for commands. The Am9551 chip select signal is generated conditionally on an I/O access which is decoded from the CPU $\overline{ST1}$-$\overline{ST3}$ lines using one half

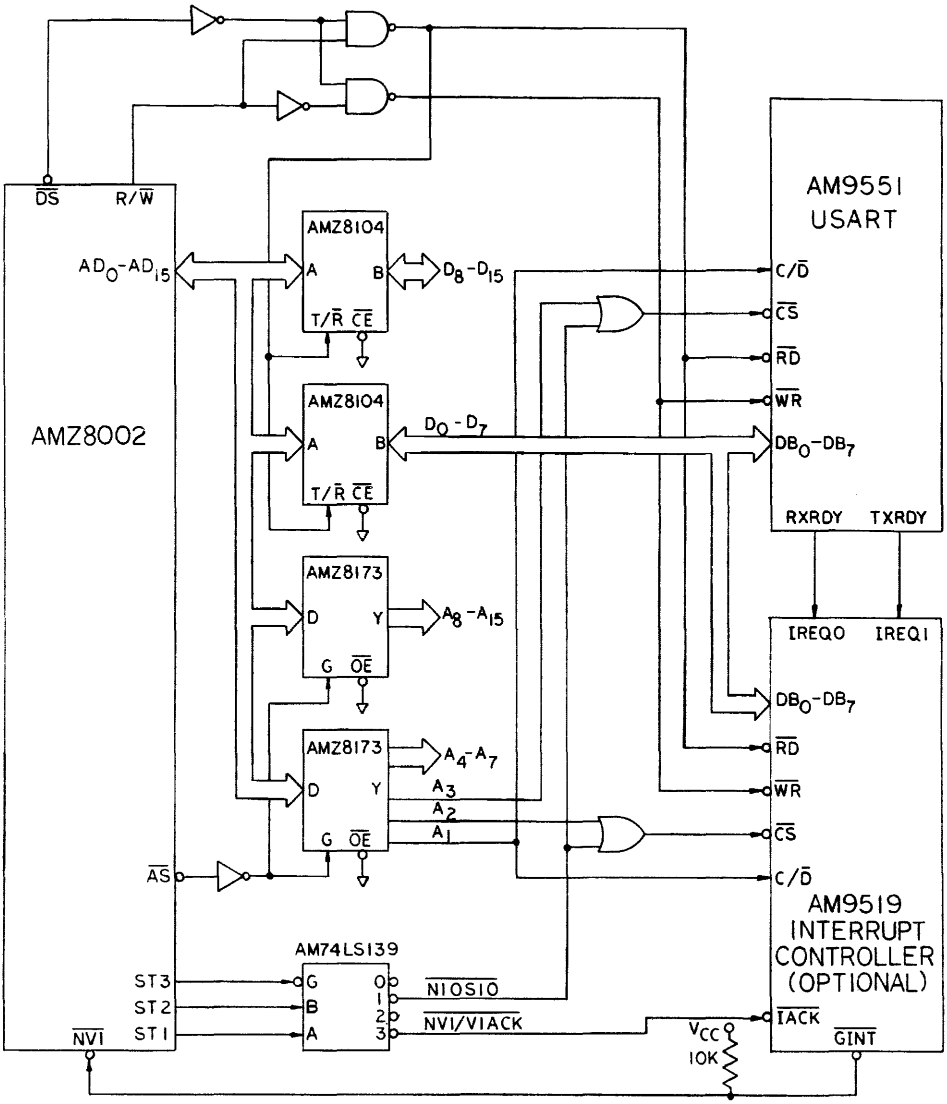


Figure 3-14. AmZ8000 to Am9551 Interface.

of a 74LS139 2 to 4 decoder. Since the least significant status line STØ is unused, only a partial decode is available. For example, no distinction can be made between an internal operation and a refresh cycle. In some small systems, however, this and the other compromises shown may be tolerable. The advantage of this implementation is that the other half of the decoder package is available for handling the memory transaction decode, for use in memory interfaces.

The programmed I/O connection described above puts a requirement for polling on the CPU. If an interrupt connection between the Am9551 and the AmZ8000 system is more desirable then a number of options are available. In the most general case, the transmitter ready (T_XRdy) and receiver ready (R_XRdy) lines from the Am9551 must be connected to one or more CPU interrupt inputs. This may be achieved as shown in Figure 3.14 by adding an Am9519 interrupt controller. The Am9519 will enable prioritization of the R_XRdy and T_XRdy signals from the Am9551 as well as supply vectors following an interrupt. The interface between the Am9519 and CPU is described in a later section.

In certain circumstances the Am9519 could be eliminated and replaced with a direct connection to the CPU. Now any simultaneous T_XRdy and R_XRdy interrupts will have to be resolved by the CPU and this may be done in the interrupt service routine. The CPU service routine must now also guard against repetitive interrupts from the same source since the T_XRdy R_XRdy lines from the Am9551 remain asserted after the interrupt acknowledge until the read/write data is transferred between the Am9551 and the CPU. This is most easily achieved by the correct programming of the interrupt masks within the flag and control word (FCW) in the new program status area that is loaded following the interrupt.

Am9511 Interface

The Am9511 arithmetic processor is interfaced to CPU using the implementation shown in Figure 3.15. The interface allows communication between the two units at both the programmed I/O level and the interrupt level. The design also illustrates the demand wait interface discussed in Chapter 1. The CPU address/data bus is buffered by a bidirectional octal buffer (AmZ8104). The direction of data flow through the buffers is controlled by R/$\overline{W}$ and $\overline{DS}$. When the CPU is in the latter half of a read cycle ($\overline{DS}$ = LOW) the buffer drives data onto the CPU address/data bus. Otherwise the buffer is driving data into the Am9511. This implementation ensures that during a write, the write data is valid after the trailing edge of $\overline{DS}$ should it be required. The port address is latched through two AmZ8173 Octal Latches using $\overline{AS}$ inverted as the latch enable. The latches are transparent while $\overline{AS}$ is LOW and capture the valid address when $\overline{AS}$ goes HIGH. The address remains valid until $\overline{AS}$ goes LOW in the next machine cycle. The latched address passes to two AmZ8121 8-bit comparators which generate a match on port address 0011H or 0013H. Port 0013H is accessed as a command port to write a command or read device status for the Am9511 and port 0011H is accessed as a data port to read or write data from the Am9511. Since the comparator must match both addresses, address bit 1 is not included in the comparison. The match generates a chip select if the operation is a normal I/O transaction. The latter is decoded from the CPU ST0-ST3 outputs which are input to a 74LS138 decoder. The decoder also generates interrupt acknowledge signals for both the vectored and non-vectored interrupts. ($\overline{VIACK}$ and $\overline{NVIACK}$ respectively). The C/$\overline{D}$ input to the Am9511 is driven from latched address bit 1. If this bit is HIGH, the Am9511 interprets a transaction as write command or read status. If this bit is LOW, then the transaction is interpreted as a data transaction. Since both addresses are odd, data transfers will always take place on the lower half of the CPU address/data bus. This is appropriate to the data connection described above.

The Am9511 requires a $\overline{RD}$ or $\overline{WR}$ command to indicate read or write transactions respectively. As well as indicating the direction of data flow, these command lines also provide timing information. The signals are generated from the CPU $\overline{DS}$ line and the R/$\overline{W}$ line. If R/$\overline{W}$ is HIGH, indicating a read, $\overline{DS}$ enables the $\overline{RD}$ line. If R/$\overline{W}$ is LOW, indicating a write, $\overline{DS}$ enables the $\overline{WR}$ line. By clocking the Am9511 with the CPU clock (divided by 2) any

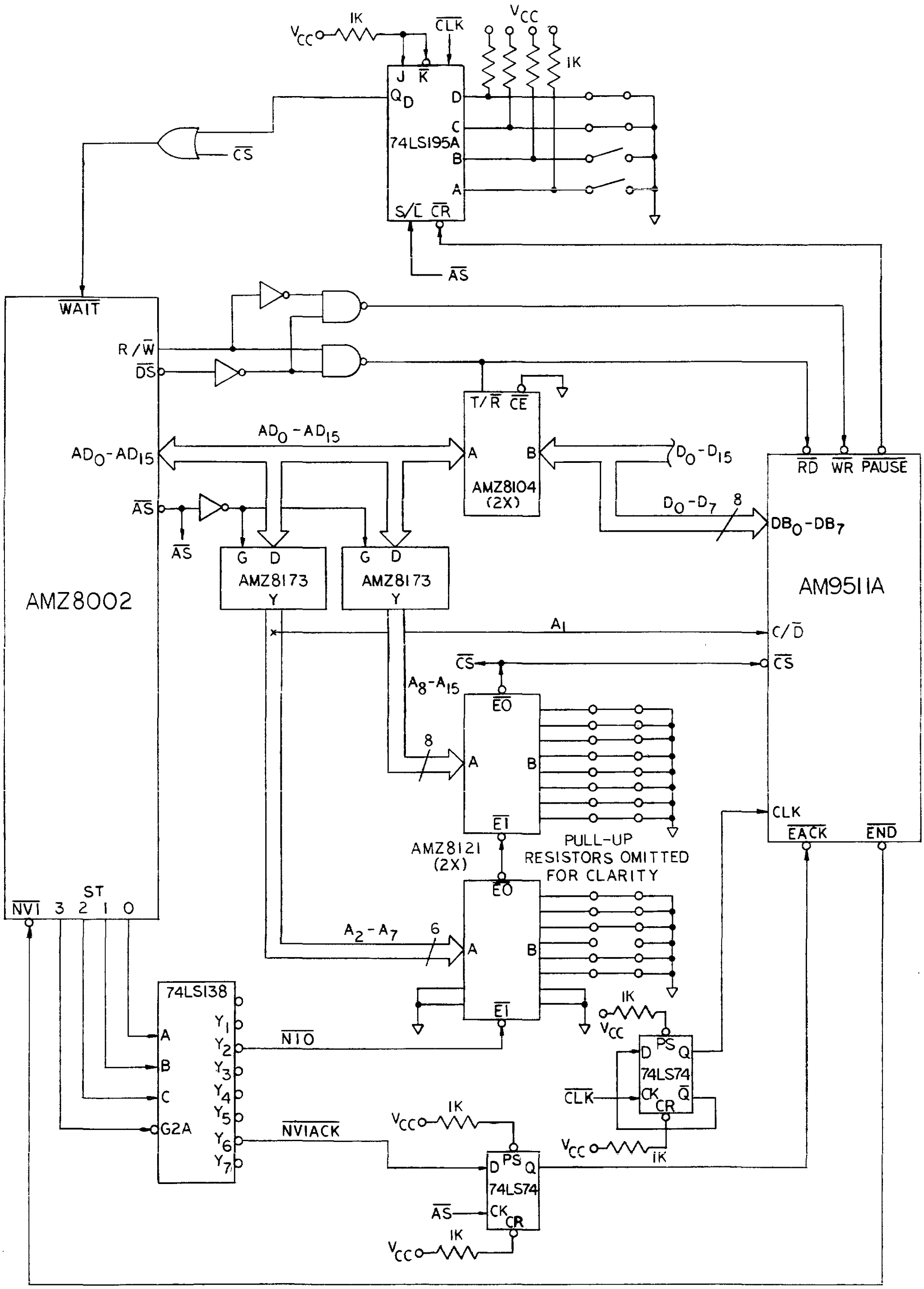


Figure 3-15. AmZ8000 to Am9511A Interface.

synchronization problems are eliminated. The division is achieved by means of a toggling 74LS74 clocked with a TTL version of the CPU clock.

The Am9511 outputs a $\overline{\text{PAUSE}}$ signal to indicate that it is not ready for the completion of a transaction. The $\overline{\text{PAUSE}}$ signal is applied to the $\overline{\text{WAIT}}$ input of the CPU. The delay involved in the Am9511 asserting the $\overline{\text{PAUSE}}$ line is too large to cause the CPU to enter the required wait state. Thus a fixed wait implementation is included in addition to the demand wait and causes the CPU to execute one conditional wait cycle, following the unconditional wait cycle associated with I/O transaction. The fixed wait is implemented using a 74LS195A shift register. The shift register is clocked by the trailing edge of the CPU clock. When $\overline{\text{AS}}$ is LOW, the register is loaded. Thereafter the register shifts on each trailing clock edge. With the shift register inputs programmed as shown, one additional wait state is inserted in the CPU timing, if $\overline{\text{CS}}$ is active for the Am9511. This additional fixed wait cycle gives the $\overline{\text{PAUSE}}$ line enough time to cause the CPU to enter extra additional wait cycles, should they be required.

This implementation includes necessary connections for the Am9511 to interrupt the CPU on completion of a task. The Am9511 $\overline{\text{END}}$ line is directly connected to the $\overline{\text{NVI}}$ line of the CPU. The $\overline{\text{NVIACK}}$ signals decoded from STØ-ST3 drives the Am9511 $\overline{\text{EACK}}$ line, enabling the CPU to acknowledge the Am9511 interrupt. A flip flop clocked by $\overline{\text{AS}}$ is included in this path to guard against spurious $\overline{\text{NVIACK}}$ pulses erroneously acknowledging the Am9511 Interrupt. The use of the Non-Vectored Interrupt is satisfactory in a system with only a few sources of interrupt. Since the interrupting device is not returning a vector, the CPU must poll each of the devices to determine the source of the interrupt. The greater the number of devices the larger the overhead spent in polling. Thus multiple devices could be interfaced to the AmZ8000 using the Am9519 interrupt controller. This is shown in a later example.

Am9519 Interface

The application involving the connection of the Am9511 to the AmZ8002 highlights the limitations of a direct interrupt connnection to the CPU. As the number of interrupting devices grows, the time spent by the CPU in determining the source of an interrupt becomes a significant overhead. The inclusion of the Am9519 Interrupt Controller in the system relieves the CPU of a large part of this task and also serves to prioritize multiple interrupts.

The implementation is shown in Figure 3.16. As described previously, AmZ8173 latches and AmZ8104 bidirectional buffers are used to generate separate address and data buses from the CPU address/data bus. The latched address is input to two AmZ8121 8-bit comparators which generate a Chip Select ($\overline{CS}$) for the Am9519 if the address is of value 0021H or 0023H and a normal I/O transaction is being executed. Latched address bit LADDR1 is not included in the comparison, but drives the C/$\overline{D}$ input to the Am9519. This defines address 0021H as being the address for data transactions and address 0023H as being the address for command/status transactions.

The lower half of the buffered data bus drives the bidirectional data bus of the Am9519. As in previous interfacing applications, this is applicable with the choice of odd port addresses, should byte I/O operations be executed. The reader will recall that in the above situation transactions take place on the lower half of the data bus. The read and write commands required by the Am9519 ($\overline{RD}$ and $\overline{WR}$ respectively) are generated with half of a 74LS139 decoder. $\overline{DS}$ is aplied to the enable input and R/$\overline{W}$ is applied to the least significant select line. The most significant select line is grounded. This is an alternative solution to the discrete gate implementation previously suggested.

The Am9519 makes a vectored interrupt request to the CPU using the Group Interrupt line ($\overline{GINT}$) which should be a LOW active output from the Am9519. $\overline{GINT}$ is reset by the CPU vectored interrupt acknowledge ($\overline{VIACK}$). The latter is decoded from the status lines and strobed into a flip-flop on the trailing edge of $\overline{AS}$. The inclusion of the flip-flop ensures that no spurious pulses from the status decode may erroneously cause an interrupt acknowledge.

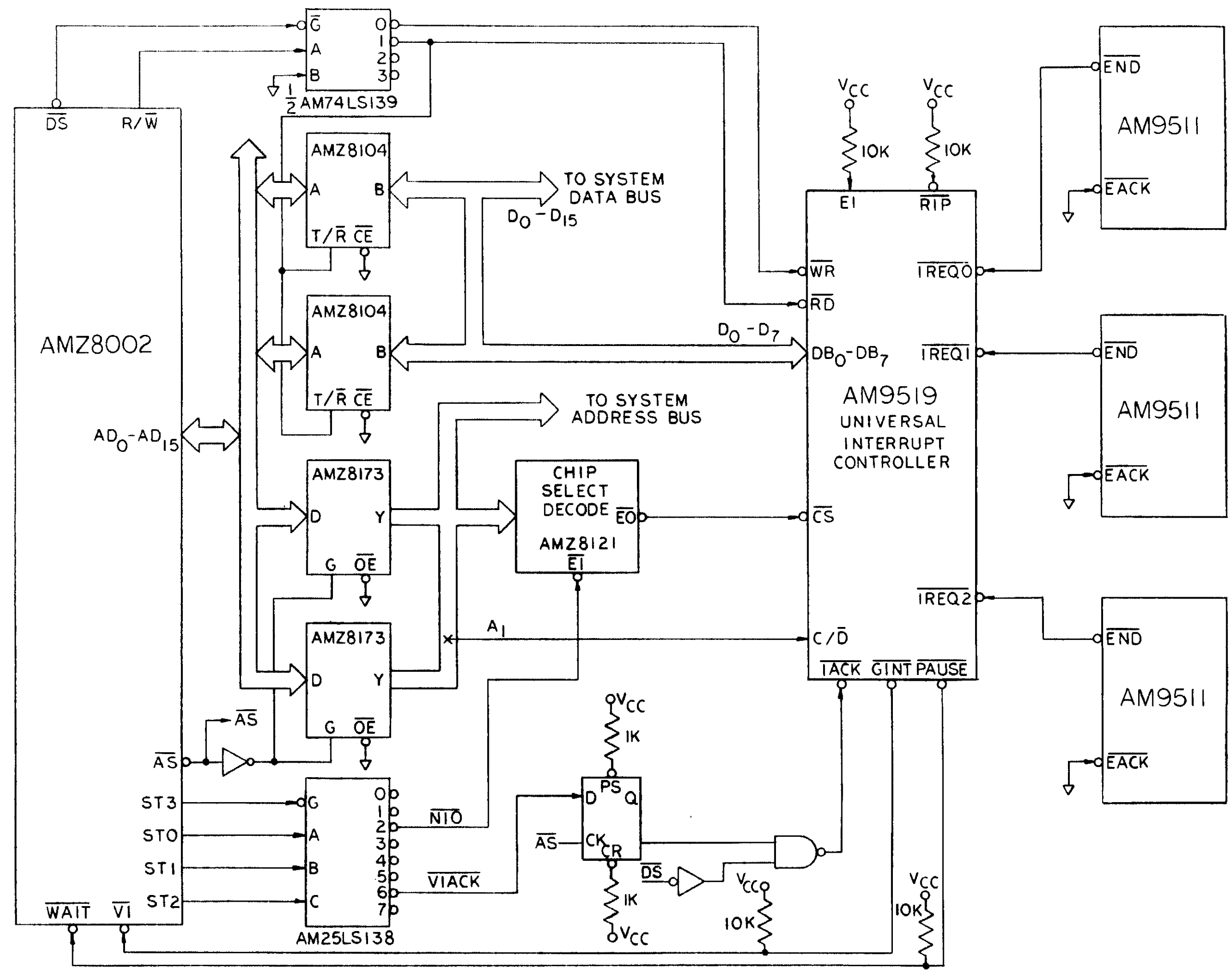


Figure 3-16. AmZ8000 to Am9519 Interface.

The Am9519 should be programmed to respond to a single interrupt acknowledge which in turn results in the transfer of 1 byte of interrupt status to the CPU. This is sufficient since the vectored interrupt mechanism in the CPU requires only 1 byte of status to form the vector. Since the Am9519 returns unique vectors for each of the 8 possible interrupts it receives from the devices, the interrupt acknowledge cycle enables the CPU to determine the source of the interrupt without any further overhead The interrupts input to the Am9519 from the devices may be levels or pulses. If levels are employed, then some form of request reset will have to be included. In the implementation shown, pulsed interrupt requests are assumed, and should be connected to the $\overline{\text{IREQ}}$ inputs of the AM9519, which should be programmed for Low Active interrupt requests. In the case of the AM 9511 device, the interrupt request will be a pulse if the $\overline{\text{EACK}}$ line is grounded. (This line was connected to the CPU acknowledge output in earlier examples.

AM9555A Interface

The AM9555A programmable I/O interface provides 24 programmable I/O pins. Its interface with the CPU is shown in fig. 3.17. The address/ data bus is divided into seperate busses by means of AM78173 octal 3-state latches driven from the CPU $\overline{AS}$ signal, and AmZ8104 octal tranceivers. This technique has been employed extensively in this chapter and therefore is not reiterated here. The Am9555A requires a chip select ($\overline{CS}$) and this is derived from two AmZ8121 comparators. Selected address bits are compared against an address programmed by selectable jumpers, on the second input of the comparator. If the jumper is present, the comparator input is pulled LOW indicating a '0'. If the jumper is absent, the comparator input is pulled HIGH by the resistor connected to +5V. The latter condition signifies a '1'. The least significant address bit (ADDRØ) must be HIGH to enable $\overline{CS}$. This requirement arises since the peripheral is connected to the lower or odd half of the data bus. During a byte transaction, this half of the data bus is accessed by means of an odd port address i.e. ADDRØ is HIGH. Address bits ADDR1 and ADDR2 are not included in the comparison but are used internally by the peripheral. The remaining 13 address bits are user definable by means of the jumper connection to enable the appropriate $\overline{CS}$.

The Am9555A $\overline{RD}$ and $\overline{WR}$ command lines are driven by a 74LS139 2 to 4 decoder. The decoder enable is driven from $\overline{DS}$ and the R/$\overline{W}$ line selects one of two outputs. The second select line is grounded. The other half of the LS139 decoder is used to decode an I/O transaction from the CPU status lines ST3-ST1. STØ is omitted from the decode, which is now an OR function of Special I/O and Normal I/O. However this is not a significant constraint in the example shown, which need not differentiate between a special and normal I/O transaction.

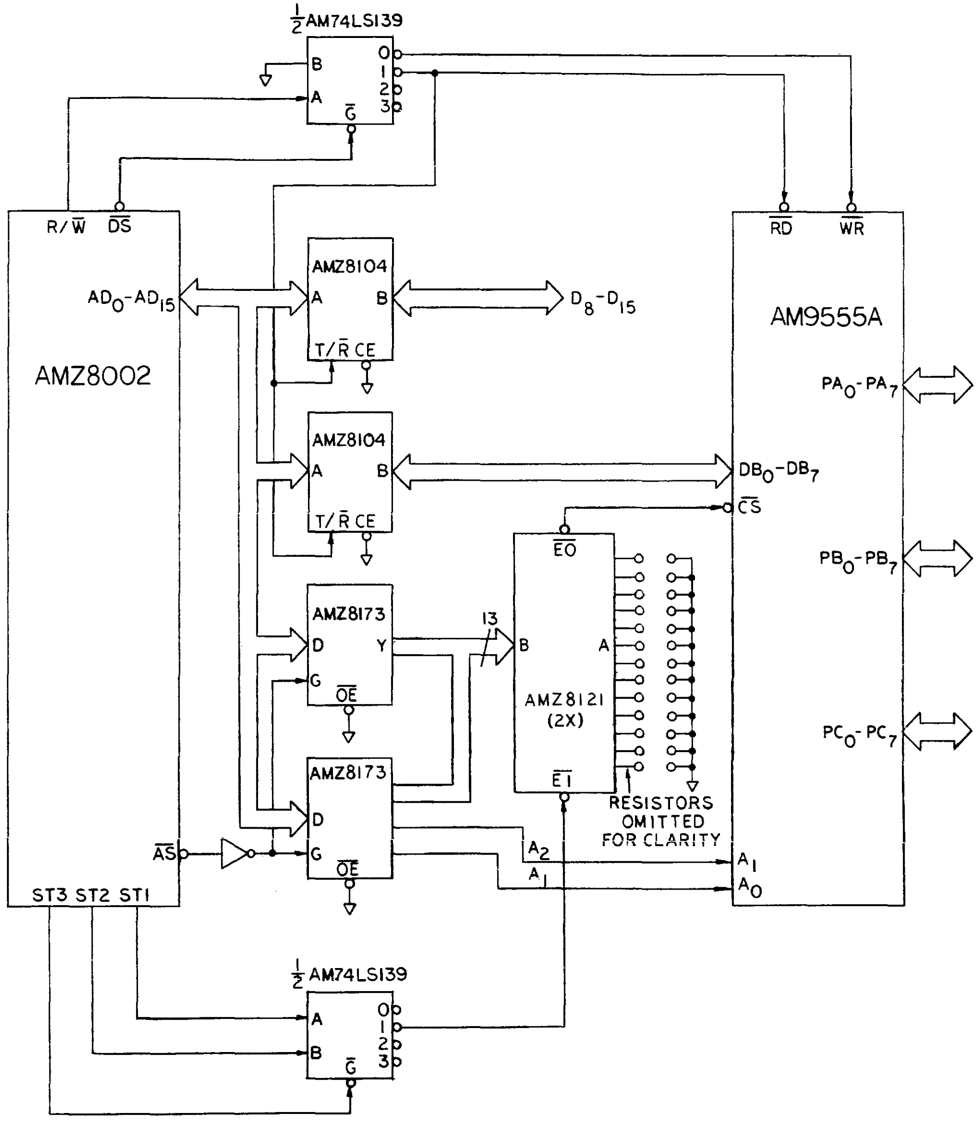


Figure 3-17. AmZ8000 to Am9555A Interface.

Am9513 Interface

The Am9513 system timing controller can be used in the AmZ8000 system to implement many timing facilities. The implementation shown in Figure 3.18 includes a real-time interrupt facility, but other tasks such as baud-rate generation, can also be delegated to the Am9513.

The device is capable of handling 16-bit transactions with the CPU, and thus the 16-bit CPU address/data bus is buffered through two AmZ8104 tranceivers before driving the 16-bit data bus of the Am9513. The address present on the CPU address/data bus during the early part of the transaction is latched in AmZ8173 3-state latches. The inverted $\overline{AS}$ signal drives the latch enable. Thus the address remains latched for the whole transactions until $\overline{AS}$ returns LOW in the following machine cycle.

The three latched address bits LADDR4-LADDR2 are input to a 74S138 3-8 decoder, which generates 8 LOW-active Chip Select signals (CSØ-CS7). CSØ when LOW, selects the Am9513. The remaining 7 chip selects may be used in other areas of the system. The least significant latched address bit (LADDRØ) when LOW enables the decoder which generates CSØ-CS7. This ensures that the 16-bit transactions between the CPU and the Am9513 are carried out with an even port address (address bit Ø = LOW) The command/data line of the peripheral is driven from LADDR1. Thus the control port is address 0002H and the data port is address 0000H. The read and write commands ($\overline{RD}$ & $\overline{WR}$) for the Am9513 are generated from R/$\overline{W}$ and $\overline{DS}$. If R/$\overline{W}$ is HIGH indicating a read, $\overline{DS}$ generates a $\overline{RD}$ command. If R/$\overline{W}$ is LOW, indicating a write, $\overline{DS}$ generates a $\overline{WR}$ command.

If the Am9513 chip is used to implement a real-time interrupt facility then some form of interrupt to the CPU must be provided from the peripheral. In this implementation one of the Am9513 OUT signals is connected through an inverter to the CPU $\overline{NVI}$ input. The Am9513 should be programmed to output a HIGH active OUT signal. Thus a HIGH or OUT causes a LOW on $\overline{NVI}$ interrupting the CPU. If vectored interrupts are required, then some form of interrupt controller such as the Am9519 should be employed in place of the direct connection between the Am9513 and the CPU $\overline{NVI}$ interrupt.

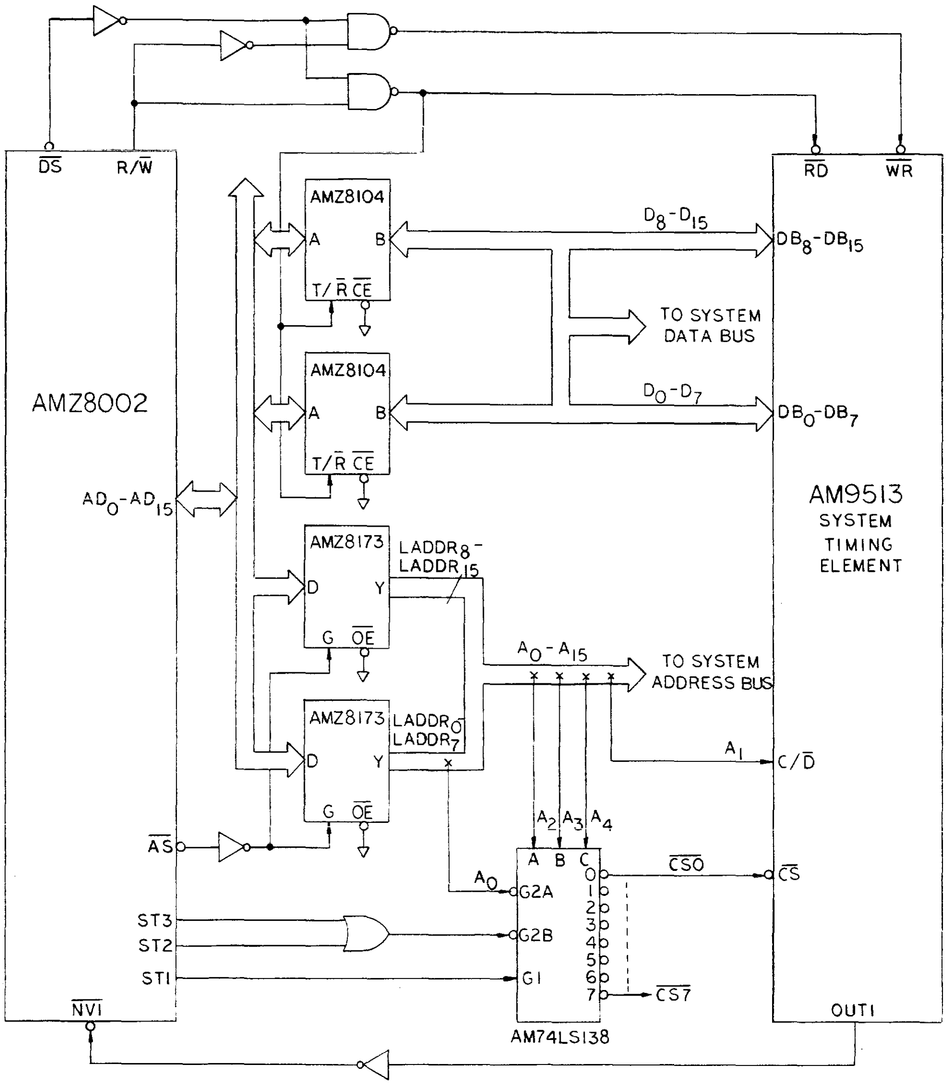


Figure 3-18. AmZ8000 to Am9513 Interface.

Am9517 Interface

The Am9517 D.M.A Controller (DMAC) can provide some useful advantages to an AmZ8000 system, both in terms of increased throughput and reduced latency in responding to peripheral requests for attention. The DMAC performs transactions between memory and I/O (or memory to memory) by gaining bus mastership from the CPU. The DMAC then controls the bus by generating the necessary control and timing signals. The interface is shown in Figure 3.19.

Bus Exchange:

The bus exchanges between the CPU and DMAC are controlled by the Hold Request (HREQ) and Hold Acknowledge (HACK) signals at the DMAC, and the Bus request ($\overline{BUSRQ}$) and BUS acknowledge ($\overline{BUSAK}$) lines from the CPU.

When the DMAC requires mastership of the bus, to perform a transaction, it drives HREQ HIGH. HREQ is inverted and drives the CPU $\overline{BUSRQ}$ line. Some time later, the CPU indicates that is has given away bus mastership and driven its bus control signals into the high impedance state by driving its $\overline{BUSAK}$ output Low. $\overline{BUSAK}$ is inverted and drives the DMAC HACK input HIGH. The DMAC responds to HACK going HIGH by enabling its bus control signals out of the high impedance state, and taking control of the bus.

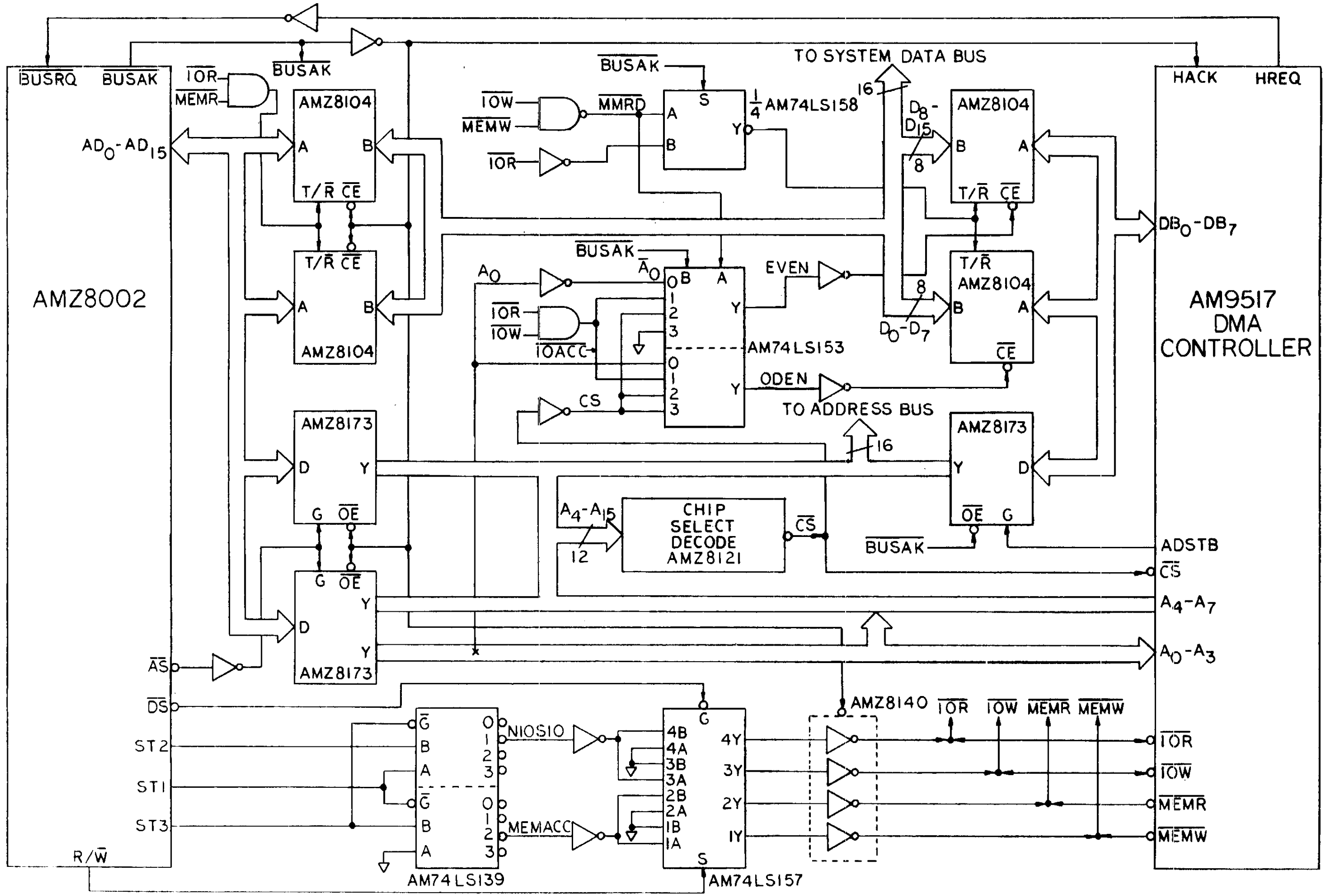


Figure 3-19. AmZ8000 to Am9517 Interface.

Address and Data busses:

The DMAC has seperate 8-bit data and 8-bit address busses. The 8-bit address bus is used for outputting the lower half of the 16-bit memory address. The 8-bit data bus, as well as handling all data into and out of the DMAC, is used for outputting the upper half of the 16-bit memory address, which is latched external to the DMAC by an AMZ8173 3-state latch. The latch is strobed by the DMAC address strobe (ADSTB) whenever the upper half of the address requires updating. Thus in a block transfer, for example, from sequential memory locations, the overhead associated with the address/data multiplexing only occurs once every 256 transfers. The 8-bit data bus is buffered by two AmZ8104 tranceivers, which fan the byte data to both halves of the 16-bit system bus during a DMAC output transaction. Similarly the buffers are used to steer either the upper or lower half of the bus onto the 8 data lines of the DMAC during a DMAC input transaction. With this configuration, memory to memory transfers, which are inherently implemented using the DMAC dataflow, can take place between any two memory byte locations. The remaining types of transfer (memory to I/O and I/O to memory) are handled without using the DMAC dataflow. In this situation, peripherals on the lower half of the bus may only access odd memory byte locations, and peripherals on the upper half of the bus may only access even memory byte locations. Thus a block transfer from a peripheral device to memory may have to be followed by a byte pack routine, if the CPU requires the byte string to occupy contiguous odd and even byte locations in memory.

The CPU shared address data bus is demultiplexed into separate address and data buses. Two AmZ8104 tranceivers drive the data bus and two AmZ8173 latches strobed with the CPU $\overline{AS}$ output hold the address valid for the duration of the transaction.

Chip Select Generation:

A DMAC Chip Select ($\overline{CS}$) is required for CPU access to the DMAC. This is generated from a comparison of the most-significant 12 address bits, since the least significant 4 address bits are input to the DMAC. The $\overline{CS}$ is only enabled when an I/O transaction is underway, decoded from the CPU status lines.

Bus Command Generation:

The DMAC carries out transactions using a set of four commands. Memory read ($\overline{\mathrm{MEMR}}$) and memory write ($\overline{\mathrm{MEMW}}$) are DMAC outputs which are generated during memory accesses. I/O read ($\overline{\mathrm{IOR}}$) and I/O write ($\overline{\mathrm{IOW}}$) may be inputs or outputs for the DMAC. If the DMAC is in control of the bus, $\overline{\mathrm{IOR}}$ and $\overline{\mathrm{IOW}}$ are DMAC outputs generated during peripheral accesses. If the CPU is in control of the bus, $\overline{\mathrm{IOR}}$ and $\overline{\mathrm{IOW}}$ are inputs to the DMAC generated by the CPU when the latter is accessing the DMAC control registers. When the DMAC has control of the bus these commands are generated directly. However, they must be decoded from the CPU control signals when it is in command of the bus.

The command signals are generated using a 74LS139 dual 2-4 decoder cascaded with a 74LS157 quad 2:1 multiplexor. The 74LS139 decodes the CPU status lines ST3-ST1 to generate $\overline{\mathrm{NIOSIO}}$ or $\overline{\mathrm{MEMACC}}$, indicating peripheral or memory access respectively. The 74LS157 multiplexor generates one of four commands, dependent upon whether $\overline{\mathrm{NIOSIO}}$ or $\overline{\mathrm{MEMACC}}$ is LOW and whether $R/\overline{W}$ is indicating a read or a write. The command that is generated is enabled by $\overline{\mathrm{DS}}$, which is applied to the 74LS157 multiplexor enable. Thus the commands generated are synchronous with $\overline{\mathrm{DS}}$. This ensures the necessary data set up and hold times are met for the peripheral and memory devices on the bus. Finally, the four command lines are buffered by an AmZ8140 3-state buffer, before being output onto the system bus.

Address and Data Bus Buffer Control:

In some circumstances, the DMAC is a responding device in the system, such as during CPU accesses. At other times the DMAC is the bus master, handling transactions between system components. The control requirements for the DMAC address and data buffers are different for both cases and thus the final stage of control generation is implemented through a 74LS153 multiplexor. The multiplexor select lines are driven from the CPU $\overline{\mathrm{BUSAK}}$ line (which defines whether the CPU or DMAC is in control of the bus) and an OR of $\overline{\mathrm{MEMW}}$ and $\overline{\mathrm{IOW}}$ which defines a write transaction.

$\overline{\text{BUSAK}}$ = HIGH (CPU Controls bus)

If the CPU has control of the bus, the DMAC data buffers on both halves of the bus are enabled if $\overline{\text{CS}}$ is LOW and $\overline{\text{MMRD}}$ is LOW. The latter signifies that neither $\overline{\text{IOW}}$ or $\overline{\text{MEMW}}$ are LOW but since $\overline{\text{CS}}$ is LOW $\overline{\text{IOR}}$ must be active. Thus the CPU can read byte data on either the upper or lower half of the bus. This requirement arises since the control locations for the DMAC must be at 16 sequential addresses. Thus the CPU may access the DMAC with both odd and even addresses and hence must be able to read data from both halves of the bus.

If the CPU is writing into the DMAC, then $\overline{\text{MMRD}}$ will be HIGH since $\overline{\text{IOW}}$ is LOW. In these circumstances the bus buffer on the lower half of the bus is enabled if $\overline{\text{CS}}$ is LOW. Thus the data for the DMAC is always written from the lower half of the bus. The transmit/receive input to the buffer is derived from a 74LS258 quad 2 input multiplexer selected by $\overline{\text{BUSAK}}$. T/$\overline{\text{R}}$ is LOW (data out of DMAC) when the CPU is performing an I/O read ($\overline{\text{IOR}}$ = LOW) and chip select ($\overline{\text{CS}}$) is LOW.

The commands generated by the CPU are enabled through an AmZ8140 buffer if $\overline{\text{BUSAK}}$ is HIGH, indicating CPU control of the bus.

$\overline{\text{BUSAK}}$ = LOW (DMAC controls the bus)

If the DMAC has control of the bus, it must be able to perform byte writes or reads on either half of the bus for memory to memory transfer. During other transfers; between memory and I/O the DMAC data flow is not utilized.

Memory to memory transfers are implemented in two transactions. A memory read followed by a memory write. Other transfers, however, are implemented in one transaction. If the read part of a memory-memory transfer is underway, $\overline{\text{MMRD}}$ will be LOW since both $\overline{\text{IOW}}$ and $\overline{\text{MEMW}}$ will be HIGH. Thus the 74LS153 outputs will be driven from the least significant address bit ADDRØ. If ADDRØ is HIGH, the odd buffer will be enabled. If ADDRØ is LOW, the even buffer will be enabled. Thus the DMA will read data off the upper or lower half of the bus, dependent upon the least significant address bit. If $\overline{\text{MMRD}}$ is HIGH, the either $\overline{\text{IOW}}$ or $\overline{\text{MEMW}}$ is LOW, and the 74LS153 outputs are driven from IOACC. If IOACC is HIGH, indicating an I/O transfer, both buffers are disabled. If IOACC is LOW signifying the write part of a memory-memory transfer than both buffers are enabled and the DMAC writes the byte data onto both halves of the bus. If the DMAC controls the bus, the address latch holding the upper 8 address bits is permanently enabled, since the $\overline{\text{OE}}$ is driven from $\overline{\text{BUSAK}}$.

The control for the CPU address and data buffers is more straight forward. The AmZ8104 data buffers are enabled if the CPU $\overline{\text{BUSAK}}$ signal is HIGH indicating that the CPU has bus control. The T/$\overline{\text{R}}$ buffer input is driven LOW (data into CPU) if an I/O or Memory read is taking place ($\overline{\text{IOR}}$ or $\overline{\text{MEMR}}$ = LOW). The address latches are strobed with the inverted $\overline{\text{AS}}$ line and the latch outputs are enabled if $\overline{\text{BUSAK}}$ is HIGH.

Peripheral Connection:

The data flow connection between the peripheral device and the AmZ8000 system remains unaltered by the presence of the DMAC. However, the peripheral must now gain the attention of the DMAC when service is required. This is achieved by means of the DREQ and DACK lines which implement a request/acknowledge handshake with the peripheral device. For more detailed information on the DMAC - peripheral interface, see AMPUB073 - "The Am9517 Multimode Direct Memory Access Controller".

SALES OFFICES AND REPRESENTATIVES

SOUTHWEST AREA
Advanced Micro Devices
9595 Wilshire Boulevard
Suite 401
Beverly Hills, California 90212
Tel (213) 278-9700
(213) 278-9701
TWX 910-490-2143

Advanced Micro Devices
1414 West Broadway Road
Suite 239
Tempe, Arizona 85282
Tel (602) 244-9511
TWX 910-950-0127

Advanced Micro Devices
1201 Dove Street
Suite 250
Newport Beach, California 92660
Tel (714) 752-6262
TWX 910-595-1575

Advanced Micro Devices
13777 No. Central Expy.
Suite 1008
Dallas, Texas 75243
Tel: (214) 234-5886
TWX: 910-867-4795

NORTHWEST AREA
Advanced Micro Devices
3350 Scott Boulevard
Suite 1002, Bldg. 10
Santa Clara, California 95051
Tel: (408) 727-1300
TWX: 910-338-0192

Advanced Micro Devices
7100 Broadway
Bldg. 6, Penthouse
Suite Q
Denver, Colorado 80221
Tel: (303) 427-3307
TWX: 910-931-2562

Advanced Micro Devices
7110 S.W. Fir Loop
Tigard, Oregon 97223
Tel (503) 620-1021
TWX 910-458-8797

MID-AMERICA AREA
Advanced Micro Devices
1111 Plaza Drive, Suite 420
Schaumburg, Illinois 60195
Tel: (312) 882-8660
TWX: 910-291-3589

Advanced Micro Devices
8009 34th Ave. S.
Bloomington, Minnesota 55420
Tel: (612) 854-6500
(612) 854-6520
TWX: 910-576-0929

Advanced Micro Devices
50 McNaughton Road
Suite 201
Columbus, Ohio 43213
Tel: (614) 457-7766
TWX: 810-339-2431

Advanced Micro Devices
33150 Schoolcraft
Livonia, Michigan 48150
Tel: (313) 425-3440
TWX: 810-242-8777

MID-ATLANTIC AREA
Advanced Micro Devices
40 Crossway Park Way
Woodbury, New York 11797
Tel: (516) 364-8020
TWX: 510-221-1819

Advanced Micro Devices
6806 Newbrook Ave.
E. Syracuse, New York 13057
Tel (315) 437-7546
TELEX: 93-7201

Advanced Micro Devices
2 Kilmer Road
Edison, New Jersey 08817
Tel: (201) 985-6800
TWX: 710-480-6260

Advanced Micro Devices
1 Gibralter Plaza, Suite 219
110 Gibralter Road
Horsham, Pennsylvania 19044
Tel: (215) 441-8210
TWX: 510-665-7572

Advanced Micro Devices
82 Washington Street
Poughkeepsie, New York 12601
Tel: (914) 471-8180

NORTHEAST AREA
Advanced Micro Devices
300 New Boston Park
Woburn, Massachusetts 10801
Tel: (617) 933-1234
TWX: 710-348-1332

SOUTHEAST AREA
Advanced Micro Devices
793 Elkridge Landing, #11N
Linthicum, Maryland 21090
Tel: (301) 796-9310

Advanced Micro Devices
1001 N.W. 62nd Street
Suite 409
Ft. Lauderdale, Florida 33309
Tel: (305) 771-6510
TWX: 510-955-9490

Advanced Micro Devices
6755 Peachtree Industrial Boulevard
Suite 104
Atlanta, Georgia 30360
Tel: (404) 449-7920
TWX: 810-766-0430

Advanced Micro Devices International Sales Offices

BELGIUM
Advanced Micro Devices
Avenue de Tervueren, 412, bte 9
B-1150 Brussels
Tel. (02) 771 9993
TELEX 61028

FRANCE
Advanced Micro Devices, S.A.
Silic 314, Immeuble Helsinki
74, Rue D'Arcueil
94588 Rungis Cedex
Tel (1) 686.91.86
TELEX: 202053

Advanced Micro Devices, S.A.
27 Blvd. General Vautrin
06400 Cannes, France
Tel: (093) 48.59.75
TELEX: 470966

GERMANY
Advanced Micro Devices
Mikro-Elektronik GmbH
Rosenheimer Str. 139
D-8000 Muenchen 80
Tel: (089) 401976
TELEX: 0-523883

JAPAN
Advanced Micro Devices, K.K.
Dai-San Hoya Building
1-8-17, Kamitakaido
Suginami-ku, Tokyo 168
Tel: (03) 329-2751
TELEX: 2324064

UNITED KINGDOM
Advanced Micro Devices (U.K.) Ltd
16, Grosvenor Place
London SW1X 7HH
Tel. (01) 235-6388
TELEX: 886833

International Sales Representatives and Distributors

ARGENTINA
Thiko, S.A. Electronica
Peru 653
1068 Buenos Aires
Argentina
Tel: 30-4132
33-4870
TELEX: 17825 SEMISAR

Koti International Corporation
1660 Rollins Road
Burlingame, California 94010
Tel (415) 692-3200
TELEX 331491

AUSTRALIA
A.J. Ferguson Pty. Ltd.
44 Prospect Rd
Prospect, S.A. 5082
Tel. (8) 269-1244
TELEX 82635

Instant Component Service
248 Wickham Road
Moorabbin, Victoria 3189
Tel 959566
TELEX AA30877

Instant Component Service
147 Ward Street
North Adelaide S.A. 5006
Tel: 267-2393
TELEX: 88095

Instant Component Service
343 Montague Road
Westend, Brisbane 4104
Queensland
Tel (07) 446667
TELEX AA43025

Instant Component Service
16 Gertrude Street
Arncliffe N.S.W. 2205
Tel 597-1444
TELEX: 26304

R and D Electronics
257 Burwood Highway
P.O. Box 206
Burwood, Victoria 3125
Tel (03) 288-8232 62
TELEX AA33288

R and D Electronics
P.O. Box 57
Crows Nest N.S.W. 2065
Sydney
Tel 439-5488
TELEX (790) 25468

AUSTRIA
Elbatex Ges.m.b.H.
Endresstrasse 54
A-1238 Wien
Tel (0222) 88 56 11
TELEX 0047-133128

BELGIUM
MCA Tronix
62 Route Du Condroz
B-4200 Ougree
Tel: 041-362780 362795
TELEX: 42052

BRAZIL
Icotron S.A.
Ind. de Componentes Electrónicos
05110-Av. Mutinga, 3650-6.0 Andar
Pirituba São Paulo
Caixa Postal 1375
Tel: (011) 261-0211
TELEX (011) 22274

DENMARK
Advanced Electronic of Denmark ApS
Godthabsvej 7
DK-2000 Copenhagen F
Tel (1) 19 44 33
TELEX 224 31

FINLAND
Komdel Oy
Box 32
02211 Espoo 21
Tel (0) 88 50 11
TELEX: 12-1926

Shipping:
Maapallonkatu 8J
02210 Espoo 21

FRANCE
A2M
18, Avenue Dutartre
F-78150 Le-Chesnay
Tel: (1) 955.32.49
TELEX: 698 376

Radio Television Francaise
73, Av. Charles De Gaulle
F-92200 Neuilly-sur-Seine
Tel. (1) 747 11 01
TELEX 611985

GERMANY
Cosmos Elektronik GmbH
Hegelstrasse 16
D-8000 Muenchen 83
Tel (089) 60 20 88
TELEX 0-522545

EBV-Elektronik
Gabriel-Max-Str 72
D-8000 Muenchen 90
Tel (089) 64 40 55
TELEX 0-524535

EBV-Elektronik
Oststr. 129
D-4000 Duesseldorf
Tel: (0211) 84846
TELEX: 0-8587267

EBV-Elektronik
In der Meineworth
D-3006 Burgwedel 1
Tel: (05139) 45 70
TLX: 0-923694

EBV-Elektronik
Myliusstr. 54
D-6000 Frankfurt 1
Tel: (0611) 72 04 16
TELEX: 0-413590

EBV-Elektronik
Alexanderstr. 42
D-7000 Stuttgart 1
Tel: (0711) 24 74 81
TELEX: 0-722271

Elbatex GmbH
Cäcilienstrasse 24
D-7100 Heilbronn
Tel: (07131) 89001
TELEX: 0-728362

Nordelektronik Vertriebs GmbH
Bahnhofstr. 14
D-2301 Kiel-Raisdorf
Tel: (04307) 54 83

Nordelektronik Vertriebs GmbH
Harksheiderweg 238-240
D-2085 Quickborn
Tel: (04106) 40 31
TELEX: 0-214299

MEV-Mikro Elektronik Vertrieb GmbH
Muenchnerstrasse 16A
D-8021 Deining
Tel: (08170) 7289
TELEX: 0-527828

HOLLAND
Arcobel BV
Van Almondestraat 6
P.O. Box 344
Oss
Tel: (04120) 24200
(04120) 27574
TELEX: 50835

HONG KONG
Ace Enterprises
Suite 1212
363 Nathan Road
Kowloon
Tel: 3-302925 27

INDIA
SRI RAM Associates
778 Blue Sage Drive
Sunnyvale, CA 94086
Tel: (408) 737-2692
TELEX: 348369

Hindustan Semiconductor
24 Sayed Abdulla
Brelvi Road
Bombay
Tel: 818105/812772
TELEX: 011-2726 APAR IN BOMBAY

ISRAEL
Talviton Electronics
P.O. Box 21104
9, Biltmor Street
Tel-Aviv
Tel: 444572
TELEX: VITKO 33400

ITALY
AMD-Elettronica, S.R.L.
Via Pascoli, 70-4
Ground Floor
I-20133 Milan
Tel: (02) 2364284
TELEX: AMD Elettronica
Via Pascoli 7Q-TF2364284
via Milan P.O. TELEX 311250 PPMI

Indelco, S.R.L. - Rome
Via C. Colombo, 134
I-00147 Rome
Tel: (06) 5140722
TELEX: 611517

Indelco, S.R.L. - Milan
Via S. Simpliciano, 2
I-20121 Milan
Tel: (02) 862963

JAPAN
Advanced Technology Corporation
of Japan
Tashi Bldg., 3rd Floor
No. 8, Minami Motomachi
Shinjuku-ku, Tokyo 160
Tel. (03) 265-9416

Dainichi Electronics
Kohraku Building
1-8, 1-Chome, Koraku
Bunkyo-ku, Tokyo
Tel. (03) 813-6876

Dainichi Electronics
Kintetsu-Takama Building
38-3 Takama-cho
Narashi 630

ISI Ltd.
8-3, 4-Chome, Lidabashi
Chiyoda-ku, Tokyo 102
Tel: (03) 264-3301

Kanematsu-Denshi K.K.
Takanawa Bldg., 2nd Floor
19-26, 3-Chome, Takanawa
Minatoku, Tokyo 108

Microtek, Inc.
Naito Building
7-2-8 Nishishinjuku
Shinjuku-ku, Tokyo 160
Tel: (03) 363-2317
TELEX: J26497

NORWAY
A/S Kjell Bakke
Nygaten 48
P.O. Box 143
N-2011 Stroemmen
Tel: (02) 711-872
(02) 715-330
TELEX: 19407

SOUTH AFRICA
South Continental Devices (Pty.) Ltd.
Suite 516, 5th Floor
Randover House
Cor. Hendrik Verwoerd
Dover Rd., Ranburg, Tvl.
Mail Address: P.O. Box 56420
Pinegowrie 2123
Tel: 48-7125
TELEX: 83324

SOUTH AMERICA
Intectra
2349 Charleston Road
Mountain View, CA 94043
Tel: (415) 967-8818/25
TELEX: 345 545

SPAIN
Regula S.A.
Avda. de Ramon y Cajal, 5
Madrid-16
Tel: 459 33 00-04/08
TELEX: 42 207

SWEDEN
Svensk Teleindustri AB
Box 5024
S-162 05 Vällingby
Tel: (08) 890435
TELEX: 13033

SWITZERLAND
Kurt Hirt AG
Thurgauerstr. 74
CH-8050 Zuerich
Tel: (00411) 512121
TELEX: 0045-53461

TAIWAN
Multitech America Co.
19046 Bonnet Way
Saratoga, CA 95070
Tel: (408) 867-6201

Multitech International Corp.
2nd Floor
977 Min Shen E. Road
Taipei, 105, R.O.C.
Tel: 768-1232
CABLE: MULTIIC

UNITED KINGDOM
Candy Electronic Components
Eden House
32 Well Road
Maidstone, Kent ME14 1XL
Tel: (0622) 54051
TELEX: 965633

Cramer Components Ltd
Hawke House
Green Street
Sunbury-on-Thames
Middlesex TW16 6RA
Tel: (01) 979-7799
TELEX: 923592

Eurosem International Ltd.
Haywood House
64 High Street
Pinner, Middlesex, HA5 5QA
Tel: (01) 868-0029
TELEX: 24506

ITT Electronic Services
Edinburgh Way
Harlow, Essex CM20 2DF
Tel: Harlow (0279) 26777
TELEX: 81146

Memec, Ltd.
Thame Park Industrial Estate
Thame
Oxon OX9 3RS
Tel: Thame (084) 421-3146
TELEX: 837508

Quarndon Electronics
Slack Lane
Derby DE3 3ED
Tel: Derby (0332) 32651
TELEX: 37163

U.S. and Canadian Sales Representatives

ALABAMA
Electronic Manufacturers Agents
2309 Starmount Circle, S.W.
Huntsville, Alabama 35801
Tel (205) 533-6440
TWX 810-726-2110

CALIFORNIA
(Northern)
I[2] Incorporated
3350 Scott Boulevard
Suite 1001, Bldg. 10
Santa Clara, California 95050
Tel (408) 988-3400
TWX 910-338-0192

(Southern)
Bestronics Inc
7827 Convoy Court
Suite 407
San Diego, California 92111
Tel (714) 278-2150
TWX 910-335-1267

CANADA (Eastern)
Vitel Electronics
3300 Cote Vertu, Suite 203
St. Laurent, Quebec
Canada H4R 2B7
Tel (514) 331-7393
TWX 610-421-3124
TLX 05-821762

Vitel Electronics
1 Vulcan St., Suite 203
Rexdale, Ontario,
Canada M9W 1L3
Tel (416) 675-2977
TWX 610-491-3728

Shipping
Vitel Electronics
84 Main Street
Champlain, New York 12919

CANADA (Western)
Venture Electronics
P.O. Box 3034
Bellevue, Washington 98009
Tel (206) 454-4594
TLX 32-8951

Shipping
1645 Rambling Lane
Bellevue, Washington 98004

COLORADO
R[2] Marketing
10018 No. Regency Place
P.O. Box 554
Parker, Colorado 80134
Tel (303) 841-5822

CONNECTICUT
Scientific Components
1185 South Main Street
Cheshire, Connecticut 06410
Tel (203) 272-2160

FLORIDA
Conley & Associates, Inc
P.O. Box 309
235 South Central Ave
Oviedo, Florida 32765
Tel (305) 365-3283
TWX 810-856-3520

Conley & Associates, Inc
1612 N.W. Second Ave
P.O. Box 700
Boca Raton, Florida 33432
Tel (305) 395-6108
TWX 510-953-7548

Conley & Associates, Inc
7515 North Armenia Avenue
Tampa, Florida 33604
Tel (813) 933-1759

GEORGIA
Electronic Manufacturers Agents
2800 Forest Vale Lane
Suite VI
Norcross, Georgia 30093
Tel (404) 448-2921

Electronic Manufacturers Agents
6755 Peachtree Industrial Blvd., N.E.
Suite 103
Atlanta, Georgia 30360
Tel (404) 449-9430
TWX 810-766-0430

ILLINOIS
Oasis Sales, Inc.
1101 Towne Road
Elk Grove Village, Illinois 60007
Tel (312) 640-1850
TWX 910-222-1775

INDIANA
C-S Electronic Sales, Inc.
2122-A Miami Street
South Bend, Indiana 46613
Tel (219) 291-6258
TWX 810-299-2535

C-S Electronic Sales, Inc.
1157-B South Jackson
Frankfort, Indiana 46041
Tel (317) 659-1874

IOWA
Lorenz Sales, Inc
5270 No. Park Pl. N.E.
Cedar Rapids, Iowa 52402
Tel (319) 393-6912

KANSAS
Kebco Manufacturers
7070 West 107th Street
Suite 160
Overland Park, Kansas 66212
Tel: (913) 649-1051
TWX 910-749-4077

Kebco Manufacturers
9813 England
Overland Park, Kansas 66211

MARYLAND
Burgin-Kreh Associates, Inc
6100 Baltimore National Pike
Baltimore, Maryland 21228
Tel (301) 788-5200
TWX 710-862-1450

MICHIGAN
S.A.I. Marketing Corp
P.O. Box N
Brighton, Michigan 48116
Tel (313) 227-1786
TWX 810-242-1518

Shipping:
First Federal Bank Building
Suite 109
9880 E. Grand River Avenue
Brighton, Michigan 48116

MISSOURI
Kebco Manufacturers
75 Worthington Drive, Ste. 10
Maryland Heights, Missouri 63043
Tel (314) 576-4111
TWX 910-764-0826

NEBRASKA
Lorenz Sales
2809 Garfield Avenue
Lincoln, Nebraska 68502
Tel (402) 475-4660

NEW MEXICO
The Thorson Company
1101 Cardenas, N.E.
Suite 109
Albuquerque, New Mexico 87110
Tel (505) 265-5655
TWX 910-989-1174

NEW YORK
Nycom, Inc.
10 Adler Drive
East Syracuse, New York 13057
Tel (315) 437-8343
TWX 710-541-1506

NORTH CAROLINA
Burgin-Kreh Associates, Inc
P.O. Box 19510
Raleigh, North Carolina 27609
Tel (919) 781-1100
TWX 510-928-0540

Shipping
3901 Barrett Drive
Raleigh, North Carolina 27609

OHIO
Dolfuss-Root & Co.
13477 Prospect Road
Strongsville, Ohio 44136
Tel (216) 238-0300
TWX 810-427-9148

Dolfuss-Root & Co.
354 Silvertree Lane
Centerville, Ohio 45459
Tel (513) 433-6776

PENNSYLVANIA
(Western)
Bacon Electronic Sales
115 South High Street
Waterford, Pennsylvania 16441
Tel (814) 796-2381
TWX 510-699-6870

(Eastern)
GCM Associates
1014 Bethlehem Pike
Erdenheim, Pennsylvania 19118
Tel (215) 233-4600
TWX 510-661-9170

TENNESSEE
(Western)
Burgin-Kreh Associates, Inc
350 E. Race Street
Kingston, Tennessee 37763

EMA
11305 Silver Springs Drive
Knoxville, Tennessee 37922

(Eastern)
Burgin-Kreh Associates, Inc
P.O. Box 268
12 Skyline Dr
Kingston Heights
Kingston, Tennessee 37763
Tel (615) 690-6100

TEXAS
Bonser-Philhower Sales
13777 N. Central Expressway
Suite 212
Dallas, Texas 75243
Tel: (214) 234-8438

Bonser-Philhower
3300 Chimneyrock, Suite 208
Houston, Texas 77056
Tel: (713) 783-0063

Bonser-Philhower Sales
8330 Burnett Rd
Suite 133
Austin, Texas 78758
Tel (512) 458-3569

UTAH
R[2]
940 North 400 East, Suite B
North Salt Lake, Utah 84054
Tel (801) 290-2631
TWX 910-925-5607

VIRGINIA
Burgin-Kreh Associates, Inc
P.O. Box 4254
5521 Fort Avenue
Lynchburg, Virginia 24502
Tel (804) 239-2626

WASHINGTON
Venture Electronics
P.O. Box 3034
Bellevue, Washington 98009
Tel (206) 454-4594
TELEX 32-8951

Shipping.
1645 Rambling Lane
Bellevue, Washington 98004

WISCONSIN
Oasis Sales, Inc
N 81 W 12920 Leon Road
Suite 11
Menomonee Falls, Wisconsin 53051
Tel (414) 445-6682

9-5-79

U.S. AND CANADIAN STOCKING DISTRIBUTORS

ALABAMA
Hamilton/Avnet Electronics
805 Oster Dr. N.W.
Huntsville, Alabama 35805
Tel: (205) 533-1170

Hall-Mark Electronics
4739 Commercial Drive
Huntsville, Alabama 35805
Tel: (205) 837-8700

ARIZONA
Liberty Electronics
8155 North 24th Avenue
Phoenix, Arizona 85021
Tel: (602) 249-2232

Hamilton/Avnet Electronics
2615 S. 21st Street
Phoenix, Arizona 85034
Tel: (602) 275-7851
TWX: 910-951-1535

CALIFORNIA
Avnet Electronics
350 McCormick Avenue
Irvine Industrial Complex
Costa Mesa, California 92626
Tel: (714) 754-6084
TWX: 910-595-1928

Bell Industries
1161 N. Fairoaks Avenue
Sunnyvale, California 94086
Tel: (408) 734-8570
TWX: 910-339-9378

Hamilton Electro Sales
10912 W. Washington Blvd.
Culver City, California 90230
Tel: (213) 558-2100
(714) 522-8220
TWX: 910-340-6364
910-340-7073
TELEX: 67-36-92

Hamilton/Avnet Electronics
1175 Bordeaux
Sunnyvale, California 94086
Tel: (408) 743-3300
TWX: 910-339-9332

Hamilton/Avnet Electronics
8917 Complex Drive
San Diego, California 92123
Tel: (714) 279-2421
TELEX: 69-54-15

Liberty Electronics
9525 Chesapeake Drive
San Diego, California 92123
Tel: (714) 565-9171
TWX: 910-335-1590

Schweber Electronics
17811 Gillette
Irvine, California 92714
Tel: (213) 537-4320
TWX: 910-595-1720

Liberty Electronics
124 Maryland Avenue
El Segundo, California 90545
Tel: (213) 322-8100
TWX: 910-348-7140
910-348-7111

Wyle Distribution Group/Santa Clara
3000 Bowers Avenue
Santa Clara, California 95052
Tel: (408) 727-2500
TWX: 910-338-0296
910-338-0541

Wyle Distribution Group
17981 Skypark Circle
Suite M
Irvine, California 92713
Tel: (714) 751-9850

CANADA
Hamilton/Avnet Electronics
2670 Paulus
St. Laurent, Quebec, Canada H4S1G2
Tel: (514) 331-6443
TWX: 610-421-3731

Hamilton/Avnet Electronics
6291-16 Dorman Road
Mississauga, Ontario, Canada L4V1H2
Tel: (416) 677-7432
TWX: 610-492-8867

Hamilton/Avnet Electronics
1735 Courtwood Crescent
Ottawa, Ontario, Canada K2C3J2
Tel: (613) 226-1700
TWX: 610-562-1906

RAE Industrial Electronics, Ltd.
3455 Gardner Court
Burnaby, British Columbia
Canada V5G 4J7
Tel: (604) 291-8866
TWX: 610-929-3065
TLX: 04-356533

Future Electronics
5647 Ferrier Street
Montreal, Quebec, Canada H4P2K5
Tel: (514) 731-7441
TWX: 610/421-3251
05-827789

Future Electronics
4800 Dufferin Street
Downsview, Ontario
Canada M3H 5S9
Tel: (416) 663-5563

Future Electronics
Baxter Centre
1050 Baxter Rd.
Ottawa, Ontario
Canada K2C 3P2
Tel: (613) 820-8313

COLORADO
Elmar Electronics
6777 E. 50th Avenue
Commerce City, Colorado 80022
Tel: (303) 287-9611
TWX: 910-936-0770

Hamilton/Avnet Electronics
5921 N. Broadway
Denver, Colorado 80216
Tel: (303) 534-1212
TWX: 910-931-0510

Bell Industries
8155 W. 48th Avenue
Weatridge, Colorado 80033
Tel: (303) 424-1985
TWX: 910-938-0393

CONNECTICUT
Hamilton/Avnet Electronics
643 Danbury Road
Georgetown, Connecticut 06829
Tel: (203) 762-0361

Schweber Electronics
Finance Drive
Commerce Industrial Park
Danbury, Connecticut 06810
Tel: (203) 792-3500

Arrow Electronics
295 Treadwell Street
Hamden, Connecticut 06514
Tel: (203) 248-3801
TWX: 710-465-0780

Wilshire Electronics
2554 State Street
Hamden, Connecticut 06517
Tel: (203) 281-1166
TWX: 710-465-0747

FLORIDA
Arrow Electronics
115 Palm Road N.W.
Suite 10
Palm Bay, Florida 22905
Tel: (305) 725-1480

Arrow Electronics
1001 N.W. 62nd St., Suite 402
Ft. Lauderdale, Florida 33300
Tel: (305) 776-7790

Hall-Mark Electronics
7233 Lake Ellenor Dr.
Orlando, Florida 32809
Tel: (305) 855-4020
TWX: 810-850-0183

Hall-Mark Electronics
1302 West McNabb Road
Ft. Lauderdale, Florida 33309
Tel: (305) 971-9280
TWX: 510-956-9720

Hamilton/Avnet Electronics
6800 N.W. 20th Ave.
Ft. Lauderdale, Florida 33309
Tel: (305) 971-2900

Hamilton/Avnet Electronics
3197 Tech Drive North
St. Petersburg, Florida 33702
Tel: (813) 576-3930

Pioneer/Florida
6220 S. Orange Blossom Trail
Suite 412
Orlando, Florida 32809
Tel: (305) 859-3600
TWX: 810-850-0177

GEORGIA
Arrow Electronics
3406 Oak Cliff Road
Doraville, Georgia 30340
Tel: (404) 455-4054
TWX: 810-757-4213

Hamilton/Avnet Electronics
6700 1-85
Suite 2B
Norcross, Georgia 30071
Tel: (404) 448-0800

ILLINOIS
Arrow Electronics
492 Lunt Avenue
Schaumburg, Illinois 60193
Tel: (312) 893-9420

Hall-Mark Electronics
180 Crossen Avenue
Elk Grove Village, Illinois 60007
TEL: (312) 437-8800
TWX: 910-222-2859

Hamilton/Avnet Electronics
3901 North 25th Avenue
Schiller Park, Illinois 60176
Tel: (312) 678-6310
TWX: 910-227-0060

Pioneer/Chicago
1551 Carmen Drive
Elk Grove Village, Illinois 60007
Tel: (312) 437-9680
TWX: 910-222-1834

KANSAS
Hall-Mark Electronics
11870 West 91st Street
Congleton Industrial Park
Shawnee Mission, Kansas 66214
Tel: (913) 888-4747
TWX: 510-928-1831

Hamilton/Avnet Electronics
9219 Quivira Road
Overland Park, Kansas 66215
Tel: (913) 888-8900

MARYLAND
Arrow Electronics
4801 Benson Avenue
Baltimore, Maryland 21227
Tel: (301) 247-5200

Hall-Mark Electronics
665 Amberton Drive
Baltimore, Maryland 21227
Tel: (301) 796-9300
TWX: 710-862-1942

Hamilton/Avnet Electronics
7235 Standard Drive
Hanover, Maryland 21076
Tel: (301) 796-5000
TWX: 710-862-1861
TELEX: 8-79-68

Pioneer/Washington
9100 Gaither Road
Gaithersburg, Maryland 20760
Tel: (301) 948-0710
TWX: 710-828-0545

MASSACHUSETTS
Arrow Electronics
96D Commerce Way
Woburn, Massachusetts 01801
Tel: (617) 933-8130
TWX: 510-224-6494

Hamilton/Avnet Electronics
50 Tower Office Park
Woburn, Massachusetts 01801
Tel: (617) 935-9700
TWX: 710-393-0382

Schweber Electronics
25 Wiggins Road
Bedford, Massachusetts 01730
Tel: (617) 275-5100

Wilshire Electronics
One Wilshire Road
Burlington, Massachusetts 01803
Tel: (617) 272-8200
TWX: 710-332-6359

MICHIGAN
Arrow Electronics
3921 Varsity Drive
Ann Arbor, Michigan 48104
Tel: (313) 971-8220
TWX: 810-223-6020

Hamilton/Avnet Electronics
32487 Schoolcraft
Livonia, Michigan 48150
Tel: (313) 522-4700
TWX: 810-242-8775

Pioneer/Michigan
13485 Stamford
Livonia, Michigan 48150
Tel: (313) 525-1800
TWX: 810-242-3271

MINNESOTA
Arrow Electronics
9700 Newton Avenue South
Bloomington, Minnesota 55431
Tel: (612) 888-5522

Hall-Mark Electronics
9201 Penn Avenue South
Suite 10
Bloomington, Minnesota 55431
Tel: (612) 884-9056
TWX: 910-576-3187

Hamilton/Avnet Electronics
7449 Cahill Rd.
Edina, Minnesota 55435
Tel: (612) 941-3801

MISSOURI
Hall-Mark Electronics
13789 Rider Trail
Earth City, Missouri 63045
Tel: (314) 291-5350
TWX: 910-760-0671

Hamilton/Avnet Electronics
364 Brookes Lane
Hazelwood, Missouri 63042
Tel: (314) 731-1144
TELEX: 44-23-48

NEW JERSEY
Arrow Electronics
Pleasant Valley Road
Moorestown, New Jersey 08057
Tel: (609) 235-1900

Arrow Electronics
285 Midland Ave.
Saddle Brook, New Jersey
Tel: (201) 797-5800
TWX: 710-988-2206

Hamilton/Avnet Electronics
10 Industrial Road
Fairfield, New Jersey 07006
Tel: (201) 575-3390

Hamilton/Avnet Electronics
1 Keystone Avenue
Cherry Hill, New Jersey 08003
Tel: (609) 424-0100

Schweber Electronics
18 Madison Road
Fairfield, New Jersey 07006
Tel: (201) 227-7880
TWX: 710-480-4733

Wilshire Electronics
1111 Paulison Avenue
Clifton, New Jersey 07015
Tel: (201) 340-1900
TWX: 710-989-7052

NEW MEXICO
Bell Industries
121 Elizabeth N.E.
Albuquerque, New Mexico 87123
Tel: (505) 292-2700
TWX: 910-989-0625

Hamilton/Avnet Electronics
2450 Baylor Drive S.E.
Albuquerque, New Mexico 87119
Tel: (505) 765-1500

Electronic Devices Co., Inc.
3301 Juan Tabo N.E.
Albuquerque, New Mexico 87111
Tel: (505) 293-1935

NEW YORK
Arrow Electronics
900 Broad Hollow Road
Farmingdale, New York 11735
Tel: (516) 694-6800
TWX: 510-224-6155

Hamilton/Avnet Electronics
167 Clay Road
Rochester, New York 14623
Tel: (716) 442-7820

Hamilton/Avnet Electronics
5 Hub Drive
Melville, New York 11746
Tel: (516) 454-6000
TWX: 510-224-6166

Hamilton/Avnet Electronics
6500 Joy Road
E. Syracuse, New York 13057
Tel: (315) 437-2642
TWX: 710-541-0959

Summit Distributors, Inc.
916 Main Street
Buffalo, NY 14202
Tel: (716) 884-3450
TWX: 710-522-1692

Wilshire Electronics
110 Parkway South
Hauppauge
Long Island, NY 11787
Tel: (516) 543-5599

Wilshire Electronics
1260 Scottsville Road
Rochester, New York 14623
Tel: (716) 235-7620
TWX: 510-253-5226

Wilshire Electronics
10 Hooper Road
Endwell, New York 13760
Tel: (607) 754-1570
TWX: 510-252-0194

NORTH CAROLINA
Arrow Electronics
1377-G South Park Drive
Kernersville, North Carolina 27284
Tel: (919) 996-2039

Hall-Mark Electronics
1208 Front Street, Building K
Raleigh, North Carolina 27609
Tel: (919) 832-4465
TWX: 510-928-1831

Hamilton/Avnet Electronics
2803 Industrial Drive
Raleigh, North Carolina 27609
Tel: (919) 829-8030

OHIO
Arrow Electronics
6238 Cochran
Solon, Ohio 44139
Tel: (216) 248-3990

Arrow Electronics
3100 Plainfield Road
Kettering, Ohio 45432
Tel: (513) 253-9176
TWX: 810-459-1611

Hamilton/Avnet Electronics
954 Senate Drive
Dayton, Ohio 45459
Tel: (513) 433-0610
TWX: 810-450-2531

Hamilton/Avnet
761 Beta Drive, Suite E
Cleveland, Ohio 44143
Tel: (216) 461-1400

Arrow Electronics
10 Knollcrest Drive
Reading, Ohio 45237
Tel: (513) 761-5432
TWX: 810-461-2670

Pioneer/Cleveland
4800 E. 131st Street
Cleveland, Ohio 44105
Tel: (216) 587-3600
TWX: 810-422-2211

OKLAHOMA
Hall-Mark Electronics
4846 South 83rd E. Avenue
Tulsa, Oklahoma 74145
Tel: (918) 835-8458
TWX: 910-845-2290

PENNSYLVANIA
Hall-Mark Electronics
458 Pike Road
Pike Industrial Park
Huntingdon Valley,
Pennsylvania 19006
Tel: (215) 355-7300
TWX: 510-667-1750

Schweber Electronics
101 Rock Road
Horsham, Pennsylvania 19044
Tel: (215) 441-0600

Pioneer/Pittsburgh
560 Alpha Drive
Pittsburgh, Pennsylvania 15238
Tel: (412) 782-2300
TWX: 710-795-3122

TEXAS
Hall-Mark Electronics
P.O. Box 22035
11333 Page Mill Road
Dallas, Texas 75222
Tel: (214) 234-7300
TWX: 910-867-4721

Hall-Mark Electronics
8000 Westglen
Houston, Texas 77063
Tel: (713) 781-6100
TWX: 910-881-2711

Hall-Mark Electronics
10109 McKalla Drive
Suite F
Austin, Texas 78758
Tel: (512) 837-2814
TWX: 910-874-2010

Hamilton/Avnet Electronics
4445 Sigma Road
Dallas, Texas 75240
Tel: (214) 661-8661
TELEX: 73-05-11

Hamilton/Avnet Electronics
3939 Ann Arbor Street
P.O. Box 42802
Houston, Texas 77042
Tel: (713) 780-1771

Hamilton/Avnet Electronics
10508A Boyer Blvd.
Austin, Texas 78757
Tel: (512) 837-8911

Schweber Electronics
14177 Proton Road
Dallas, Texas 75240
Tel: (214) 661-5010
TWX: 910-860-5493

Schweber Electronics
7420 Harwin Drive
Houston, Texas 77036
Tel: (713) 784-3600

UTAH
Bell Industries
3639 W. 2150 South
Salt Lake City, Utah 84120
Tel: (801) 972-6969
TWX: 910-925-5686

Hamilton/Avnet Electronics
1585 West 2100 South
Salt Lake City, Utah 84119
Tel: (801) 972-2800
TWX: 910-925-4018

WASHINGTON
Hamilton/Avnet Electronics
14212 N.E. 21st Street
Bellevue, Washington 98005
Tel: (206) 746-8750
TWX: 910-443-2449

Liberty Electronics
1750 132nd Avenue N.E.
Bellevue, Washington 98005
Tel: (206) 453-8300
TWX: 910-443-2526

WISCONSIN
Arrow Electronics
434 W. Rawson Avenue
Oak Creek, Wisconsin 53154
Tel: (414) 764-6600
TWX: 910-262-1193

Hall Mark Electronics
9657 S. 20th Street
Oak Creek, Wisconsin 53154
Tel: (414) 761-3000

Hamilton/Avnet Electronics
2975 Moorland Road
New Berlin, Wisconsin 53151
Tel: (414) 784-4510

ADVANCED MICRO DEVICES, INC.
901 Thompson Place
Sunnyvale
California 94086
(408) 732-2400
TWX: 910-339-9280
TELEX: 34-6306
TOLL FREE
(800) 538-8450

9-5-79

ADVANC
MIC
DEVICES, I
901 Thompson P
Sunny
California 94
(408) 732-2
TWX: 910-339-9
TELEX: 34-6
TOLL F
(800) 538-8
8