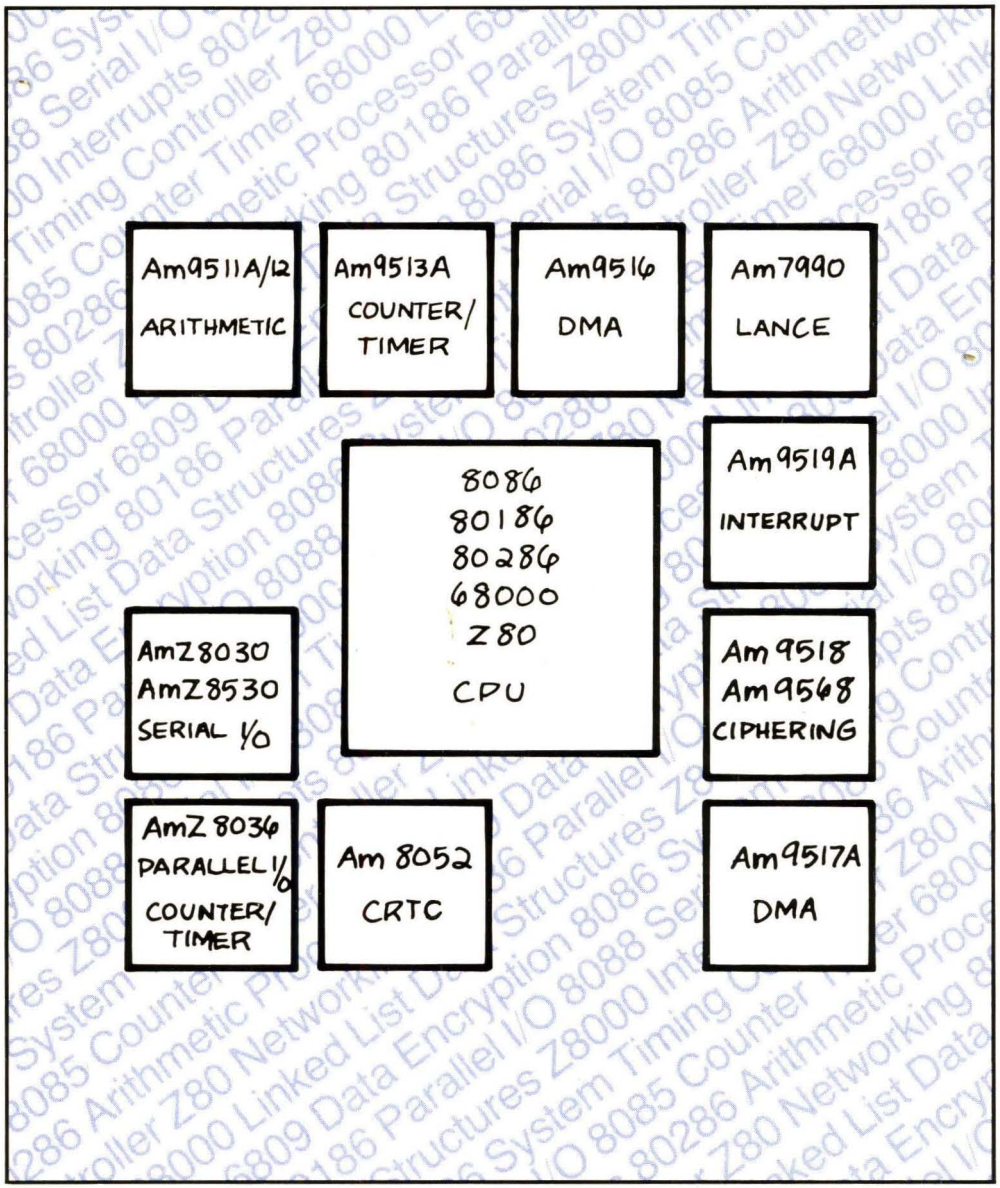
Peripheral Processor Interface Guide

Technical Manual



ADVANCED MICRO DEVICES

Advanced Micro Devices

Peripheral Processor Interface Guide Technical Manual

The International Standard of Quality guarantees a 0.05% AQL on all electrical parameters, AC and DC, over the entire operating range.

INT·STD·500

Printed in U.S.A.

TABLE OF CONTENTS

1.0 INTRODUCTION

OBJECTIVE

Supporting the CPU with modern, smart peripheral devices is the most effective way to optimize a microprocessor-based system. Working with the CPU, intelligent devices unburden the processor by performing tasks such as counting or Direct Memory Access (DMA), but at a much higher throughput than a MOS microprocessor working alone.

Indeed, in cases such as high speed counting or controlling an advanced CRT display, the job simply could not be done by using a MOS CPU alone. Furthermore, the modern breed of peripherals has broken old design barriers by performing complex tasks such as linked-list DMA, which, in the Am9516 for example, can easily be implemented for control of data information. The entire complex task can be done in an advanced peripheral device with a single instruction from the CPU.

Unfortunately, exponential increases in system performance do not come without cost. Sophisticated peripherals are more complex, more difficult to understand and often more expensive than the CPU itself. A careful cost/benefit analysis will quickly tell the prudent design engineer if a particular device is the best solution. This guide addresses the first and second considerations.

INTENDED AUDIENCE

The materials presented here assume an understanding of digital design, a working knowledge of commercial CPUs, and familiarity with the peripherals discussed.

Detailed product specifications of any AMD device discussed here are available by calling your local sales office (see inside back cover) or our Literature Department at (408) 749-2264.

WHY AMD IS PUBLISHING THIS MANUAL

Due to its very broad product line, Advanced Micro Devices is in a unique position to help designers understand CPU/peripheral interfaces. AMD not only makes the old 8080 and 8085 processors, but also the newer Z8000 micros and the state-of-the-art iAPX86 machines. More importantly, we want to make designing with our devices as easy as possible.

The Am9500 series is AMD's peripheral flagship. The Am9511A is the only general purpose floating point processor on the market. The 8087 is faster, but works as a dedicated co-processor to the 8086. The Am9513A is the most versatile counter/timer on the market. Both the Am9517A and the Am9519A are one generation beyond the 8257 and the 8259 that they respectively replace. The Am9516, a 16-bit Universal DMA Controller, is in a class by itself.

AMD is also an active participant in the Z8000 design sphere, with the AmZ80XX series of peripherals several of which have become quite popular. Both, the AmZ8030 Serial Communication Controller, an improved variation of the well-received Z80-SIO, and the AmZ8036 Counter/PIO, which combines parallel I/O with three 16-bit counters, have no equivalent in other microprocessor lines. The Z8038/8060 bidirectional FIFO buffers offer unique solutions to the problem of buffering between different systems. The AmZ8016, brother of the Am9516, represents a new level of DMA sophistication. The Am8052 CRT Controller is another device that is in a class by itself.

We have a tradition of designing advanced peripherals which are more powerful and sophisticated than the industry-standard devices they replace. Because of our position in peripheral designs, it is appropriate that we should talk about peripheral interfacing.

DESIGNER'S ROLE IN THE INTELLIGENT PERIPHERAL ENVIRONMENT

While the selection of a CPU is usually based on complex software issues, sometimes even by corporate edicts, peripheral selection is governed more by performances and "features for cost" considerations. It is the peripherals in most cases that will greatly distinguish the final product. For example, the presence of an Am8052 CRT Controller has a much greater impact on a workstation than the kind of microprocessor used.

The specification of peripherals is squarely the designer's job. While the choice of interesting devices is almost an overwhelming one, the supporting application information is usually available only for connecting a CPU to its family of peripherals. Because of this, some designers are under the impression that it is difficult to mix and match CPUs and peripherals; that is, to use Z8000 peripherals with the 8086 or 68000; or to connect 8080/8086 peripherals to the Z80, Z8000, 6800, or 68000.

Although such cross-breeding is more complex than the traditional family connection, most of the difficulties stem from the need to understand the idiosyncrasies of different manufacturers' devices and their sometimes confusing documentation. The actual hardware needed to match peripherals to a "foreign" CPU, or a CPU to "foreign" peripherals, is usually quite simple. Sometimes by incorporating a few TTL SSI/MSI packages or even a single PAL device is all that it takes.

ORGANIZATION OF THIS BOOK

This book is organized into three sections:

Section A	Discusses the general interface problem.
Section B	Interfacing today's three most common 16-bit processors: the 8086, the 68000, and the Z8000.
Section C	Interfacing with commercial 8-bit processors such as the Z80, the 8085, the 6809, the 8088, and the 8051.

In sections B and C, after a brief overview of the particular CPU, all interfaces to that microprocessor are explained. In turn, each individual micro/peripheral interface is presented by discussing :

1) The peripheral itself.
2) Any critical timing constraints peculiar to the device.
3) Any required software initialization.
4) The system's interrupt structure (if any).
5) Bus structures.
6) A complete block diagram with all interconnections.

HOW TO USE THIS BOOK

The digital design engineer should first review Section A and takes notes on some of the interfacing problems to watch out for.

For those who are designing around a particular processor with added features to the system, turn to the specific section that discusses that microprocessor. All interfaces to that CPU listed in this book will be found there.

For answers to more technical questions, please call the AMD Sale Office in your area (see inside back cover) and ask to speak to a Field Applications Engineer.

SECTION A
INTERFACING

2.0 THE INTERFACE PROBLEM

Several problems are associated with interfacing a peripheral device to a CPU when the two families are not designed to be pin to pin compatible. One major problem involves the various control signals that each chip uses. What's more, the popular MOS microprocessors differ not only in their instruction set and internal architecture, but also in their I/O interface signals and timing, and in the way they handle DMA and interrupts. Part of the pin incompatibility involves genuine signal differences or timing differences. Some pin to pin differences require nothing more than name changes.

PROBLEMS TO LOOK FOR WHEN INTERFACING

The following is a list of areas to watch out for when interfacing, and are discussed in detail:

I. TIMING

A. Setup and hold times

i. $\overline{CS}$ setup and hold to strobe.
ii. Address setup and hold to strobe.
iii. Data setup and hold to strobe (rising or falling)
Note: Strobe may be $\overline{RD}$, $\overline{WR}$, or $\overline{DS}$.

B. Pulse widths

i. $\overline{AS}$ or $\overline{ALE}$ pulse widths.
ii. $\overline{RD}$, $\overline{WR}$, $\overline{DS}$ pulse widths.
Pulse widths can usually be stretched by inserting Wait States if necessary.

C. Interrupts

Interrupt structures vary. Mixing interrupt structures is the toughest because two or more different sets of timings may need to be generated.

II. METASTABLE OPERATION

When to expect a cause of problem and the solutions.

III. BUS STRUCTURES

i. MULTIBUS
ii. iLBX
iii. iSBX Multichannel

2.1 TIMING

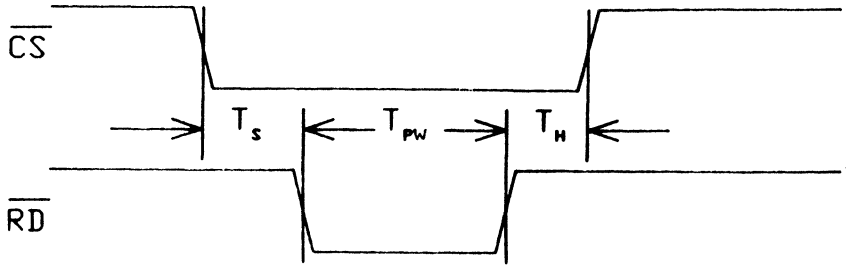
SETUP AND HOLD TIMES

If one remembers nothing else from this book, one should remember the following: *The most common interfacing error is the violation of setup and hold times!* More specifically, three areas of setup and hold time relationships need close attention, and we will discuss them after a brief overview.

Setup and hold time are the minimum time the proper signal must be on the bus before and after the strobe. Minimum setup and hold time criterion must be met for proper interface operation. When interfacing involves direct hookup, individual specification can be looked up and the worse case calculated to meet timing requirements. If the interface involves additional components, such as buffers or transceivers, the data to strobe timing relationship may change and correction by delay devices such as flip-flops or latches may be necessary to bring the system circuit within timing requirements. The following are areas that need special attention:

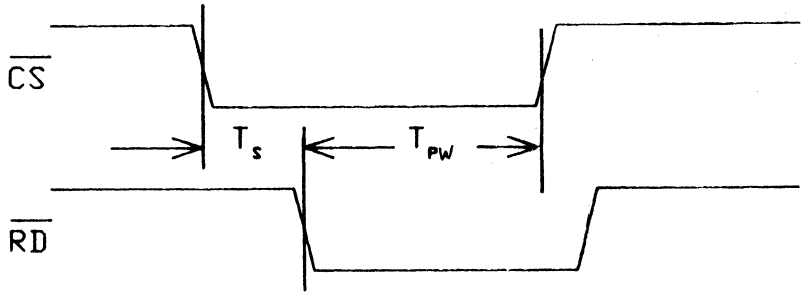
i) Chip select ($\overline{CS}$) setup and hold time to strobe. (Figure 2.1a)

Here we use $\overline{RD}$ as the strobe. T_S is the minimum $\overline{CS}$ to strobe setup time. T_H is the minimum hold time. Comparisons are made in conditions where these requirements are not met and could lead to the wrong device being selected (Figures 2.1b and 2.1c).



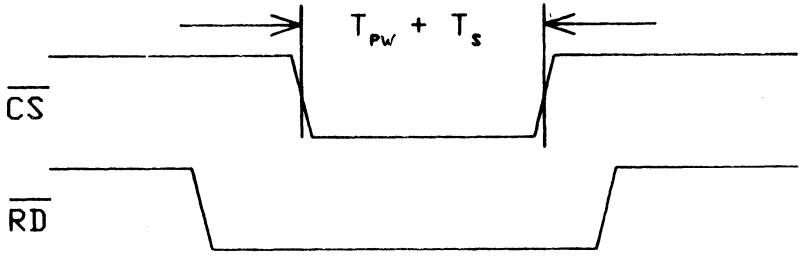
02188A-1

Figure 2.1a



02188A-2

Figure 2.1b



02188A-3

Figure 2.1c

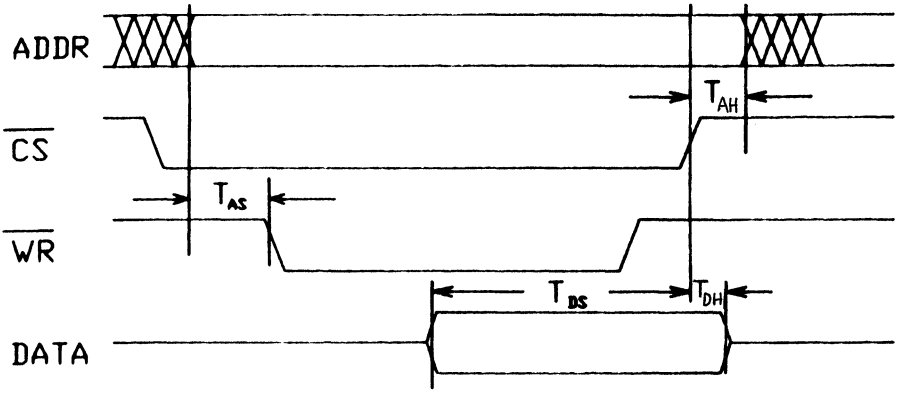
Note that it is often assumed that $\overline{CS}$ and $\overline{RD}$ or $\overline{CS}$ and $\overline{WR}$ are simply gated together. This is true for most peripherals but not in all cases. Check the manufacturer's specification. It is often possible to implement and simplify the design but manufacturers do not guarantee operation in these unusual modes.

ii) Address setup and hold times to strobe.
(Figures 2.1d and 2.1e)

The figures illustrate the relationships between strobe to address setup and hold, and $\overline{CS}$. Address setup time is measured with respect to the leading edge of the strobe. If the strobe is not sufficiently delayed, after the address is valid, the address setup time requirement will not be met. Address hold time will not be met if $\overline{CS}$ is, due to internal delay, late in going high. This can be compensated by delaying the $\overline{WR}$ strobe. Failure in meeting either requirement will result in incorrect addressing.

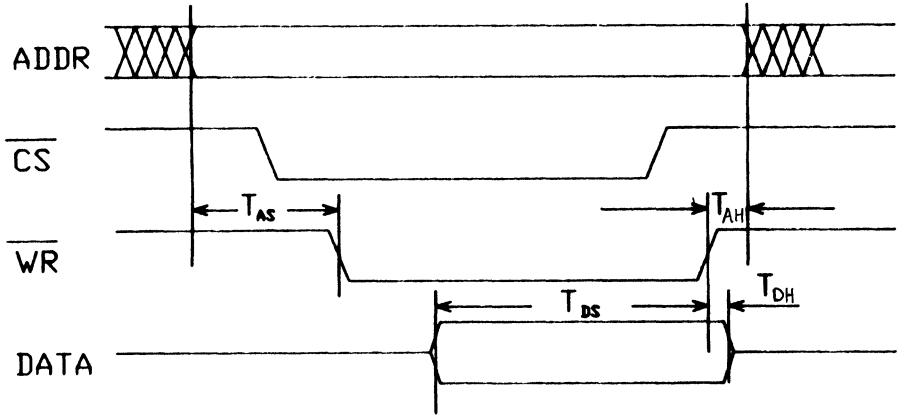
iii) Data setup and hold times to strobe ($\overline{WR}$).
(Figures 2.1d and 2.1e)

The data setup and hold times are measured with respect to the rising edge of the strobe. The data is represented here as a window. If the strobe goes high too late, data hold time will not be met, and incorrect data may be read. This may happen if the strobe is purposely delayed to meet Address setup and hold times. The data can be latched to allow a larger window so that data hold time can be met.



02188A-4

Figure 2.1d



02188A-5

Figure 2.1e

2.2 METASTABLE OPERATION

Metastable conditions occur in all flip-flops when the active clock edge samples the input at exactly the same time the input changes state. When this happens, the cross-coupled latch at the output can reach a balanced, symmetrical condition in which it will remain for some arbitrary time before returning to its proper state. This problem is faced by designers when interfacing asynchronous digital signals. Although most difficulties can be overcome somewhat easily, there is a more fundamental problem that defies a perfect solution. The following is a general overview of the metastable problem.

Latches and flip-flops are normally considered bi-stable devices, since they have two unconditionally stable operating points, either HIGH or LOW. There is, however, a third operating point when the cross-coupled arrangement is exactly balanced. This operating point is stable only if there is no noise in the system and the system is perfectly balanced. The condition is called metastable (meta=Greek for "between"). A metastable condition will last only long enough for the circuit to fall into one of the two stable operating points. This period can be many micro-seconds, or milliseconds, for devices as fast as a 74S74 flip-flop. If a flip-flop has reached the balanced, metastable condition, it may remain in this state for an undetermined time, perhaps 1000 times longer than its normal response speed. Previously, the designer of asynchronous system had only one remedy for the metastable problem; two or more synchronizer flip-flops could be cascaded. This can reduce the probability of a metastable output but will also increase the throughput delay.

WHEN TO EXPECT THIS TO CAUSE A PROBLEM?

In most digital systems, certain asynchronous events (keystrokes, incoming data, interrupts) must be synchronized to the computer clock. The textbook solution is a fast, clocked flip-flop, like the 74S74, in which the asynchronous signal is applied to the D input and clocked with the system clock. This usually results in a perfectly synchronized output.

If the data sheet specified a setup time requirement (3 ns), this means that any signal that arrives at least 3 ns before the clock edge will achieve the intended result, i.e., an **H**igh will set, an **L**ow will reset the flip-flop. Great for synchronous systems! But what happens when the asynchronous input violates this setup time requirement and changes less than 3 ns before the clock edge? Most of the time, nothing. The actual moment where the flip-flop samples the D input is somewhere in the guaranteed range, i.e., somewhere less than 3 ns before the clock. So the flip-flop makes the decision. It either senses the change on the asynchronous input and therefore changes the Q output, or it ignores the change and doesn't change the Q output. So the only thing lost is one clock cycle. Unfortunately that's not always true. If the D input changes exactly at the same moment that the flip-flop makes its decision, it might transfer exactly the amount of energy to kick the output latch into the metastable balanced condition, from which it will recover after an unpredictable delay (measured in nanoseconds, microseconds or even milliseconds).

Any latch, flip-flop or register has a "moment of truth" somewhere inside the guaranteed range of setup time where it actually makes up its mind. If the input changes at that very moment, the output is no longer synchronous. This "moment of truth" is a very short window. For TTL flip-flops, it is of the order of 10 ps; for MOS devices, it is more like 50 ps to 100 ps. For purposes of this discussion, this timing window will be called "**t**".

Here are two extreme examples. In each case there is a need to synchronize asychronous inputs that have no phase or frequency relationship with the computer clock.

* Data signal derived from a disk, roughly 6 MHz with enough frequency modulation and jitter to make it totally asynchronous to the 10 MHz computer clock.

Every time the Data Signal falls into the "window", the probability of hitting the window is **t** divided by the clock period, or even simpler: clock frequency times **t**.

M = Metastable Rate = f(D) . f(C) . t
f(D) = Device Frequency = 6 MHz . 10 MHz . 10 ps
f(C) = Clock Frequency = 600 Hz

The synchronizer goes metastable 600 times per second.

* Keyboard entry: one keystroke per second synchronized with a 100 kHz clock.

M = Metastable Rate = 1 Hz . 10^5 Hz . 10 ps = 10^{-6}Hz

The synchronizer goes metastable with a statistical probability of once per 10^6 sec., i.e., once every six weeks (assuming 5 eight-hour days/week).

"Going metastable" here means that the synchronizer output is within a mid-level or oscillation range for an unpredictable time. Most occurrences will last less than 50 ns, but may occasionally last much longer - perhaps many microseconds. This certainly can upset the timing chain.

A metastable latch or flip-flop has an unpredictable delay and will therefore change its output at a time that differs from the value obtained from the worst case timing analysis. In a slow system this usually doesn't matter, but in a fast system it can lead to a "crash".

SOLUTIONS

We cannot eliminate the basic problems but we can reduce the probability in two ways:

1) Cascade two or more synchronizer flip-flops, which is the method employed in all "metastable free" systems.
2) Use flip-flops that are less prone to metastable operation than the popular 74S74. For example, the AMD Am29821-26 registers are "metastable-hardened". They show no oscillations and only a minimal increase in output delay when hit right in the window.

Metastable operation is an inherent, so far incurable disease of all asynchronous interfaces. Once understood, the problem can be handled by reducing its probability to an acceptable level. AMD's Am29821—26 registers vastly minimize this problem. The Am29800 registers, while not totally immune to this problem, are "metastable hardened" by means of a unique circuit design that reduces both the probability and the delay of any metastable condition. An artificially induced metastable condition that failed to produce any output oscillation merely increased the clock-to-output delay by 6 ns. This is an improvement of many orders of magnitude over previously available designs.

2.3 BUS STRUCTURES

In the course of interfacing, special attention must be applied to the different standard bus systems. They are standard because they have the acceptance and support of the industry; different because each evolved to satisfy a particular function.

Bus systems are developed to increase the flexibility and expandability of the mother board. They are also developed to cut cost and hardware overheads. Since each bus structure has its unique area of operation, each must be carefully matched for proper interfacing.

The following are brief explanations on different major bus structures and areas of their involvement:

BIG MULTIBUS VS LITTLE MULTIBUS

The big Multibus architecture connects the single board computer with its CPU, memory and I/O to the outside world. The so-called little multibus is the bus between the CPU and the other components on the single board computer.

MULTIBUS

The Multibus evolved, in structure, from 8-bit to 16-bit capability; and in architecture, expansion via the iSBX bus and Multichannel bus, and iLBX bus. Expansions were necessary because a single system bus structure was no longer capable of supporting the demands of today's high-speed, high-performance VLSI microprocessor technology, and its increasingly complex configurations.

iLBX

The iLBX bus provides users the architectural solution that extends the high performance benefits of a processor's on-board local bus to off-board memory resources. Powerful iLBX system modules can be created using the bus to connect a single board computer and multiple memory cards. The iLBX bus preserves the advantages in performance and architecture of on-board memory, while allowing a wide range of memory capabilities to match application requirements.

iSBX

The iSBX bus provides users with a low cost, on-board expansion solution for Multibus single board computers. The iSBX boards allow addition in the areas of parallel I/O, serial I/O, peripheral controllers, and high-speed math, without going to the expense of additional full Multibus board. iSBX bus compatible boards enable users to buy exactly the capabilities they require for their Multibus systems, which keeps both system size and system cost at a minimum.

MULTICHANNEL

Multichannel bus provides users with a separate path for DMA I/O block transfers. The new VLSI microprocessors that process data at very high rates require the connection of numerous high-speed I/O devices to the system bus. The Multichannel bus provides for high speed block transfers of data over an 8/16-bit wide data path between peripherals and single board computer resources.

Motorola has its own equivalents to these busses. The signals and interfaces are slightly different but the functions are similar, and the objective of higher throughput is the same.

3.0 RELATED DESIGN ISSUES

This chapter shows other related areas that the designer should consider.

3.1 OSCILLATOR CONSIDERATIONS

Basically, a crystal oscillator can be thought of as a closed-loop system composed of an amplifier and feedback network containing the crystal. Amplitude of oscillation builds up to the point where non-linearities decrease the loop gain to unity. The frequency adjusts itself so that the total phase shift around the loop is 0 or 360 degrees. The crystal which has a larger reactance frequency slope, is located in the feedback network at a point where it has the maximum influence on the frequency of oscillation. A crystal oscillator is unique in that the impedance of the crystal changes so rapidly with frequency that all other circuit components can be considered to be of constant reactance. This reactance is calculated at the nominal frequency of the crystal. The frequency of oscillation will adjust itself so that the crystal presents a reactance to the circuit which will satisfy the phase requirement. If the circuit is such that a loop gain of greater than unity does not exist at the frequency at which the phase requirement can be met, oscillation will not occur.

Applying the principles of oscillator design usually is difficult because many factors play an important part. As a result, the design of crystal oscillators is often a "cut and try" procedure. A popular method, which is highly experimental, consists of giving a qualitative explanation of how the circuit works and presenting a number of typical schematic diagrams for that oscillator configuration. The configuration used depends mainly on the type of application. The use of logic gates as crystal oscillators is common in systems where the oscillator outputs must drive digital hardware. These oscillators are very simple but are prone to problems with respect to free-running and spurious oscillations. These problems, however, can be eliminated by proper design of the feedback network.

A basic, low frequency, gate oscillator is shown in Figure 3-1a. R1 is used to bias the gate near a point where it can operate as a linear amplifier and R2 is used to raise the effective output impedance of the gate. The gate provides the necessary gain and produces a phase shift of 180 degrees. The network consisting of C1, C2 and the crystal produces an additional 180 degree phase shift. The crystal looks inductive and is resonant with capacitors C1 and C2. The frequency of oscillation automatically adjusts itself so that this is true, therefore, the combination of crystal and C1 alone has a net inductive reactance at the operating frequency. Current I_1 lags voltage e_2 by 90° and voltage e_1 being developed across C1 lags current I_1 by 90°, making it 180° behind e_2. This explanation is valid only at low frequencies where the gate produces no phase shift. In more usual cases some phase compensation is necessary and occurs at the input of the pi network due to the presence of R2. At high frequencies, R2 in fact can be considered as the output impedence of the gate and does not need to be added externally. In its simplest form, the gate oscillator thus can be shown as in Figure 3-1b. For frequencies higher than a few MHz, it is necessary to use TTL gates for satisfactory operation. Unfortunately , a large value resistor R is not adequate for biasing a TTL gate into active region. Low values of R, however, produce a considerable signal feedback, thus reducing the gain. The arrangement shown in Figure 3-1c was originally suggested by Dr. Newel of *CTS, Knights, Inc.*, and seems to be reasonably acceptable. Measurements on this configuration showed slight improvement in stability, however, it needs an extra resistor from input to ground.

A more elaborate scheme of biasing is shown in Figure 3-1d. This circuit works very well with 74LS04 but not with 74S04. Although there is no free running or relaxation oscillation present when the crystal is disconnected, this circuit has a serious defect that it sometimes changes its mode of oscillation from 3rd harmonic to fundamental. These observations were confirmed with Dr. Newel, who also noted similar results. This configuration has been regarded as undesirable in our application since it needs a greater number of external components.

EXPERIMENTAL RESULTS

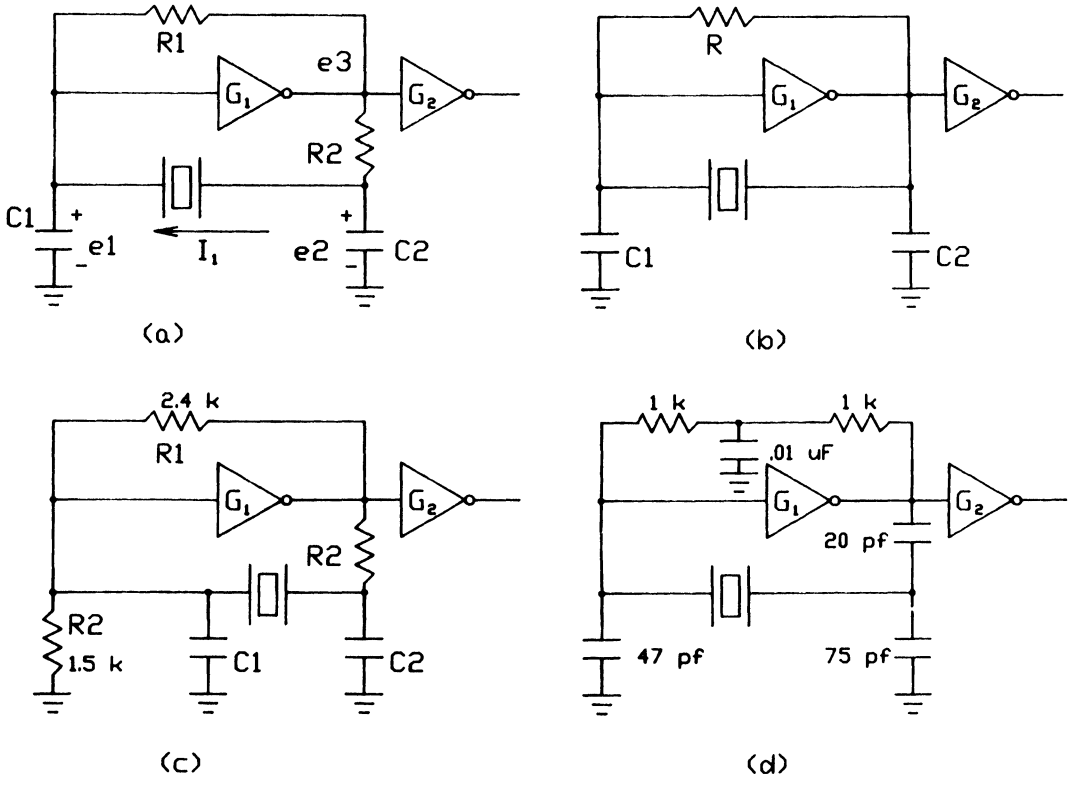
The models described in Figures 3-1b and 3-1c have been thoroughly tested to determine with reasonable assurance that the circuit will perform properly when produced in quantity. Also, the stability of frequency of oscillation with respect to variation in supply voltage and temperature is within the acceptable limit. Since the oscillator is a part of the whole clock generator chip, it is important to find out how much current it takes and how this current can be optimized. The oscillator circuit will require two buffers, one for driving the internal gates and the other to supply outside the chip, each having the capacity to sink at least 24 mA. A 74S04 having six inverters each with current sinking capacity of 20 mA has been used to evaluate the performance of the oscillator. Adhering to parts made by the AMD process, a 74S158 with select and strobe lines connected to GND was used to give four inverters similar to those in 74S04. Only three of the gates are used, one biased to act as the oscillator and the other two as buffers and the rest of the gates are connected to remain in the high state. Initially, a biasing resistor of 390 ohms is used for the model of Figure 3-1b and subsequent performance with a 680 ohm bias resistor has also been examined. To find the mode of oscillation to which the circuit is locked, different combinations of C1 and C2 are chosen and different frequency XTAL is plugged into the circuit. Table 3-1 shows the typical results where C2 has been kept constant.

When this type of data was taken with other values of C2, it is observed that:

a) The circuit oscillates even without the crystal; the frequency of oscillation, the natural frequency, can be chosen by varying C1 and C2.

b) When the crystal is inserted in the circuit, the frequency of oscillation is locked to either fundamental or some overtone of the XTAL depending on the natural frequency of the circuit.

c) The natural frequency of the circuit needs to be slightly lower than the frequency at which the crystal is desired to be operated.



02188A-6

Figure 3-1

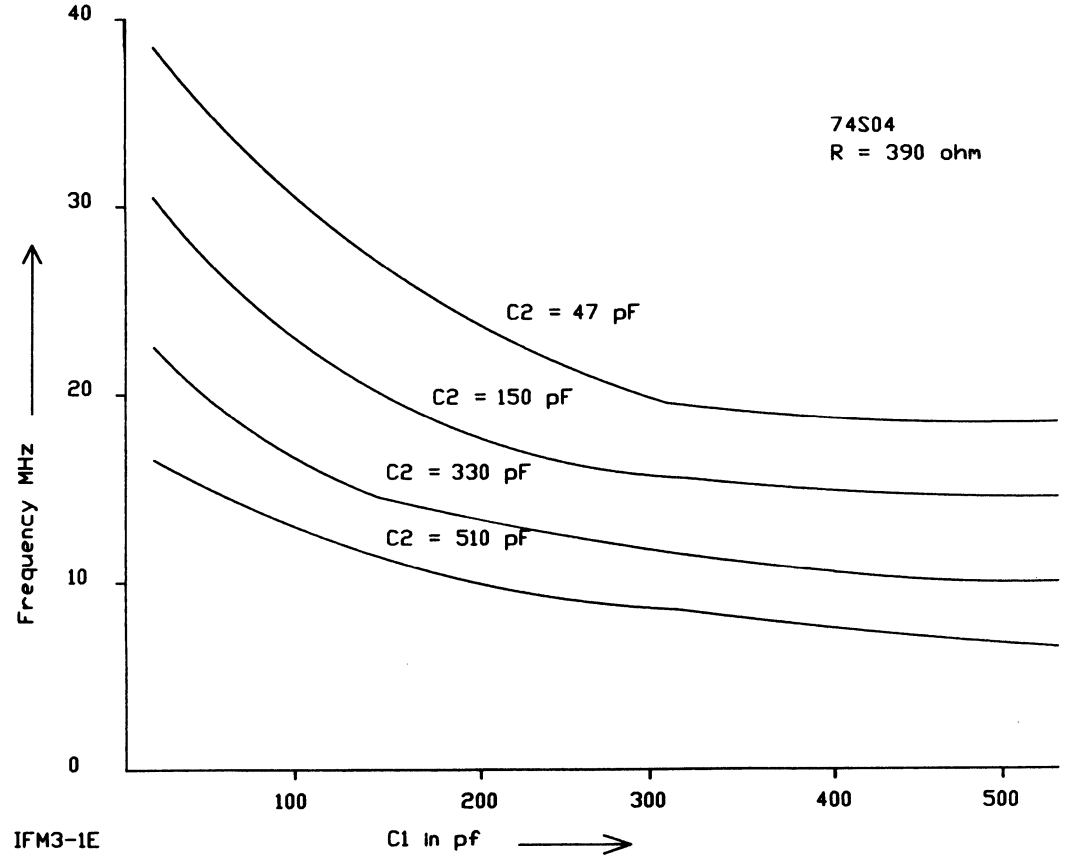
Table 3-1

$C1_{pF}$	$C2_{pF}$	Frequency of Oscillation in MHz				
		Without XTAL	17 MHz XTAL	27 MHz XTAL	36 MHz XTAL	=40MHz XTAL
0	47	58.425	STOP	STOP	STOP	STOP
18	"	42.959	41.980	44.998[5]	45.974	44.260
39	"	36.380	51.715[3]	44.998[5]	37.582	39.833[3]
47	"	34.651	51.715[3]	44.998[5]	36.000[3]	39.833[3]
150	"	25.393	25.857	27.000[3]	36.000[3]	39.833[3]
330	"	15.917	17.227[1]	27.000[3]	19.924	16.080
510	"	14.114	17.226[1]	14.999	15.110	14.938
620	"	11.106	17.226[1]	12.954	11.989[1]	13.265[1]
750	"	9.346	17.226[1]	13.822	9.111	9.065

NOTES:
(1) XTAL is working at fundamental mode.
(3) XTAL is working at third overtone.
(5) XTAL is working at fifth overtone.

Since C1 and C2 are connected outside the chip, the customer has the flexibility of choosing his operating frequency and trim to the accuracy he desires. A series of graphs as shown in Figure 3-1e can also be specified to the customer for selecting the natural frequency of oscillation for a particular bias resistance used.

Measurements of deviation in frequency in PPM (Parts per million) due to change in supply voltage and measurements of current taken by both the oscillator and the two buffer circuits are also made for both models and the typical results are given in Table 3-2.



02188A-7

Figure 3-1e

Table 3-2

MODEL	R1 Ω	R2 Ω	Vcc volts	ICC mA @ 25°C	fo KHz	PPM/volt (5.5v→4.5v)
I with a single resistor biasing	390	-	5.5	38.4	36001.361	147/36
			5	34.2		
	390	-	5.5	38.3	27000.853	90/27
			5	34.1		
	680		5.5	37.7	36001.372	12/36
			5	33.3		
	680	-	5.5	37.0	27000.834	46/27
			5	33.1		
II with two resistor biasing	2.4K	1.5K	5.5	36	36001.270	89/36
			5	31.4		
	2.4K	1.5K	5.5	33.8	27000.503	20/27
			5	30.1		

10% increase in Icc has been observed at 125°C. Icc depends on the biasing resistor to some extent but to a larger extent on the gates used. For example, keeping the buffer gates the same (74S04), if the oscillator section is changed from 74S04 to 74LS04, about 8 mA of current can be saved. Using a higher value of bias resistor in Model I can apparently improve the stability factor (PPM), but it has the adverse effect of decreasing the equivalent negative resistance of the oscillator. The equivalent negative resistance of an oscillator is a measure of the highest crystal resistance that can be used without stopping the oscillation. It has been found that by increasing the bias resistance from 390 ohms to 680 ohms, the negative resistance has decreased by 24 ohms at 36 MHz and by 60 ohms at 27 MHz.

Measurements of the performance of the oscillator at 125°C and at −55°C have been made. A maximum deviation in frequency of 7 PPM has been observed by varying temperature from −55°C to 125°C with 5.5v V_{cc} at 36 MHz.

CONCLUSION

The result of these brief experiments demonstrated the performance of the simple oscillator of Model I is satisfactory and adequate for general purposes. The Model II oscillator is more stable in the sense that it does not oscillate when the XTAL is not in the circuit. It has a smaller Icc figure but it needs an extra resistor. Since the bias resistors are connected outside the chip, the customer has the option to choose his model.

Since the buffer gates need to sink 24 mA each, only the 74S04 type of gates are to be used. Some amount of current can be reduced by using high resistance values in the oscillator gate. At times the oscillator works even with 74LS04 or 74S04 type of gate, but the performances at high frequency and low temperature with 4.5v V_{cc} are poor. This is due to the higher propagation delay in the gates. A 74S04 type of gate (4K ohms on the base of the input transistor and 1.6K ohms on the phase splitter) made in Schottky process will have less propagation delay and work satisfactorily as an oscillator with a reduction of at least 5 mA in current.

No reliable measurement has been made on the drive level of the crystal. Dr. Newel has confirmed that the drive level for the crystal in these models is quite safe.

3.1.1 MOS OSCILLATORS

While gate oscillators are quite popular, they can cause problems ranging from temperamental operation to lack of oscillation. The gain elements are the primary problem source and it is not possible to reliably identify the analog characteristics of digital gates, and there is no guarantee that gates from various manufacturers will produce the same results when plugged into the oscillator circuit. Furthermore, some circuits seem to favor certain gate locations within the IC package.

There are four major reasons for crystal resonant frequencies drift:

— Aging
— Temperature change
— Supply-voltage or load variations
— Mechanical disturbances

For good frequency stability and reasonably simple circuit design, a Pierce-type oscillator that employs discrete components should be considered.

The Pierce-type oscillator is specially good for controlling temperature effects. Other advantages are:

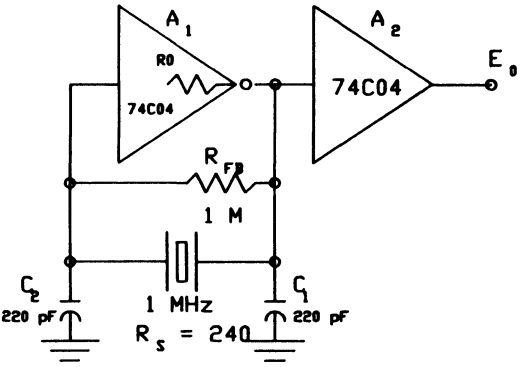
— Lower power dissipation above 20MHz.

— Paralleled fixed and variable capacitors allow fine frequency trimming. Trimming with variable capacitor, of 15 to 40 pf in value, in series with the crystal.

— Gain is not as high as other types of oscillators but Zin is higher. High gain transistors can be used in either single or multiple stage amplifiers.

— Closely resembles the type AMD uses.

Figure 3-1.1a shows a typical Pierce-type oscillator using B series CMOS inverters. The disadvantage of this circuit is A_1's low gain. To increase the loop gain, however, A_1's output resistance R_0 serves in place of a discrete resistor to develop the necessary RC phase lag. This scheme does reduce frequency stability slightly and increases crystal power dissipation. To ensure that oscillations start when power is applied, R_{FB} dc-biases A_1's input and output for Class A operation. A_1's output is not logic-level compatible, so A_2 increases the amplitude to a full rail-to-rail swing. A_2 also minimizes circuit rise and fall times and buffers the oscillator output. With 74C Series CMOS inverters, this circuit has a maximum operating frequency of about 2 MHz. High performance CMOS such as those found on modern peripherals have CMOS oscillators that operate up to 20 MHz. For higher frequency operation, ECL devices can be used to integrate 20-MHz Pierce oscillators (Figure 3-1.1b).



02188A-8

Figure 3-1.1a

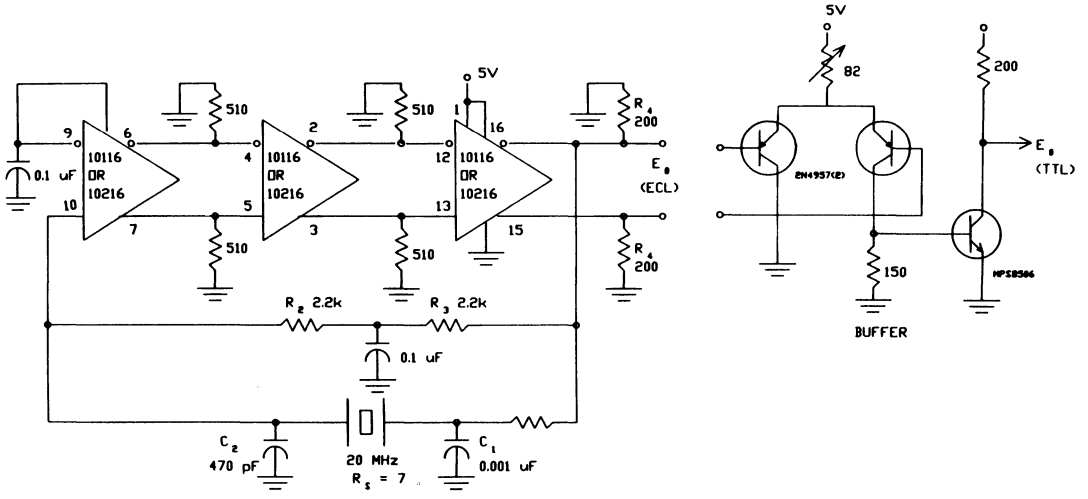


Figure 3-1.1b

02188A-9

The three line receivers reside in one DIP, and all are series cascaded to develop sufficient circuit gain. This design also includes a buffer-circuit arrangement, for applications that involve other than ECL devices. Because the Pierce oscillator and TTL are not compatible, due to the logic family's poor switching parasitics at low signal amplitudes, the buffer also provides a means of employing a high-frequency IC-based Pierce oscillator in a TTL system. One can provide frequency-trimming capability by connecting a variable capacitor (15 to 40 pf) in series with the crystal.

It should be noted here that the total capacitance should include the C_{in} of the part, and not just the values of the bypass capacitors used in the circuit. C_{in} of the part should be in the range of 5 to 10pf. Use manufacturer's recommended capacitors for proper oscillation but use total capacitance in calculations.

Table 3-3 shows the frequency shift sensitivity to the supply voltage, in a Pierce-type oscillator.

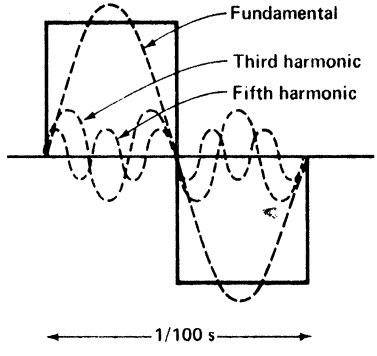
Table 3-3

Frequency Sensitivity To Supply Voltage

Circuit	Frequency Shift ΔF (PPM) (For Δ Vcc = 2V DC)
Pierce * 1 MHz	0.9
* 20 MHz	1.5
* 4 kHz	0
* 1 MHz at series resonance	0.8
* CMOS 1 MHz	3
* ECL 20 MHz	2

3.1.2 OVERTONE CRYSTAL OSCILLATOR

Due to processing limitations during manufacture, the maximum resonant frequency of most quartz crystals is approximately 20 MHz. For frequencies above 20 MHz, or if using a crystal with a fundamental frequency that is too low for an application, the third harmonic of that frequency can be used. Figure 3-1.2a shows how a square wave corresponds to a fundamental sine wave with odd harmonics. The square wave has only odd harmonics, and when we add the first and third harmonics together, the superpositioning produces a waveform that is more square than sinusoidal (Figure 3-1.2b). Figure 3-1.2c shows an overtone crystal oscillator with MECL 10,000 devices. The MECL circuits in crystal oscillators has several advantages over conventional circuitry. When a second gate is used as a buffer for the oscillator, there is minimum frequency change with output loading. Also, the high input impedance and 0.8 volt logic swing prevent the possibility of overdriving the crystal. In addition, this particular circuit is frequency insensitive to +/−20% power supply variations, and the frequency versus temperature characteristics closely follow the crystal characteristics. The circuit in Figure 3-1.2c employs an adjustable resonant tank circuit which insures operation at the desired crystal overtone. C_1 and L_1 form the resonant tank circuit, which, with the values specified, has a resonant frequency adjustable from approximately 50 MHz to 100 MHz. Overtone operation is accomplished by adjusting the tank circuit frequency at or near the desired frequency. The tank circuit exhibits a low impedance shunt to off-frequency oscillations, suppressing the first and fifth harmonic; and a high impedance to the desired frequency, allowing feedback from the output.



02188A-10

Figure 3-1.2a

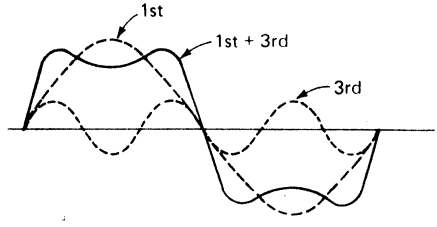


Figure 3-1.2b

02188A-11

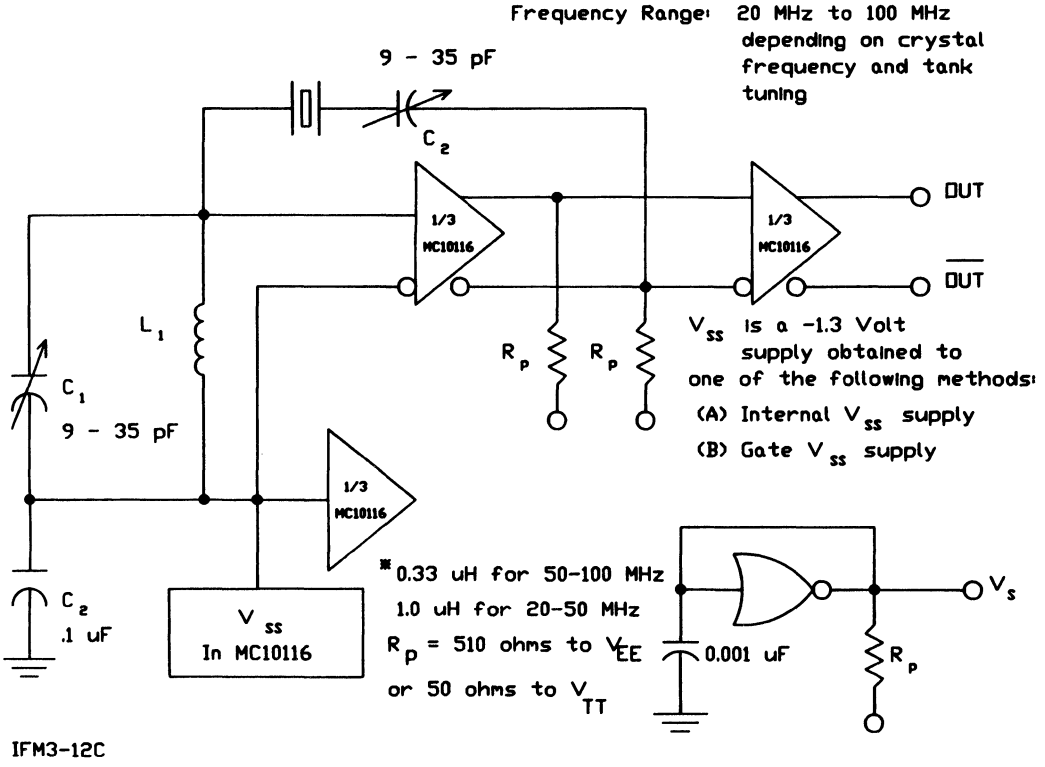


Figure 3-1.2c

Operation in this manner guarantees that the oscillator will always start at the correct overtone.

The values of inductor and capacitor to be used are calculated by using the following formula:

$$f = \frac{1}{2\pi\sqrt{C_L L_1}}$$

3.2 TEMPERATURE CONSIDERATIONS

DEFINITION OF THERMAL RESISTANCE

THERMAL RESISTANCE

The reliability of an integrated circuit is largely dependent on the maximum temperature which the device will attain during operation. Because the stability of a semiconductor junction declines with increasing temperature, knowledge of the thermal properties of the packaged device becomes an important factor during device design. In order to increase the operating lifetime of a given device, the junction temperatures must be minimized. This demands knowledge of the thermal resistance of the completed assembly and specification of the conditions in which the device will function properly. As devices become both smaller and more complex and the requirement for high speed operation becomes more important, heat dissipation will become an ever more critical parameter.

Thermal resistance is defined as the temperature rise per unit power dissipation above some reference condition. The unit of measure is typically °C/watt. The relationship between junction temperature and thermal resistance is given by:

$$T_j = T_x + P_d * R_{JX} \quad (1)$$

where T_j = junction temperature
T_x = reference temperature
P_d = power dissipation
R_{JX} = thermal resistance
X = some defined test condition.

In general, one of the following three conditions is defined for measurement of thermal resistance:

R_{JC}—Thermal resistance measured with reference to the temperature at some specified point on the package surface.

R_{JA}—Thermal resistance measured with respect to the (still air) temperature of a specified volume of still air.

R_{JA}—Thermal resistance measured with respect to the (moving air) temperature of air moving at a specified velocity.

The relationship between R_{JA} is

$$R_{JA} = R_{JC} + R_{CA}$$

where R_{CA} is a measure of the heat dissipation due to natural convection (still air) or forced convection (moving air) and the effect of heat radiation and mounting techniques. R_{JC} is dependent solely on material properties and package geometry; R_{JA} include the influence of the surface area of the package and environmental conditions. Each of these definitions of thermal resistance is an attempt to simulate some manner in which the package device may be used.

The thermal resistance of a packaged device, however measured, is a summation of the thermal resistances of the individual components of the assembly. These in turn are functions of the thermal conductivity of the component materials and the geometry of the heat flow paths. Like other material properties, thermal conductivity is usually temperature dependent. For alumina and silicon, two common package materials, this dependence can amount to a 30% variation in thermal conductivity over the operating temperature range of the device. The thermal resistance of a component is given by

$$R = \frac{L}{K(T)A}$$

where L = length of the heat flow path
A = cross-sectional area of the heat flow path
$K(T)$ = thermal conductivity as a function of temperature

and the overall thermal resistance of the assembly (discounting convective effects) will be:

$$R = \Sigma R_n = \Sigma \frac{L}{K_n A}$$

But since the heat flow path through a component is influenced by the materials surrounding it, determination of L and A is not always straightforward.

A second factor that effects the thermal resistance of a packaged device is the power dissipation level and, more particularly, the relationship between power level and die geometry, i.e. power distribution and power density. By rearrangement of equation 1 to

$$P_d = \frac{1}{R_{JX}} (T_j - T_x) = \frac{1}{\Sigma R_n} (T_j - T_x)$$

the relationship between P_d and T_j can be more clearly seen. Thus, to dissipate a greater quantity of heat for a given geometry, T_j must increase and, since the individual R_n will also increase with temperature, the increase in T_j will not be a linear function of increasing power levels.

A third factor of concern is the quality of the material interfaces. In terms of package construction, this relates specifically to the die attach bond, and for those packages having a heatsink, the heatsink attach bond. The quality of the die attach bond will most severely influence the package thermal resistance as this is the area which first impedes the transfer of heat out of the silicon die. Indeed, it seems likely that the initial thermal response of a powered device can be directly related to the quality of the die attach bond.

EXPERIMENTAL METHOD

The technique for measurement of thermal resistance involves the identification of a temperature-sensitive parameter on the device and monitoring this parameter while the device is powered. For bipolar integrated circuits the forward voltage of the substrate isolation diode provides a convenient parameter to measure and has the advantage of a linear dependence on temperature. MOS devices which do not have an accessible substrate diode present greater measurement difficulties and may require simulation through use of a specially designed thermal test die. Choice of the parameter to be measured must be made with some care to insure that the results of the measurement are truly representative of the thermal state of the device being investigated. The measurement of the substrate isolation diode which is generally diffused across the area of the die yields a weighted average of the condition of the individual junctions across the die surface. Measurement of a more local source would yield a less generalized result.

For those MOS devices for which no useful parameter is available, simulation is accomplished by using the thermal test die. The basis for this test die is a 2 mil square cell containing an isolated diode and a 1 k/ohm resistor. The resistors are interconnected from cell to cell on the wafer before it is cut into multiple arrays of the basic unit cell. In use, the device is powered via the resistors with voltage or current adjusted for the proper level and the voltage drop of the individual diodes is monitored as in the case of actual devices.

Prior to the thermal resistance test, the diode voltage/ temperature calibration must be determined. This is done by measuring the forward voltage at 1 mA current level at two different temperatures. The diode calibration factor is then:

$$K_f = \frac{T_2 - T_1}{V_2 - V_1} = \frac{T}{V}$$

in units of °C/mV. For most diodes used for this test the voltage/temperature relationship is linear and these two measurement points are sufficient to determine the calibration. The actual thermal resistance measurement has two alternating phases: measurement and power on. The device under test is pulse powered with an ON duty cycle of 99% and a repetition rate of <100 Hz. During the brief OFF states the device is reverse-biased with a 1 mA current and the voltage drop is measured. The series of voltage readings are averaged over short periods and compared to the voltage reading obtained before the device was first powered ON. The thermal resistance is then computed as:

$$R_{JX} = \frac{K_f(V_f - V_i)}{V_H I_H} = \frac{K_f V}{P_d}$$

where K_f = calibration factor
V_i = initial forward voltage value
V_f = current forward voltage value
V_H = heating voltage
I_H = heating current

The pulsing measurement is continued until the device has reached thermal equilibrium and the final value measured is the equilibrium thermal resistance of the device under test.

When the end result desired is R_{JA} (still air), the device and the test fixture (typically a standard burn-in socket) are enclosed in a box containing approximately 1 cubic foot of air. For R_{JC} measurements the device is attached to a large metal heatsink. This ensures that the reference point on the device surface is maintained at a constant temperature. Through the use of heaters attached to the metal fixture, the "case" temperature may be maintained at any specified value above ambient. The requirements for measurement of R_{JA} (moving air) are rather more complex. They involve the use of a small wind tunnel with capability for monitoring air pressure, temperature and velocity in the area immediately surrounding the device tested. Standardization of this last test requires much careful attention.

AMD's parts are designed not to exceed 160°C junction temperature, at the maximum ambient temperature. Table 3.4 lists the thermal resistance for packages currently in production.

SECTION B
16-BIT PROCESSORS

4.0 INTERFACING TO THE 8086/80186

4.1 8086 (ALSO CALLED iAPX 86) OVERVIEW

The 8086 is a general purpose 16-bit microprocessor CPU. The CPU has a 16 bit data bus multiplexed with 16 address outputs. There are 4 additional address lines (segment addresses which are multiplexed with STATUS) that increase the memory range to 1 Mbyte. 8086 addresses are specified as bytes. In a 16 bit word, the least significant byte has the lower address and the most significant byte has the higher address. This is compatible with 8080, 8085, Z80 and PDP11 addressing schemes but differs from the Z8000 and 68000 addressing.

The data bus is "asynchronous", i.e., the CPU machine cycle can be stretched without clock manipulation by inserting Wait states between T2 and T3 of a read or write cycle to accommodate slower memory or peripherals. Unlike the 68000, the 8086 has separate address space for I/O (64 kBytes).

The 8086 can operate in MIN. or MAX. mode. Maximum mode offloads certain bus control functions to a peripheral device and allows the CPU to operate efficiently in a co-processor environment. A brief discussion on both the MIN. and MAX. modes are as follows:

MIN. mode: I/O addressing is defined by a High or the IO/$\overline{M}$ output, and activated by the $\overline{RD}$ output for reading from memory, or I/O or activated by the $\overline{WR}$ output for writing to memory or I/O.

DMA: The Bus is requested by activating the HOLD input to the 8086. Bus Grant is confirmed by the HLDA output from the 8086.

MAX. mode: (8086 plus 8288) I/O operation is controlled by two outputs from the 8288.

- $\overline{IORC}$: active during Read from I/O
- $\overline{IOWC}$: active during Write to I/O
- $\overline{MRDC}$: active during Read from memory
- $\overline{MWTC}$: active during Write to memory

DMA: The Bus is requested and Bus Grant is acknowledged on the same pin ($\overline{RQ}/\overline{GT0}$ OR $\overline{RQ}/\overline{GT1}$) through a pulsed handshake.

INTERRUPTS IN MIN. AND MAX. MODES:

Interrupt is requested by activating the INTR or NMI inputs to the 8086.

Interrupt is acknowledged by the $\overline{INTA}$ pin on a MIN. mode 8086 or by the $\overline{INTA}$ pin on the 8288 in MAX. mode.

Note: There is no $\overline{RD}$ or $\overline{IORC}$ during the interrupt-acknowledge sequence.

4.2 8086 AND Z8000 PERIPHERALS

Z8000 peripherals may be interfaced to the 8086 in many ways. One thing distinguishes the Z80xx peripherals from others is that they require data to be valid prior to the falling edge of $\overline{DS}$ during a write. The way to go about meeting this requirement is to use a 74LS74 to delay the falling edge of $\overline{WR}$ and OR $\overline{RD}$ with $\overline{WR}$. By using the asynchronous ready mode of the 8284A and the other half of the 74LS74, one Wait State is inserted in both the read and write cycles, and thus stretches $\overline{DS}$ to meet the minimum pulse width requirements. In the examples that follow, one assumes that there are other devices in the system that require the use of the Am9519A. In another example, the Am9519A is not needed to interface the peripherals interrupt structure. $\overline{INTA}$ is tied high since the vector will be supplied by the Am9519A. The interrupt service routine can read the peripheral status to determine the cause of the interrupt. The service routine should also clear IP and IUS before returning to mainline code. Note, since $\overline{INTA}$ is tied High, IUS needs not be cleared. If $\overline{INTA}$ is activated, IUS must also be cleared.

Many of the design examples in this manual use PAL devices. PAL devices, in many cases, will be the most cost effective solution. The SSI examples show what is being accomplished much clearer than by using a PAL device, which is a black box to many. Gates are often left over from one circuit which can be used to implement other circuits. Whether to use a PAL device or do a discreet design is an individualistic choice and is not always clear-cut. As a general rule, if you can replace four or more packages, use a PAL device.

4.2.1 8086 TO Z8000 PERIPHERALS WITHOUT INTERRUPTS

This interface is designed to work with both the 5 MHz and 8 MHz 8086 (Figure 4-2.1a). It is also compatible with the iAPX 186. The 5 MHz CPU should use 4 MHz peripherals and the 8 MHz should use 6 MHz peripherals. A timing diagram for this interface is shown in Figure 4-2.1b.

Chip Select ($\overline{CS}$) is latched on the falling edge of ALE in the Z8000 peripherals. Alternate address decoding schemes may be used. Care must be taken to meet the $\overline{CS}$ setup time with respect to the rising edge of $\overline{AS}$ (falling edge of ALE). Reset is accomplished on the Z80xx peripherals by asserting $\overline{AS}$ and $\overline{DS}$ simultaneously. This is accomplished by the two NOR gates shown. Reset may be done by software, thus eliminating the extra gates. The SCC and CIO will always recognize a software reset, so a hardware reset needs not be implemented.

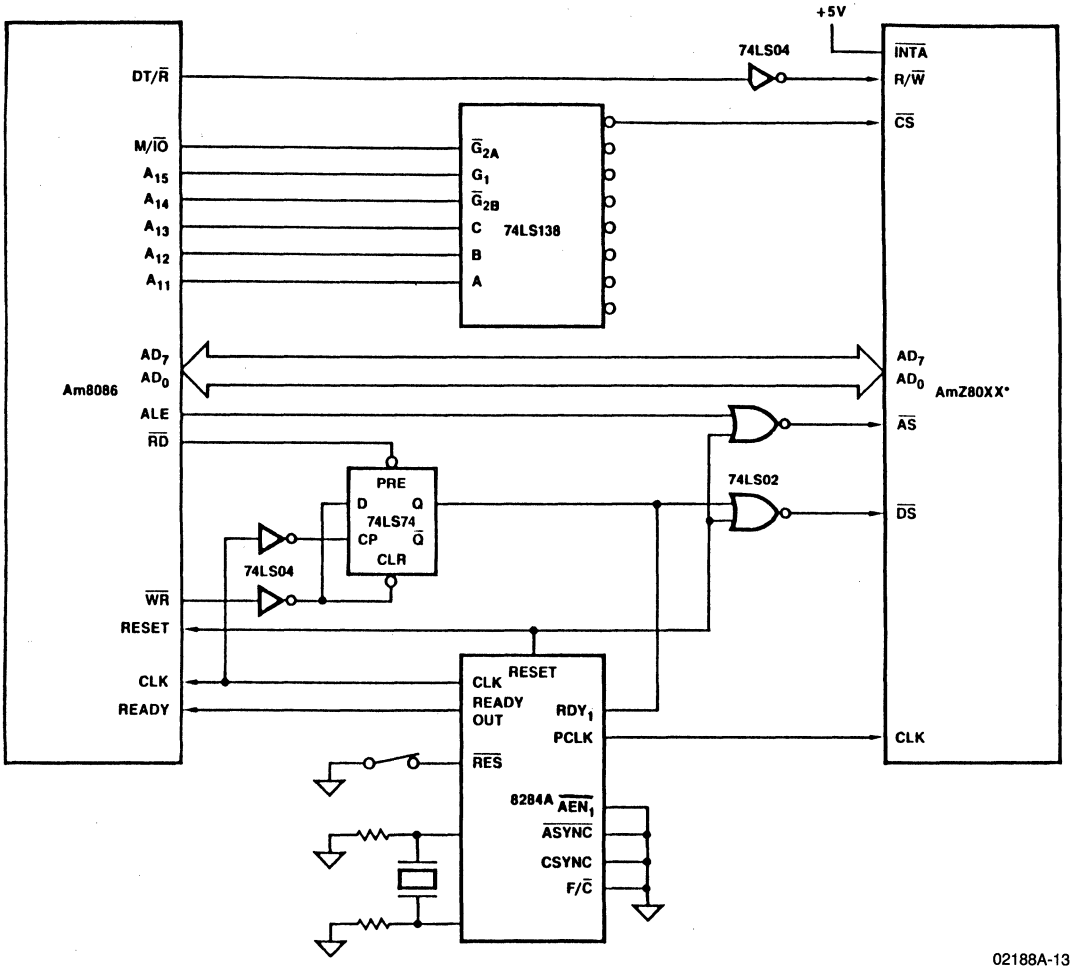


Figure 4-2.1a

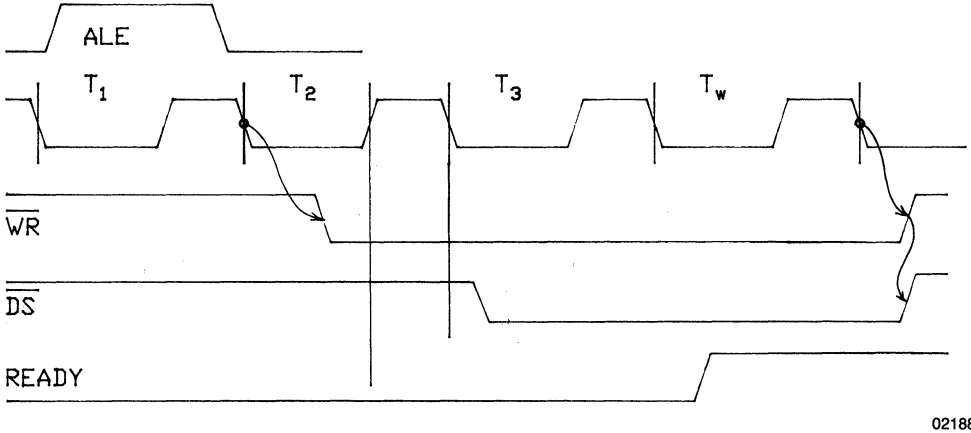


Figure 4-2.1b

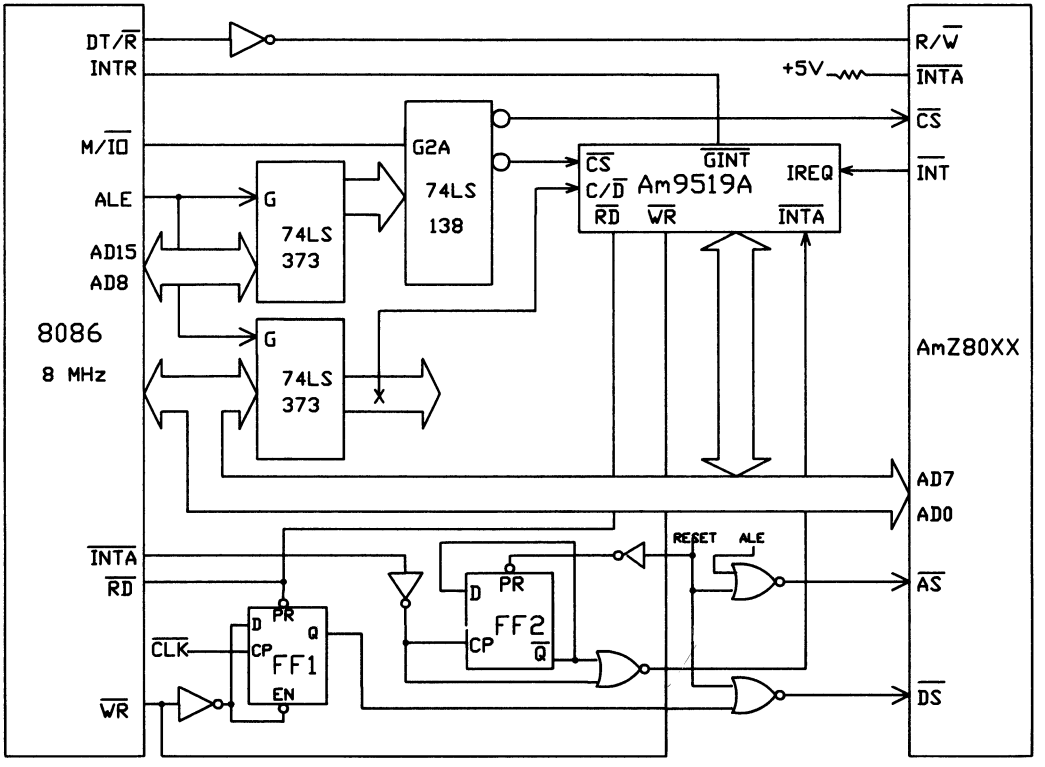
A 74LS74 is used to delay the falling edge of $\overline{WR}$; $\overline{RD}$ is ORed with $\overline{WR}$ by the flip-flop which takes the advantage of the preset overriding clear. By using the asynchronous ready mode of the 8284A, one Wait State is inserted in both the read and write cycles. This stretches $\overline{DS}$ to meet the minimum pulse width requirements.

When programming this part, it is important not to violate the recovery time when two successive I/O operations are to be performed. Intervening instruction fetches will usually be sufficient; however, NOP's may need to be inserted with faster processors. Block I/O instructions should also be avoided. Consult individual data sheets for the recovery time on each part. The SCC has an unusually long recovery time of 6 PCLKS which is 13 CPU clocks. This is the simplest level of interface, as the interrupts were not used. Note that $\overline{INTACK}$ must be tied high when not used.

4.2.2 8086 TO Z8000 PERIPHERALS WITH INTERRUPTS

Z8000 peripherals maybe interfaced to the 8086 with other devices in the system. In the example given here (Figure 4-2.2), it is assumed that other devices in the system require the Am9519A Interrupt Controller. The Z80XX peripheral generates an interrupt ($\overline{INT}$) and since $\overline{INTA}$ is tied high the vector will be supplied by the Am9519A. The interrupt service routine can read the peripheral status to determine the cause of the interrupt. $\overline{INT}$ is cleared with software.

FF_1 and the NOR gates delay the falling edge of $\overline{DS}$ and supply hardware reset as discussed in Chapter 4.2.1. The hardware to insert one Wait state is not shown. The second conversion necessary is to make the two CPU Interrupt Acknowledge pulses appear as one $\overline{INTA}$ pulse to the peripheral. This is done by FF_2 in this example. Although this design was for 8 Mhz, it also works at 5 Mhz.



02188A-15

Figure 4-2.2

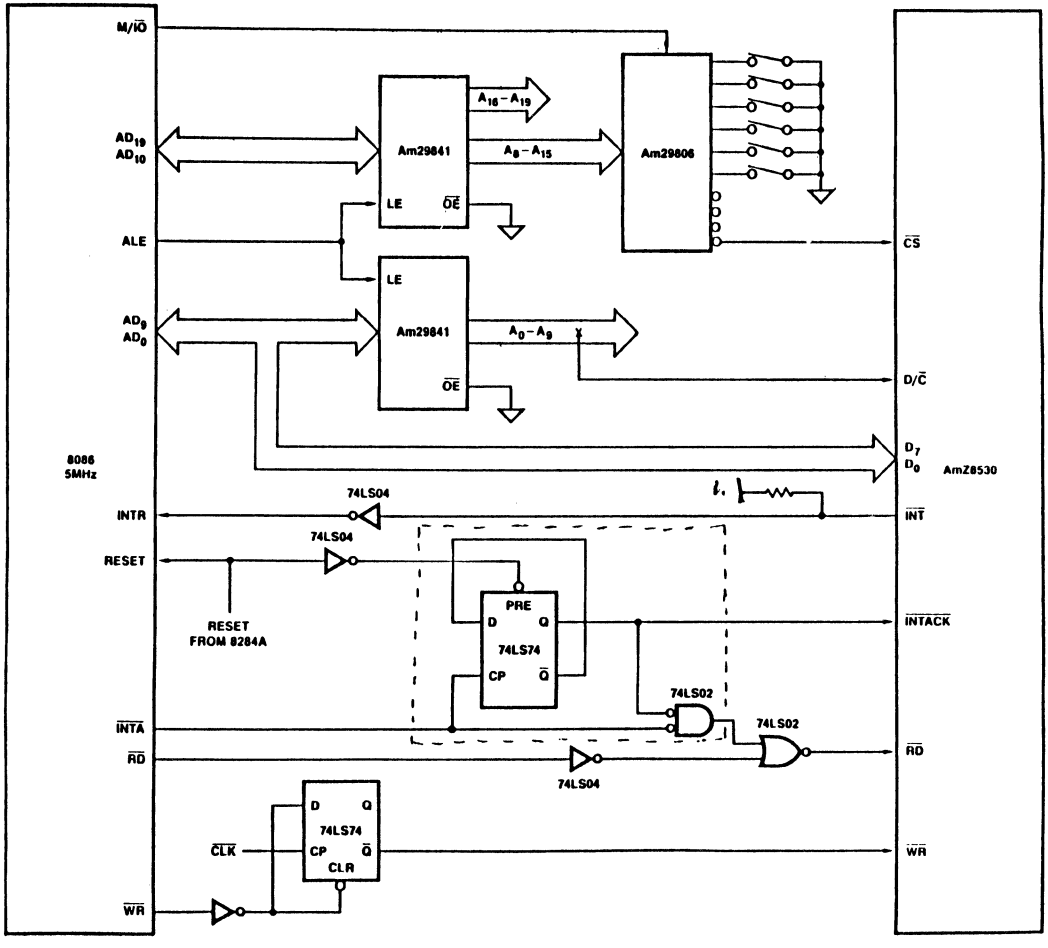
4.2.3 THE 8086 AND AmZ8530 INTERFACE

Most common systems demultiplex address and data. The AmZ85XX Family of peripherals was developed to be compatible with these systems. These devices are identical to the Z8000 peripherals as far as functionality, but the bus interface has been changed.

Interface between the 8086 and the AmZ8530 peripheral device shows how to take advantage of the AmZ85XX interrupt structure. $\overline{\text{INTACK}}$ is generated by the 8086's first $\overline{\text{INTA}}$ pulse. This allows about 800 nsec for the interrupt daisy-chain to settle. The second $\overline{\text{INTA}}$ pulse is then gated to the $\overline{\text{RD}}$ pin which places the vector on the bus. At 8 MHz, two Wait States must be inserted. This design is the same when interfacing to the 186. It requires no additional Wait States. Diagrams in Figure 4-2.3b show the connections for 74LS74 in both 5 MHz and 8 MHz operations of the 8086. Figure 4-2.3c is an alternate implementation which can be used in place of the logic in the dotted area in Figure 4-2.3a.

Note that the falling edge of $\overline{\text{WR}}$ must be delayed to meet data setup time requirements. A Wait State must be inserted (not shown) to meet pulse width requirements during a write. See the Z80XX interface for discussion of recovery time.

The Am29841 is a high speed 10-bit latch with high drive capability. Other latches may be used instead. Most designers latch $\overline{\text{BHE}}$ even though S_7 is the same, the few extra bits become quite useful when trying to keep parts count down.



02188A-16

Figure 4-2.3a

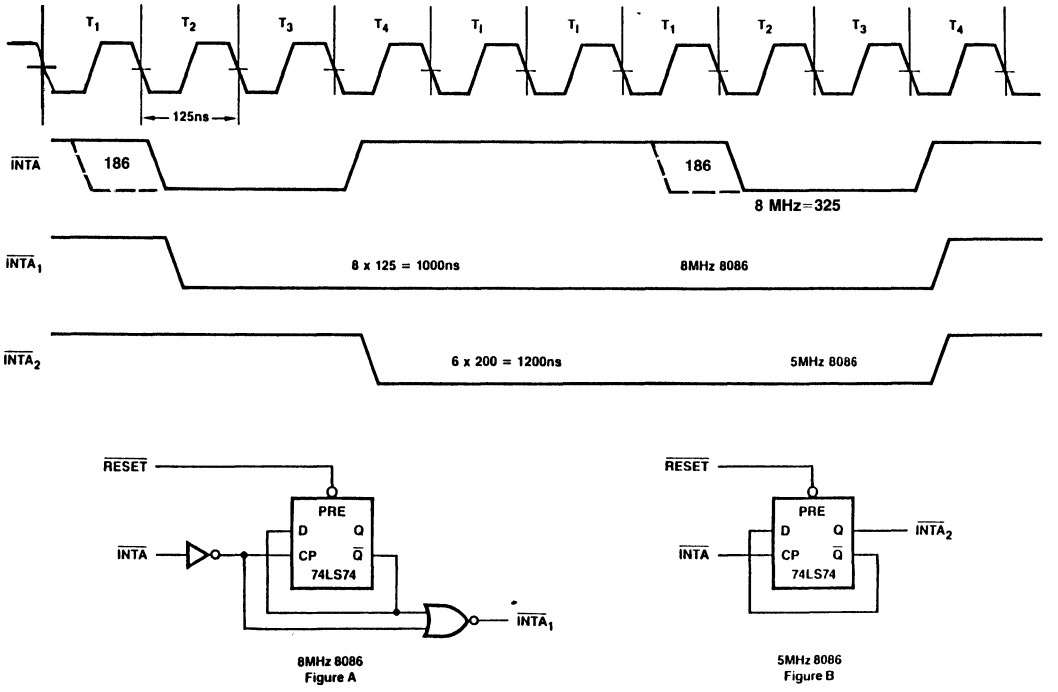


Figure 4-2.3b

02188A-17

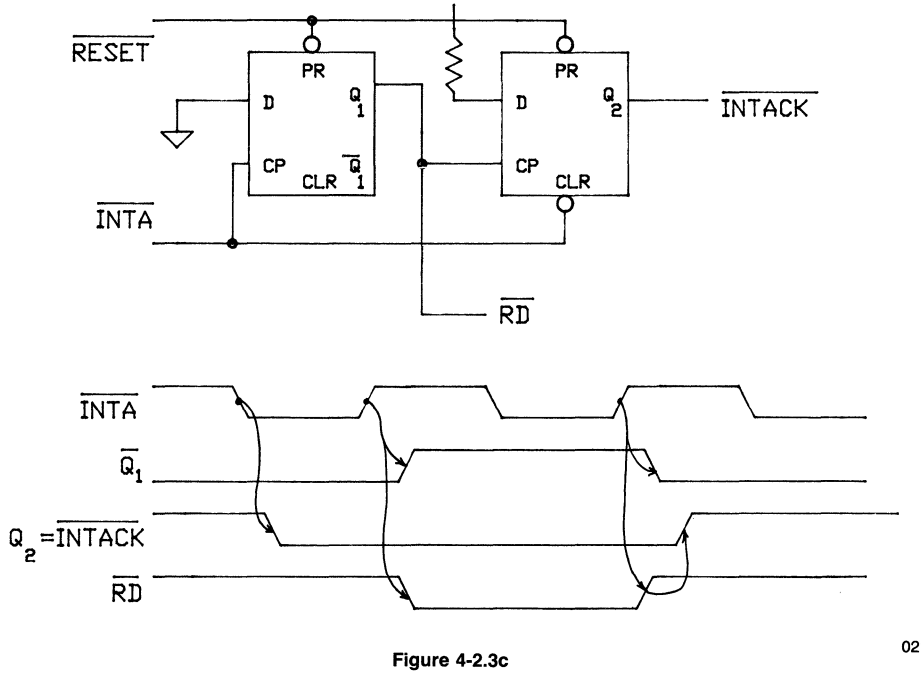


Figure 4-2.3c

02188A-18

The sample assembly initialization routine, Figure 4-2.3d, takes into account the READ/WRITE recovery time and has been tested in an 8 MHz system. The interrupt service routines for Channel B were for testing only, and are not intended as realistic examples. Also, Figure 4-2.3e shows a sample initialization sequence written in "C".

```
AMC AMDOS 8/8 64K, Version 2.00 - 9/26/80

type int8530.a86
;       THIS ROUTINE WRITTEN FOR THE 8086
;       INITIALIZES THE 8530 SCC FOR ASYNCHRONOUS
;       OPERATION AT 9600 BAUD.

;       REGISTER USAGE
;       AX      INPUT AND OUTPUT DATA
;       BX      COUNTER
;       DX      ADDRESS OF PORT
;       BP      POINTER TO MEMORY

CMD     EQU     0F902H          ;COMMAND/STATUS PORT
DATA    EQU     0F900H          ;DATA PORT
TAB     EQU     OFFSET TABLE
ET      EQU     OFFSET ETB

        ORG     0100H

START:  MOV     DX,CMD          ;GET ADDRESS OF CMD PORT
        IN      AL,DX           ;READ STATUS TO SET STATE
        MOV     BX,ET-TAB       ;GET TABLE LENGTH
        MOV     BP,TAB          ;POINT TO TABLE
RPT:    MOV     AL,[BP]         ;GET DATA FROM TABLE
        OUT     DX,AL           ;OUTPUT ADDRESS OF REGISTER
        INC     BP              ;INCREMENT TABLE POINTER
        MOV     AL,[BP]         ;GET DATA
        OUT     DX,AL           ;OUTPUT DATA TO REGISTER
        INC     BP              ;INCREMENT TABLE POINTER
        DEC     BX              ;COUNT-1
        JZ      RPT             ;REPEAT IF NOT DONE
;PROGRAM CONTINUES AS DESIRED

TABLE   DB      009H            ;ADDRESS WR9
        DB      0C0H            ;RESET COMMAND
        DB      003H            ;ADDRESS WR3
        DB      041H            ;# OF BITS ETC.
        DB      004H            ;ADDRESS WR4
        DB      04CH            ;DATA
        DB      005H            ;ADDRESS WR5
        DB      028H            ;BITS/CHAR AND TXEN
        DB      00BH            ;ADDRESS WR11
        DB      056H            ;SELECT BAUD GEN
        DB      00CH            ;ADDRESS WR12
        DB      00CH            ;UPPER TC
        DB      00DH            ;ADDRESS WR13
        DB      000H            ;LOWER TC
        DB      00EH            ;ADDRESS WR14
        DB      007H            ;SET DTR AND ENABLE BAUD GEN
ETB:
        END
```

Figure 4-2.3d

```
/*****************************************************************************/
/*              Initializing the AmZ8530 asynchronuously                     */
/*          (using pointer referring to memory map I/O device)               */
/*****************************************************************************/


#include        "stdio.h"
#define         spaddress       0x22F85         /* status port address */
#define         dpaddress       0x22F87         /* dataport address */
#define         rxbit           2               /* receive ready bit */
#define         txbit           1               /* transmit ready bit */
#define         tablesize       16

main ()
{

unsigned char table[tablesize];
unsigned long *ptspa, *ptdpa;
unsigned char value;
int i;

/* data table for initialization */

table[0]=0x9;   /* address WR9 */
table[1]=0xC0;  /* hardware reset SCC */
table[2]=0x3;   /* address WR3 */
table[3]=0x41;  /* set # of bits etc. */
 ıble[4]=0x4;   /* address WR4 */
.able[5]=0x4C;  /* misc mode */
cable[6]=0x5;   /* address WR5 */
table[7]=0x28;  /* bits/char and TXEN */
table[8]=0xB;   /* address WR11 */
table[9]=0x56;  /* select baud generator */
table[10]=0xC;  /* address WR12 */
table[11]=0xC;  /* upper TC */
table[12]=0xD;  /* address WR13 */
table[13]=0;    /* lower TC */
table[14]=0xE;  /* address WR14 */
table[15]=0x7;  /* set DTR/REQ enable baud generator */

/* output data from table to perform initialization */

ptspa=spaddress;        /* pointer to status port address */
ptdpa=dpaddress;        /* pointer to data port address */
value=*ptspa;   /* read status to set the set state machine */
for (i=0; i<tablesize; i++)     /* perform initialization */
        *ptspa=table[i];

/* receive and echo character routine */

for ( ; ; )
   {    while (((*ptspa) & rxbit) == 0);  /* wait for receive ready bit */
        value=*ptdpa;                     /* receive character */
        while (((*ptspa) & txbit) == 0);  /* wait for transmit ready bit */
        *ptdpa=value;                     /* transmit character */
   }
```

Figure 4-2.3e

4.3 THE 8086 AND AMD PROPRIETARY PERIPHERALS

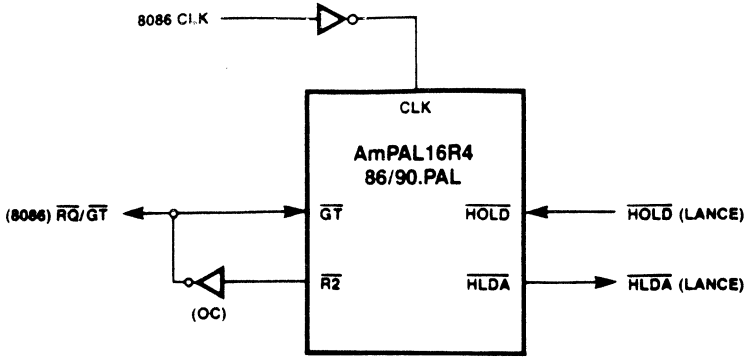
4.3.1 8086 And Am7990 LANCE

The LANCE, Am7990, has been designed to be interfaced easily with the popular 16-bit microprocessors (8086/80186, 68000, Z8000*, LSI-11**). Most of the interface logic is embedded inside the chip and is program selectable.

Although the LANCE itself has a multiplexed bus, it can easily be interfaced to demultiplexed buses with a minimal amount of effort. The following designs assume that the processor and the LANCE reside on the same board. Address buffers and data transceivers are set up to be shared between the processor and the LANCE. All of these designs use PALs to reduce the parts count.

The 8086 to LANCE interface requires a different Bus Request handshake, depending on whether the 8086 is configured in MAX. or MIN. mode. The 8086 has a bidirectional signal for both Bus Request and Bus Grant ($\overline{RQ}/\overline{GT}$). Both Bus Request ($\overline{RQ}$), and Bus Grant ($\overline{GT}$) to/from 8086 are one CPU clock wide, and are synchronous to the CPU clock. Figure 4-3.1a shows a PAL design for the conversions in MAX. mode. This PAL device is utilized to include other external logic requirements for interfacing the LANCE to 8086. The interface diagram is similar to the one for the 80186 to LANCE interface (Chapter 4.3.2) except for the changes made in programming the PAL device. Interface timing diagram is shown in Figure 4-3.1b.

Figure 4-3.1c shows a block diagram on the 8086 to Am7990 interface, in MIN. mode. The interface also employs a PAL device to minimize parts count. The PAL equations are given in Figure 4-3.1d.




```
AMPAL16R4                       RASOUL OSKOUY
PAT001                          MARCH 20, 1984

8086 RQ/GT CONVERSION TO LANCE HOLD/HLDA
ADVANCED MICRO DEVICES

CLK NC HOLD GT   NC NC NC NC NC GND
NC  NC NC   HLDA D2 R2 R1 NC NC VCC

      R1 := HOLD
      D2 := R1
      R2 := R1*D2 + R1*D2
    HLDA := GT*R2 + HLDA*D2
```

This PAL converts the request and grant ($\overline{RQ}/\overline{GT}$) of 8086, when configured in maximum mode, to the LANCE Am7990 $\overline{HOLD}/\overline{HLDA}$. Both request (input to 8086) and grant (output from 8086) are one clock wide and are synchronous to the CPU clock.

02188A-19

Figure 4-3.1a 8086 $\overline{RG}$/ $\overline{GT}$, Am7990 $\overline{HOLD}$/ $\overline{HLDA}$ Conversion PAL device

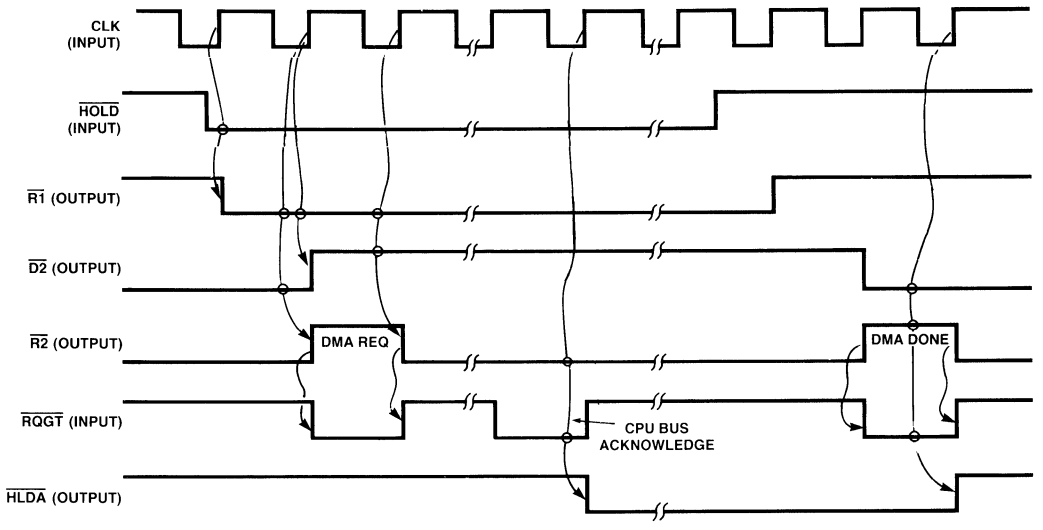
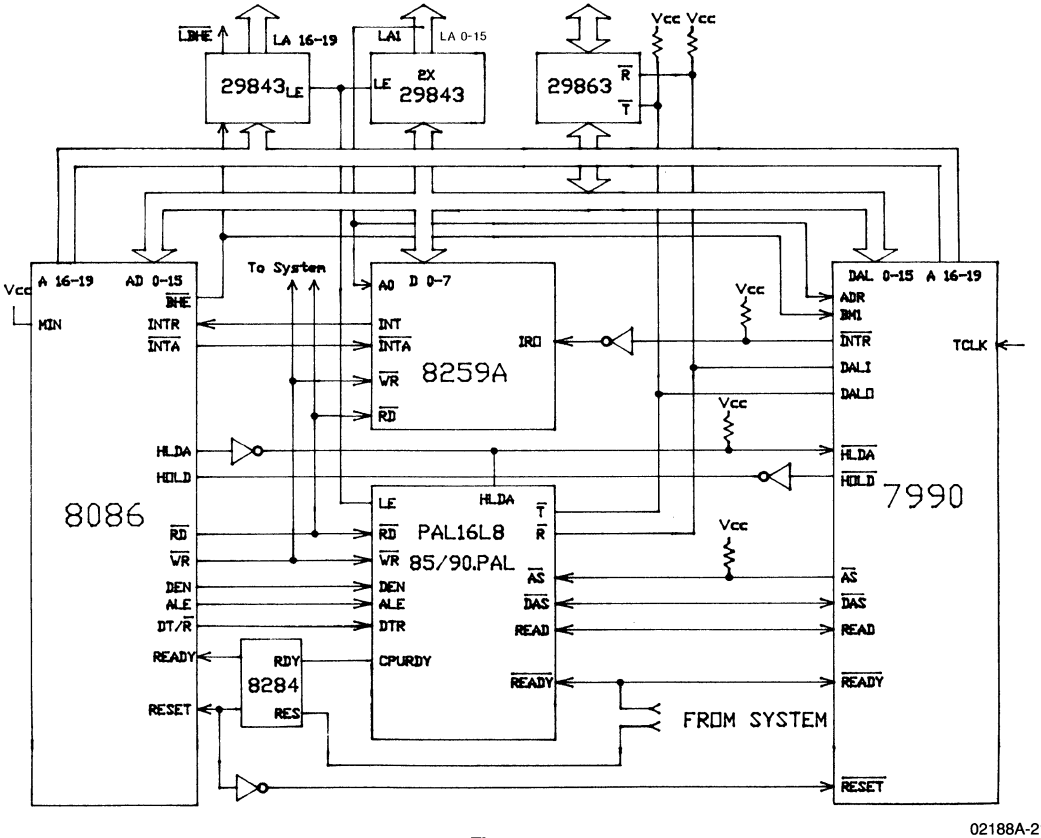


Figure 4-3.1b AmPAL16R4, 8086 (Max. Mode) LANCE Interface Timing Diagram

02188A-20



02188A-21

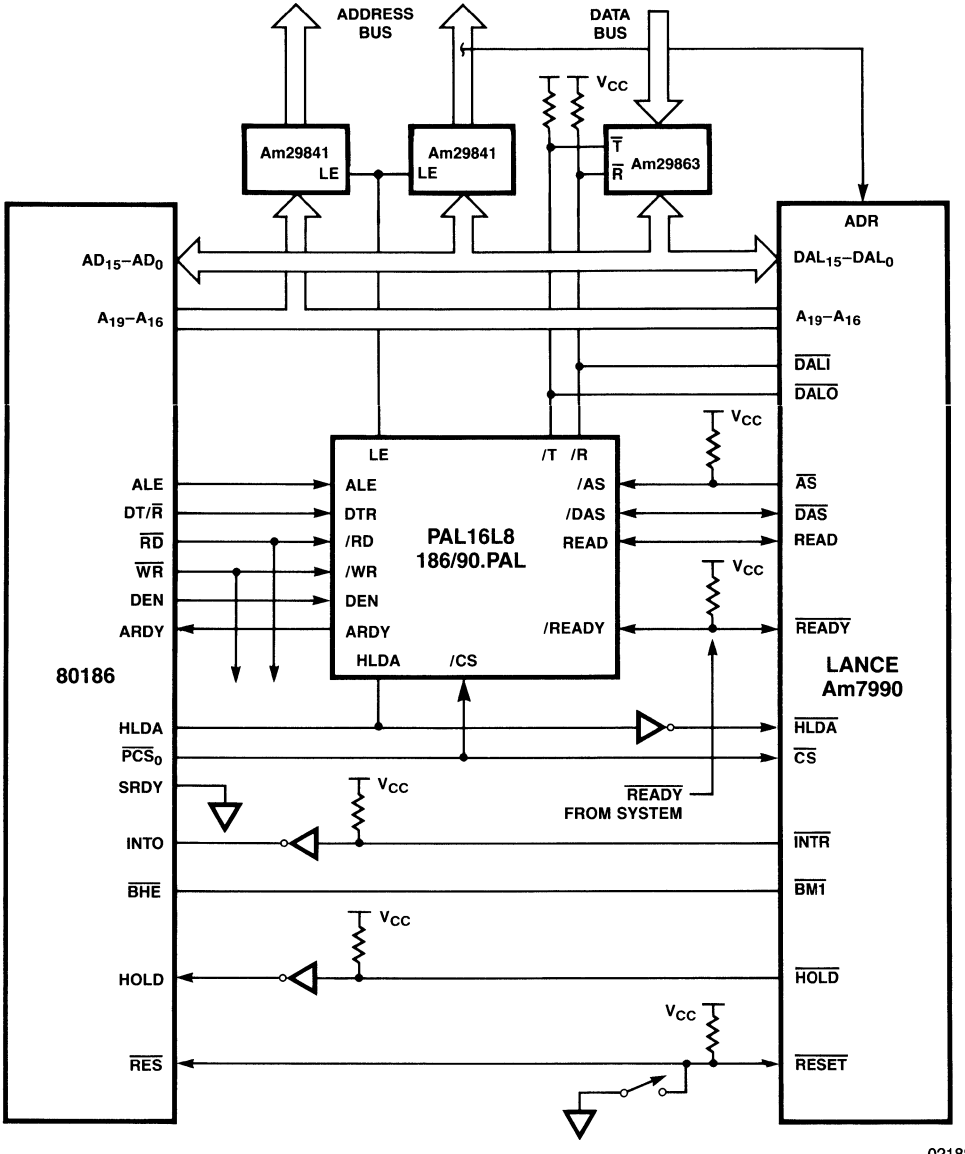
Figure 4-3.1c

```
PAL16L8                                          KHUYNH NGUYEN
PAT
FILE: 86-90-A.PAL                                AUG '85

8086 MINIMUM MODE TO LANCE INTERFACE
AMD

ALE  /AS  DTR  NC  NC  DEN  NC  /READY  HLDA  GND
NC  /DPURDY  READ  /R  /T  /DAS  /WR  /RD  LE  VCC

/LE = /ALE + /AS
/CPURDY = /READY

; CPU (86) IS BUS MASTER

IF(/HLDA) DAS = RD + WR
IF(/HLDA) /READ = DTR
IF(/HLDA) T = DTR
IF(/HLDA) R = /DTR * DEN

; LANCE IS BUS MASTER

IF(HLDA) RD = READ * DAS
IF(HLDA) WR = /READ * DAS
```

Figure 4-3.1d

4.3.2 80186 TO Am7990 LANCE INTERFACE

Similar to the 8086 to Am7990, this interface uses a PAL device design to reduce the parts count. Figure 4-3.2a shows the interface block diagram. 80186/LANCE address and data buses can be connected directly together since they both have multiplexed buses. It seems natural to program the LANCE for ALE output. However, the PAL device equations or indeed a discrete design is easier if $\overline{AS}$ (CSR3, ACON=1) is used. This is because the LANCE tristates ALE, the 186 does not. The $\overline{INTR}$, $\overline{READY}$, and $\overline{HOLD}$ signals from the LANCE are open drain and should be pulled up. The $\overline{BM1}$ signal from the LANCE or $\overline{BHE}$ from the 186 along with A0 can be used to decode the data transfer type (Word/Byte). The external address buffers and data transceivers are enabled by the LANCE and the 186. The buffers and transceivers are enabled by whichever device is the master. The user should program the BCON, BSWP to 0, and ACON to 1 in CSR3.



02188A-22

Figure 4-3.2a 80186 to Am7990 Interface

4.3.3 THE 8086 AND Am8052 CRTC INTERFACE

The 16-bit multiplexed address/data bus of the 8086 is directly connected to the multiplexed address/data lines of the Am8052 (Figure 4-3.3a). The upper address (7 bit for segmented mode or 8 bit for linear mode) is strobed out on the lower half of the bus (AD_{0-7}) and is stored in a register (Am29823). The Am8052 may be programmed for segmented or linear mode depending on whether address roll-over is desired. The register output is enabled ($\overline{OE}$=Low) when the Am8052 is bus master. Clocking is enabled ($\overline{EN}$=Low) when R/$\overline{W}$ is Low while the Am8052 is bus master (upper address update cycle). The trailing edge of Address Strobe clocks the register.

$\overline{RD}$ and $\overline{WR}$ from the 8086 are logically ORed to generate $\overline{DS}$. ALE is inverted and connected to $\overline{AS}$ of the Am8052. DT/$\overline{R}$ is also inverted to form R/$\overline{W}$. All three signals are passed through a three-state buffer which is enabled when the 8086 is bus master. Memory/IO (M/$\overline{IO}$) is pulled High when the Am8052 is bus master since the Am8052 only addresses memory.

BUS CLOCK

The Bus Master timing is synchronized to the bus clock (CLK 1) of the Am8052. In order to get a similar and synchronous bus timing when the 8086 or the Am8052 is driving the bus, the Am8052 bus clock can be connected to the 8086 bus clock. However, in proportional spacing applications, the video timing must be derived from the bus clock and therefore the bus clock must be synchronized to the character clock (CLK 2).

For these applications the Am8152A/53A provides the synchronized clocks (CLK1,CLK2) with the right timing and DC specification.

In non-proportional spacing applications, the Am8052 can operate with the 8086 bus clock if the duty cycle is adjusted. In this case, the Am8152A/53A cannot be used as the clock driver, and a separate clock driver needs to be provided. This clock driver must provide a clock satisfying the special clock input specification (MOS specification) such as clock High and Low width and voltage, and capable of driving the input capacitance of the Am9516A. Most CMOS drivers or a discrete clock driver shown in Figure 4-3.3b satisfies these specifications. This design must be changed for different frequencies. Figure 4-3.3c shows circuitry which adjusts the duty cycle for the Am8052. The required delay time needs to be adjusted for the chosen bus clock frequency.

DETAILED TIMING ANALYSIS

The following timing analysis is based on an 8-MHz 8086-2 and an 8-MHz Am8052. At this frequency the minimum clock High (TCHCL) and Low (TCLCH) times for the 8086-2 become 43 ns and 68 ns, respectively. Some of the subsequent calculations are based on these values for TCHCL and TCLCH. Numbers beside each timing analysis are data sheet parameter numbers.

SLAVE READS AND WRITES

#21 $\overline{CS}$ set-up time to the trailing edge of $\overline{AS}$ (minimum 0 ns). The 8086-2 provides a set-up time of 28 ns of AD_{0-15} before the trailing edge of ALE. Let us assume 0 ns of minimum propagation delay since neither the inverter nor the driver specifies one. The maximum propagation delay allowed for the decoder is, therefore, 28 ns (68 ns - 40 ns). The decode time for the Am29806/809 decoders is 13 ns.

#22 $\overline{CS}$ hold time after the trailing edge of $\overline{AS}$ (minimum 25 ns). The 8086-2 provides a minimum address hold time of 33 ns.

#23 C/$\overline{D}$ set-up time before the trailing edge of $\overline{AS}$ (minimum 0 ns). The 8086-2 provides an address set-up time of 28 ns.

#24 C/$\overline{D}$ hold time after the trailing edge of $\overline{AS}$ (minimum 25 ns). The 8086-2 provides a minimum address hold time of 33 ns.

#25 Delay from $\overline{CS}$ to $\overline{DS}$ (minimum 30 ns). The worst case (shortest delay) can be calculated as:

(TCLCH - TCHLL) + TCLRL + (28 ns - 13 ns) = (68 ns - 55 ns) + 10 ns + (28 ns - 13 ns) = 37 ns.

#26 Access time (maximum 150 ns). The 8086-2 expects an I/O access time no longer than:

2 * TCLCL - TCLRL - TDVCL =
2 * 125 ns - 100 ns - 20 ns = 130 ns.

This means that one Wait State must be inserted.

#27 Data hold time (minimum 10 ns). The 8086-2 requires a max data hold time of 0 ns, i.e. no hold time.

#28 + 29 R/$\overline{W}$ $\overline{DS}$. Since DT/$\overline{R}$ is connected to the R/$\overline{W}$ input of the CRTC, this timing is not guaranteed by design.

#32 Data hold time during slave writes (minimum 20 ns). The 8086-2 provides at least 38 ns.

#33 Data set-up time in slave writes (minimum 90 ns). The 8086-2 provides more than one clock period (125 ns) data set-up time.

#34 The Am8052 requires a minimum Data Strobe pulse width of 100 ns. The 8086-2 provides

TWLWH = 2 * TCLCL - 40 ns = 210 ns.

#35 Recovery time (minimum 330 ns). The 8086-2 provides more than 3 clock periods = 375 ns.

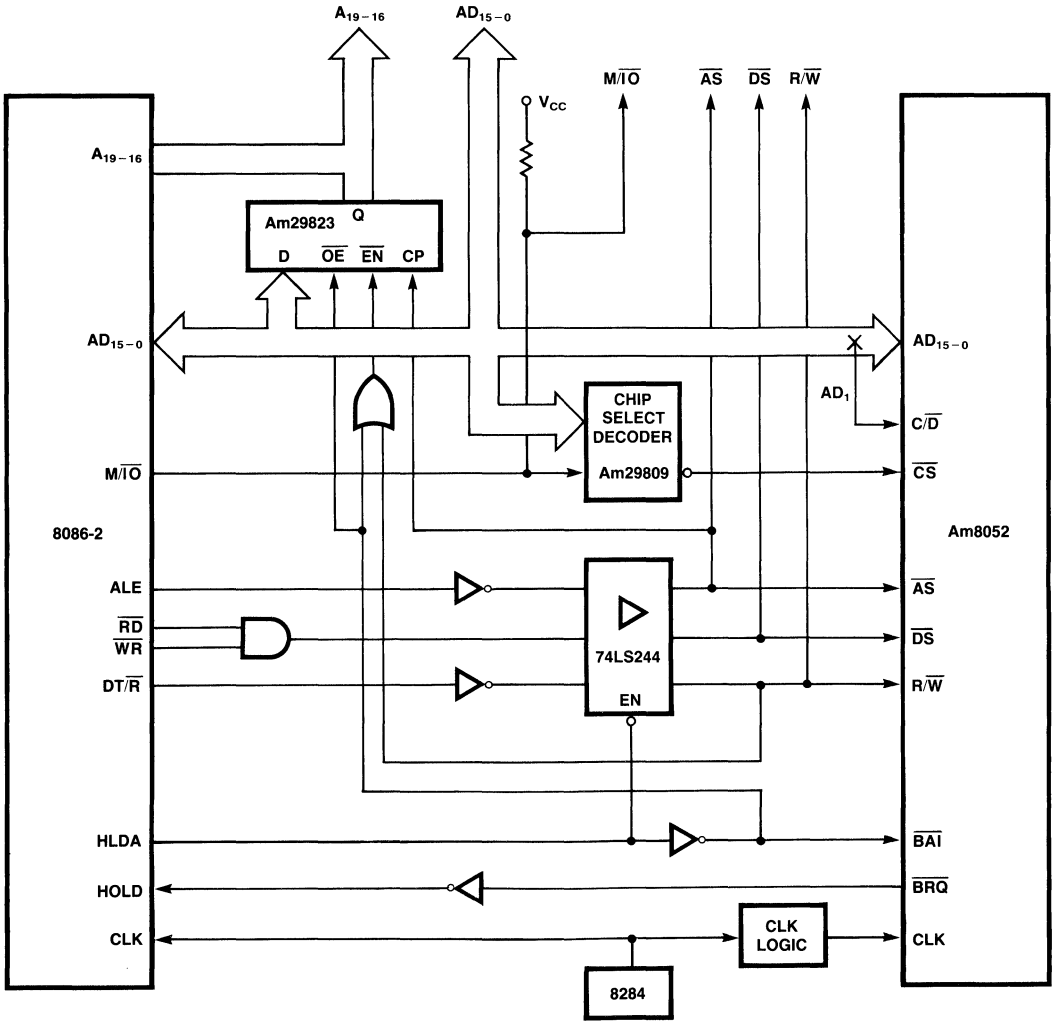
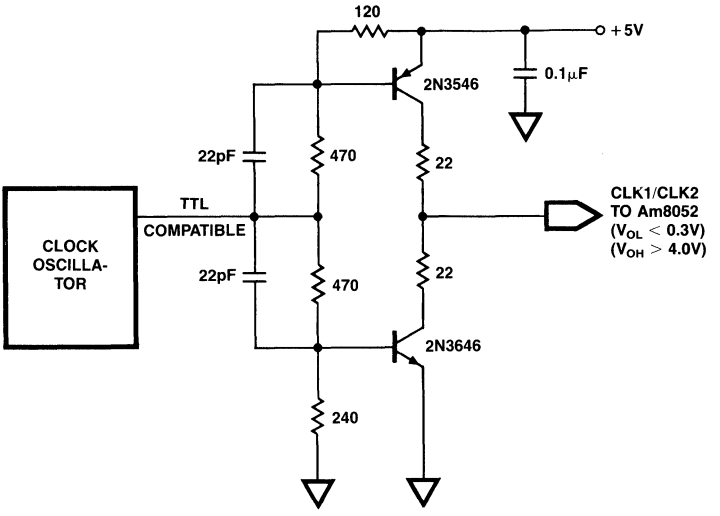
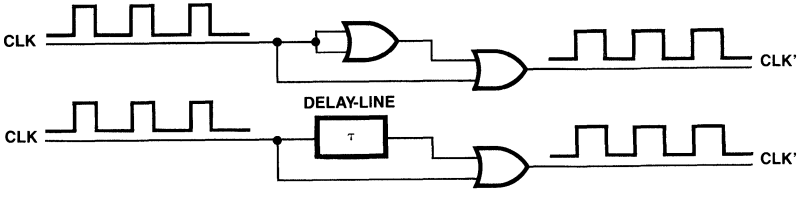


Figure 4-3.3a 8086–Am8052 Interface



02188A-24

Figure 4-3.3b



02188A-25

Figure 4-3.3c Duty Cycle Adjustment for the Am8052

4.3.4 iAPX186 TO AmZ8068 DATA CIPHERING PROCESSOR INTERFACE

The iAPX186 can operate in two basic modes Minimum Mode or Maximum Mode.

In Maximum Mode the 8288 Bus Controller provides command and control timing.

In Minimum Mode the bus timing of the iAPX186 is slightly different from the 8086 bus timing. Figure 4-3.4a shows the interface logic. The maximum clock rate for the DCP is 4 MHz, resulting in a maximum CPU clock rate of 8 MHz. No Wait States are required.

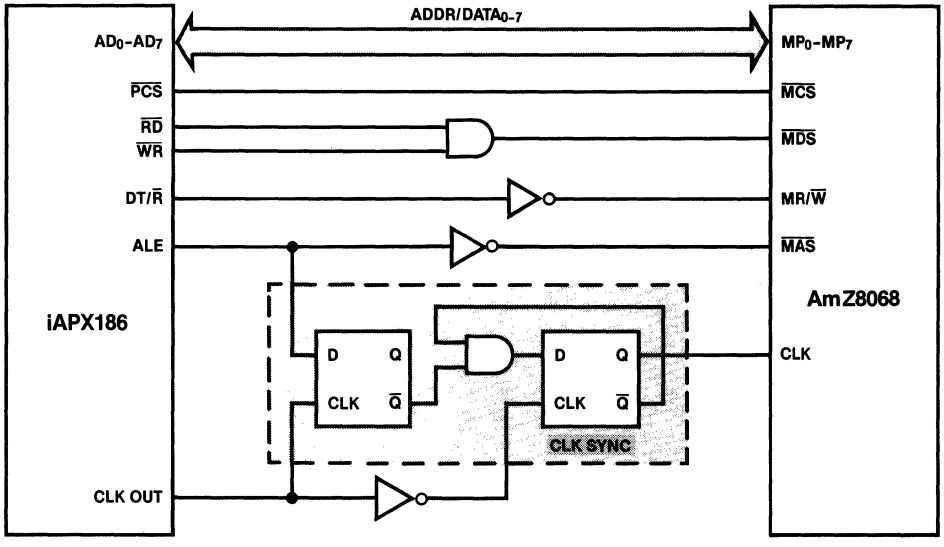
An AmZ8068 must be used in this application because of the wider range in delay time from clock to the read or write control signal delay with respect to the clock. This parameter is specified for the iAPX186 as 10 to 55 ns. The AmZ8068 requires a delay of 0 to 50 ns at 4 MHz, the Am9568 0 to 30 ns at 4 MHz. Because of two delays in the clock path (Inverter and D-Flip-Flop) and only one delay in the control signal path (AND gate), the timing tolerance of these signals at the DCP is decreased to 0 to 45 ns.

At lower CPU clock rates the timing is less critical because the specified time relationship between clock and data strobe becomes wider (timing parameter 45 of the data sheet).

The maximum clock for operating without a Wait State can be calculated like this: The $\overline{RD}$ width is specified as 2 * TCLCL - 50 ns for the iAPX186. The $\overline{WR}$ width is 2 * TCLCL - 40 ns. The smaller $\overline{RD}$ width is used for the calculation. At an 8-MHz clock, the 186 generates an $\overline{RD}$ signal 200 ns wide. The AmZ8068 requires a minimum data strobe width of 200 ns for a Status Register access. The system can, therefore, operate up to this clock rate without a Wait State.

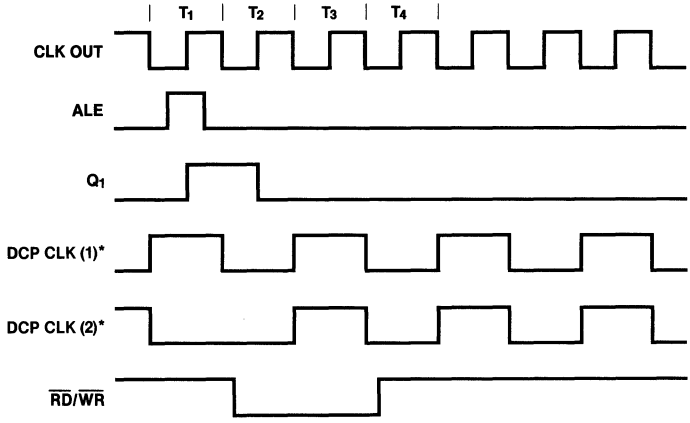
The Clock Synchronizer used is shown in Figure 4-3.4a. Figure 4-3.4b illustrates how this logic synchronizes the data strobe to the clock. DCP CLK(1) and DCP CLK(2) show the possible phases of the CPU clock before synchronization. At the end of cycle T1 the clock is synchronized. No Wait State is allowed when accessing the DCP. (An odd number of Wait States would synchronize the data strobe to the wrong edge of the clock.)

See the *Am9518, Am9568, AmZ8068 DCP Manual* for more on this part.



02188A-26

Figure 4-3.4a iAPX186–DCP Interface (Minimum Mode)



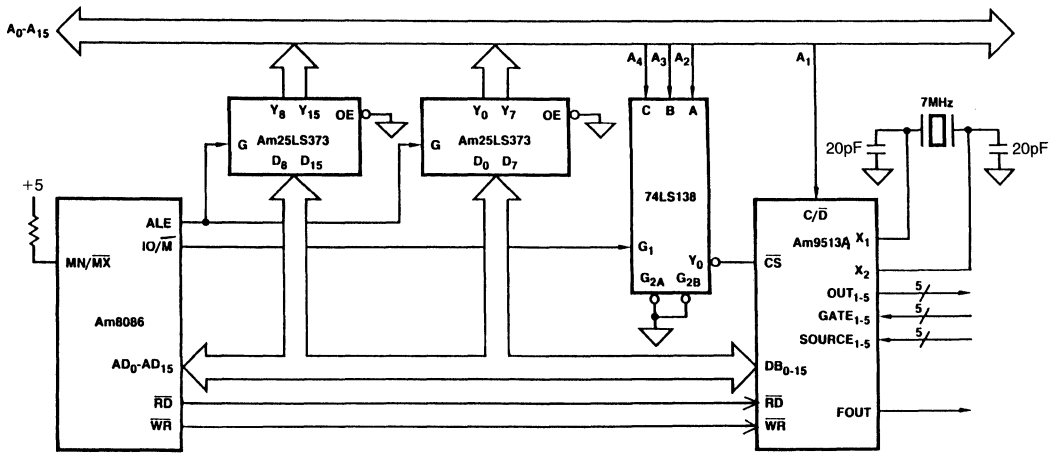
*DCP CLK (1) AND (2) SHOW TWO PHASES OF DCP CLK

02188A-27

Figure 4.3.4b DCP CLK Synchronization Timing (No Wait States)

4.3.5 THE 8086 AND Am9513A SYSTEM TIMING CONTROLLER INTERFACE

This interface is straightforward. The diagram in Figure 4-3.5 shows that data is transmitted directly between the processor and the Am9513A. The 25LS373's latch the address decoded by the 74LS138.



02188A-28

Figure 4-3.5

4.3.6 THE 8086 AND Am9516 UNIVERSAL DMA CONTROLLER INTERFACE

Am9516 IN MIN. MODE

Figure 4-3.6a illustrates the interface of the Am9516 to the 8086 in the MIN. mode configuration. Figure 4-3.6c illustrates the interface in MAX. mode. The interfaces could be accomplished by using rather complex implementation of standard SSI/MSI logics. Examples here replace the logic portion with a PAL device. The MIN. Mode uses the PAL device 16L8. This is a good example of "GARBAGE COLLECTION". It reduces the amount of real estate, interconnections, parts, and part types.

Both interface examples accomplish two major functions. First, when the Am9516 is bus master, it converts $\overline{RD}$ and $\overline{WR}$ into R/$\overline{W}$ and $\overline{DS}$, and vice versa when the Am9516 is not the bus master. Secondly the buffer controls, $\overline{TBEN}$ and $\overline{REBN}$, are generated from DEN and DT/$\overline{R}$.

The two examples show different types of latches and transceivers and there are many more to choose from. The designer selects those that best meet system requirements while trying to minimize the number of different parts that must be stocked.

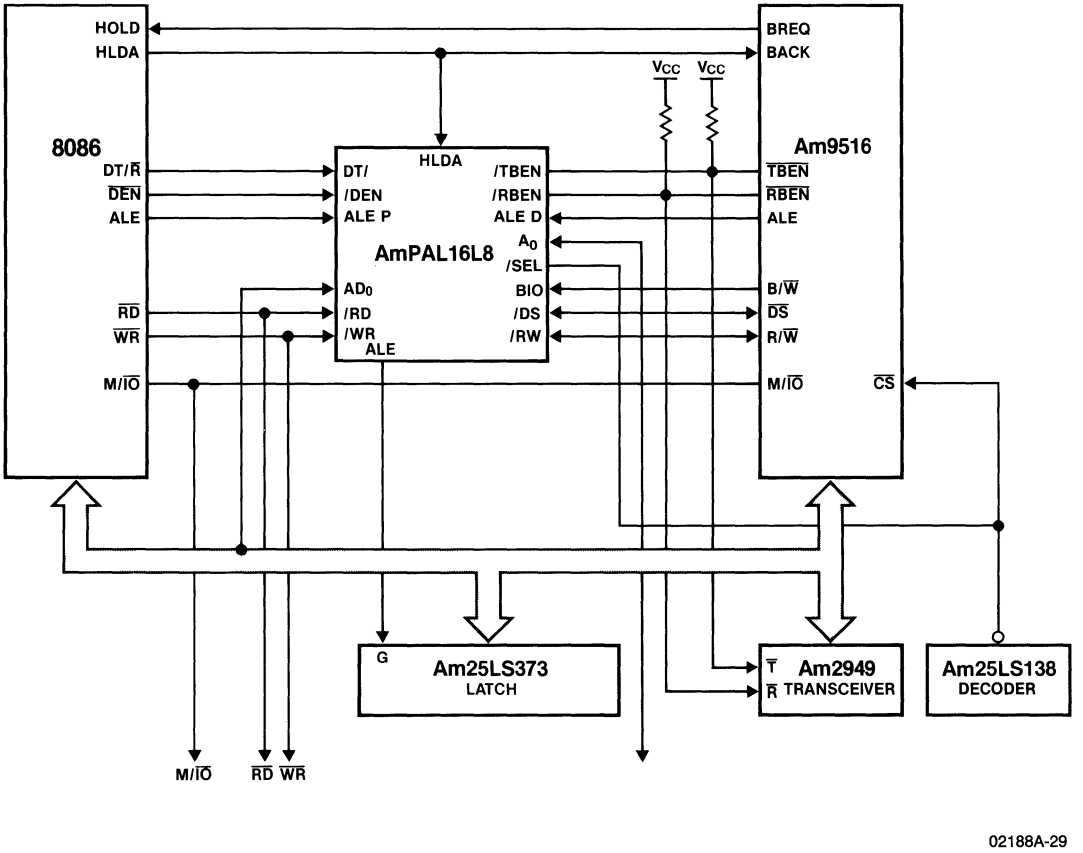


Figure 4-3.6a The Am9516 UDC to 8086 CPU Interface (Minimum Mode)

```
AmPAL16L8 PALASM FILE

PAL16L8
PAT001
Am9516 to 8086 min mode interface chip
Advanced Micro Devices

NC ALED ALEP HLDA BW AD0 DT /DEN /SEL GND
NC /RBEN /RD ALE A0 /RW /DS /WR /TBEN VCC

IF (/HLDA) DS = RD + WR
IF (/HLDA) RW = DT
IF (/HLDA) TBEN = /DT * /SEL * DEN
IF (/HLDA) RBEN = DT * /SEL * DEN
IF (HLDA) RD = /RW * DS
IF (HLDA) WR = RW * DS
ALE = /ALEP * /ALED
A0 = /AD0 * /BW * HLDA * ALED       +
     AD0 * BW * HLDA * ALED         +
     /AD0 * /HLDA * ALEP            +
     A0 * /ALEP + A0 * /ALED

DESCRIPTION

THIS PAL CONVERTS THE CONTROL SIGNALS TO INTERFACE THE 8086 IN
MIN MODE TO THE Am9516 DMA CONTROLLER.  ANOTHER EXAMPLE SHOWS HOW
THIS IS DONE IN MAX MODE.
```

Figure 4-3.6b

Am9516 IN MAX. MODE

This MAX. mode interface between the 8086 and Am9516 (Figure 4-3.6c) also uses a PAL device to reduce component count.

The example makes several assumptions which result in a slightly more complex design than absolutely necessary. The 8086 is assumed to be in MAX. Mode, and the design has to be compatible with a Multibus or similar interface; the 16R4 and 8288 could be eliminated if this is not the case.

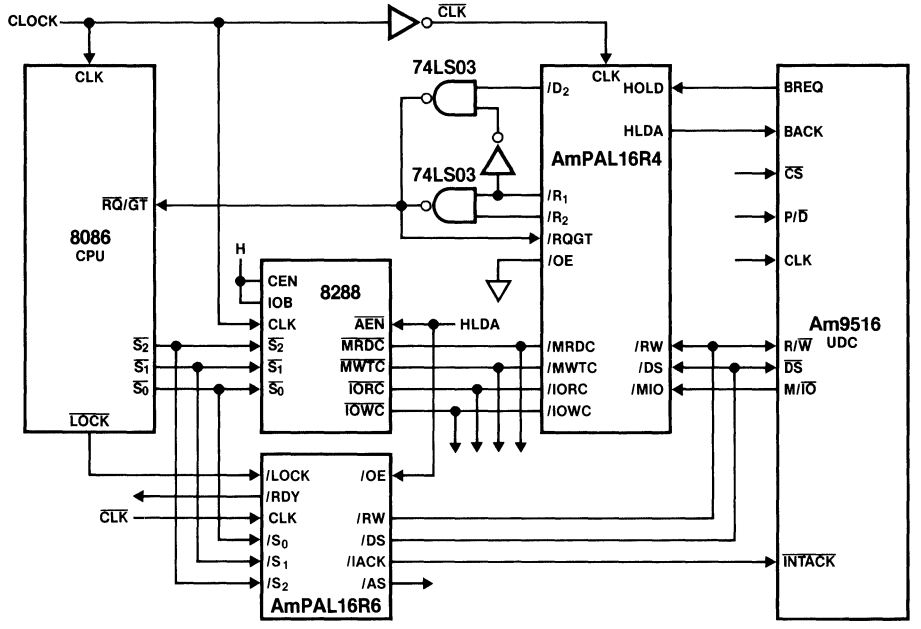
The 16R6 (Figure 4-3.6c) is a PAL device that performs a function similar to the 8288, that is, it converts the processor's status signals and clock into control signals. In this case, the signals are, R/$\overline{W}$, and Data Strobe ($\overline{DS}$), Interrupt Acknowledge ($\overline{INTA}$) and Peripheral Acknowledge ($\overline{PACK}$). This PAL device, Figure 4-3.6e, basically generates ZBUS signals for Zilog peripherals. In the example, it connects to the Am9516 (UDC). The DMA Controller is very similar to the AmZ8016 except its bus interface has been modified to interface to non-multiplexed buses. This was changed from the previous design using a 16R8 to eliminate problems due to skew between OSC and CLK.

The 16R4, Figure 4-3.6d, has two functions. The first is connecting the MIN. Mode protocol of the Am9516 or similar device to the MAX. Mode protocol for bus exchange. The 74LS03s are used to aid in this function. The timing waveform illustrates what happens in detail. The basic philosophy is that a rising edge on the HOLD input generates a pulse that is one clock wide. The CPU samples this pulse and, in response, issues a one-clock-wide pulse. The 16R4 uses this response pulse to generate HLDA. When the Am9516 has completed the necessary transfer, HOLD transitions High to Low. This generates another pulse to the CPU signalling that the Am9516 is done and that the CPU may continue.

The second function of the 16R4 is the conversion of R/$\overline{W}$, $\overline{DS}$ and M/$\overline{IO}$ into the Multibus compatible signals /MRDC, /MWTC, /IORC, and /IOWC, when HLDA is High. It is possible to collapse the 74LS03s into the PAL device, thus reducing the external logic required to only one open collector inverter. The disadvantage, however, is that it adds two additional clocks for the bus exchange overhead, one during acquisition of the bus and one on bus release. For block transfers, this is not significant but it may be undesirable when performing single transfers, or short burst, or when in Demand Mode. To eliminate the 74LS03, change the equation for R2 to

$$/R1 * /D2 + R1 * D2,$$

and then drive the /[RQ/GT] line with a 74LS05 from the R2 output.



02188A-30

Figure 4-3.6c

```
AmPAL16R4 PALASM File

B> Type Am9516 PAL
PAL16R4

8086 to Am9516 interface
Advanced Micro Devices

CLK /RQGT HOLD   NC   NC  NC /RW /DS    MIO  GND
/OE /MWTC /MRDC HLDA /D2 /R2 /R1 /IOWC /IORC VCC

IF (HLDA) IORC = /MIO * DS * /RW
IF (HLDA) IOWC = /MIO * DS * RW
IF (HLDA) MRDC = MIO * DS * /RW
IF (HLDA) MWTC = MIO * DS * RW

R1 := HOLD

R2 := /R1

D2 := R1

/HLDA := /R1 + /D2 * /HLDA +
         /RQGT * /HLDA

DESCRIPTION

THIS DEVICE CONVERTS THE MIN MODE SIGNALS HOLD AND HLDA TO THE
MAX MODE /RQGT PROTOCOL.  ADDITIONALLY IT GENERATES THE 8288
EQUIVALENT CONTROL OUTPUTS /MRDC, /MWTC, /IORC, AND /IOWC.  THIS
PAL WAS USED TO CONNECT THE Am9516 TO THE 8086 IN MAX MODE.
```

Figure 4-3.6d

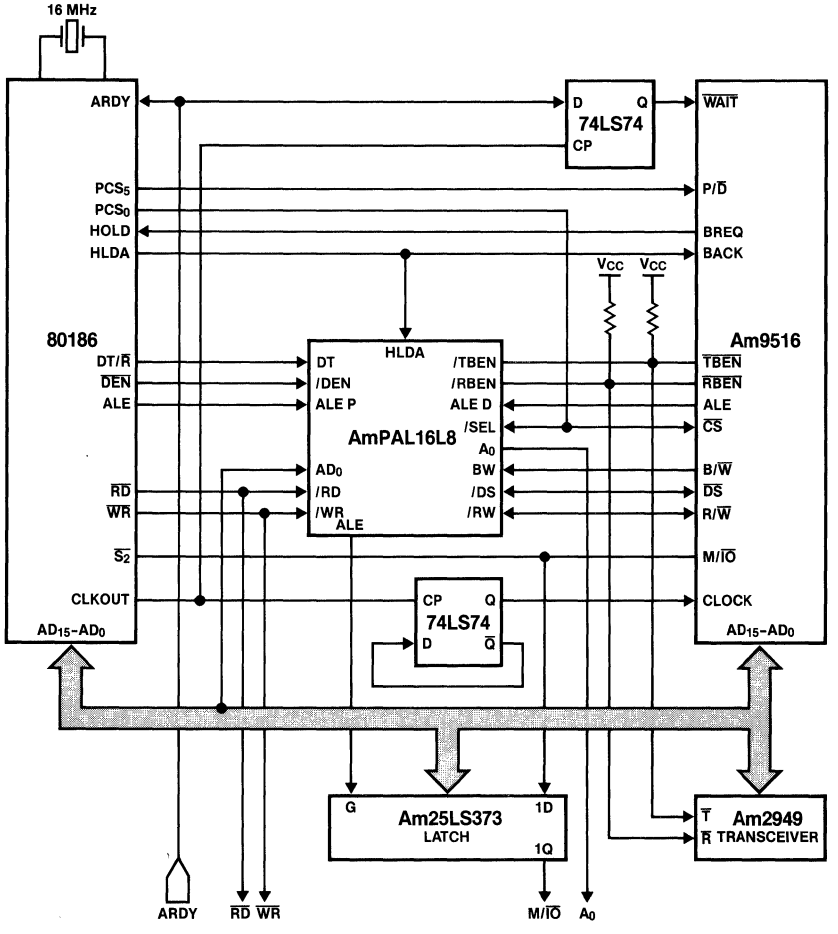
```
AmPAL16R6                                PAL DESIGN SPECIFICATION
PAT 005                                     BY JOE BRCICH 5/10/83
8086 TO 85XX PERIPHERAL INTERFACE  & JAMES WILLIAMSON 7/21/83
ADVANCED MICRO DEVICES

CLOCK   RESET  CLK    /S0    /S1    /S2   /LOCK    NC    NC    GND
/OE     /AS    /P1    /RW    /DS    /P0   /IACK   /RDY  CLKD   VCC

P0 := /RESET * S0 * /P0 * /P1                                  +
      /RESET * S1 * /P0 * /P1                                  +
      /RESET * S2 * /P0 * /P1                                  +
      /RESET * S0 * P1                                         +
      /RESET * S1 * P1                                         +
      /RESET * S2 * P1

P1 := /RESET * P0 * /P1                                        +
      /RESET * P1 * S0                                         +
      /RESET * P1 * S1                                         +
      /RESET * P1 * S2

DS := /IACK * /P0 * P1 *  S0 * /S1 * S2                        +
      /IACK * /P0 * P1 * /S0 *  S1 * S2                        +
       IACK *  S0 * S1 * S2 * P0 * /P1 * LOCK                  +
       DS * SO  *  S1 * S2
       DS * SO  * /S1 * S2
       DS * /SO *  S1 * S2

RW := S0 * /S1 * S2

IACK := /RESET * S0 * S1 * S2 * /P0 * /P1 * /LOCK
        /RESET * IACK * S0 * S1 * S2 * P0 * /P1 * /LOCK
        /RESET * IACK * LOCK  * /DS
        /RESET * IACK * /LOCK *  DS * /P0 * P1

RDY := /RESET *  S0  * /S1 * S2 * P0 * P1                      +
       /RESET * /S0  *  S1 * S2 * P0 * P1                      +
       /RESET * RDY  *  S0 * /S1 * S2                          +
       /RESET * RDY  * /S0 * S1 * S2                           +
       /RESET * IACK *  S0 * S1 * S2 * DS                      +
       /RESET * RDY  *  S0 * S1 * S2

/CLKD = CLK

AS = /CLKD * P0 * /P1 * /IACK * CLK


DESCRIPTION
THIS PAL TRANSLATES 8086 BUS SIGNALS INTO COMPATIBLE SIGNALS
FOR THE 9516. IT IS ALSO APPLICABLE TO 85XX PERIPHERALS BY
ALTERING /RW AND /DS TO /RD AND /WR. ONE FLIP FLOP IS AVAILABLE
TO GIVE THE NECESSARY DELAY TO THE FALLING EDGE OF /WR.  THE DATA
STROBE TIMING FOR A WRITE CYCLE IS DELAYED UNTIL THE FALLING EDGE
OF T2 TO MEET THE REQUIREMENTS OF THE 85XX PARTS.  THIS DESIGN
ASSERTS RDY TO DEMAND ONE WAIT STATE FROM THE 8086.  THIS WAIT
STATE IS NOT LONG ENOUGH FOR DESIGNS WHICH USE AN 8 MHz 8086.
THEREFORE, WITH AN 8 MHz CPU, 85XXA PERIPHERALS SHOULD BE USED.
AS AN ALTERNATIVE, THREE WAIT STATES CAN BE USED BY ALTERING THE
RDY EQUATION.  THIS PAL ALSO TRANSFORMS THE 8086 TWO CYCLE
INTERRUPT ACKNOWLEDGE INTO A SINGLE CYCLE OF THE TYPE NECESSARY
FOR 85XX PARTS.  THIS IS MADE POSSIBLE BY SAMPLING THE LOCK
STATUS, P0, P1, AND IACK SIGNALS.
------------------

NOTE;  THE CLK SIGNAL MUST BE EXTERNALLY INVERTED.
```

Figure 4-3.6e

4.3.7 80186 TO Am9516 UNIVERSAL DMA CONTROLLER

The addition of the Am9516 to an 80186 design is a natural choice in systems requiring four channels of DMA. Figure 4-3.7a shows the interface between the 80186 and the Am9516 with a PAL device. PCS_5 is programmed to provide a latched A_1 signal.

This interface accomplishes two major control transformations. First, it converts $\overline{RD}$ and $\overline{WR}$ into R/W and $\overline{DS}$ when the 80186 is Bus Master, and vice versa when the Am9516 is Bus Master. Secondly, the transceiver control signals, $\overline{TBEN}$ and $\overline{RBEN}$, are generated from DEN and DT/R. This example shows only one possible configuration. Other configurations can be made as dictated by system requirements. A PAL device is used here to reduce board space. MSI and SSI could also be used.



02188A-31

Figure 4-3.7a

```
AmPAL16L8 PALASM File

PAL16L8
PAT001
Am9516 to 80186 interface chip
Advanced Micro Devices

NC   ALED   ALEP   HLDA   BW   AD0    DT   /DEN   /SEL   GND
NC   /RBEN  /RD    ALE    A0   /RW   /DS   /WR    /TBEN  VCC

IF (/HLDA) DS = RD + WR
IF (/HLDA) RW = DT
IF (/HLDA) TBEN = /DT * /SEL * DEN
IF (/HLDA) RBEN = DT * /SEL * DEN
IF (HLDA) RD = /RW * DS
IF (HLDA) WR = RW * DS
ALE = /ALEP * /ALED
A0 = /AD0 * /BW * HLDA * ALED +
     AD0 * BW * HLDA * ALED   +
     /AD0 * /HLDA * ALEP      +
     A0 * /ALEP               +
     A0 * /ALED

DESCRIPTION

THIS PAL CONVERTS THE CONTROL SIGNALS TO INTERFACE THE 80186 TO
THE Am9516 DMA CONTROLLER.  OTHER EXAMPLES SHOW HOW THIS IS DONE
FOR THE 8086 IN MIN AND MAX MODES.
```

Figure 4-3.7b

4.3.8 8086/8088 TO Am9518/AmZ8068/Am9568 INTERFACE

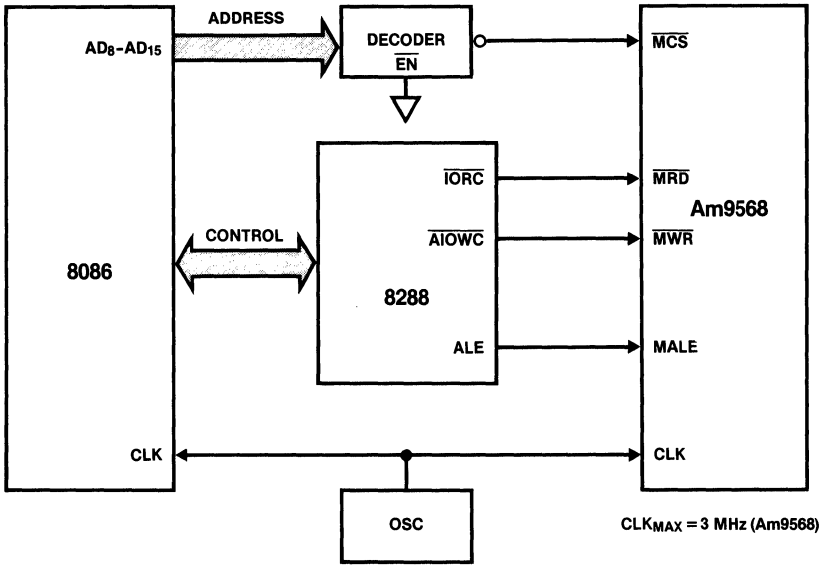
Interfacing the DCP family to 8086 or its 8-bit bus equivalent, the 8088, is straightforward. In systems with CPU clock rates up to 3 MHz, the Am9568 can be directly interfaced to the CPU (Figures 4-3.8a and 4-3.8b). The clock rate is limited to 3 MHz because of the 33%/66% duty cycle (33% High, 66% Low) of the CPU clock and to satisfy the minimum clock High time of 115 ns of the Am9568. The second critical parameter is the relationship between the clock and Data Strobe. The Am9568 requires a delay of the rising edge of $\overline{MRD}$ or $\overline{MWR}$ to the falling edge of the clock of 0 - TWL - 85 ns. TWL is the clock Low width. In this interface the minimum clock Low width is 207 ns. This determines a maximum delay of up to 122 ns. The CPU is specified to have a "Control Active Delay" of 10 to 110 ns. With a margin of 12 ns, it is obviously impossible to increase the system clock by modifying its duty cycle.

Figures 4-3.8c and 4-3.8d show a similar interface using the Am9518 and the AmZ8068. This interface needs additional logic to convert the Read or Write strobes into a Read/Write (R/$\overline{W}$) and a Data Strobe ($\overline{MDS}$) and to invert the Address Latch Enable to generate a Master Port Address Strobe ($\overline{MAS}$). Similar to the interface discussed above, the clock rate is limited by the clock Low and High widths and the requirements of the DCP. The Am9518 needs a minimum clock High width of 150 ns determining a maximum clock rate of 2.3 MHz. The minimum clock Low width of 275 ns and the DCP specification of 0 - TWL - 100 ns provides a margin of 275 ns - 110 ns - 100 ns = 65 ns.

The AmZ8068 requires a minimum clock High width of 115 ns, resulting in the same maximum clock rate as in interfacing to the Am9568 (3 MHz). The specification about the synchronization of clock and data strobe is less critical in this interface (0 - TWL - 65 ns) so the margin becomes 32 ns.

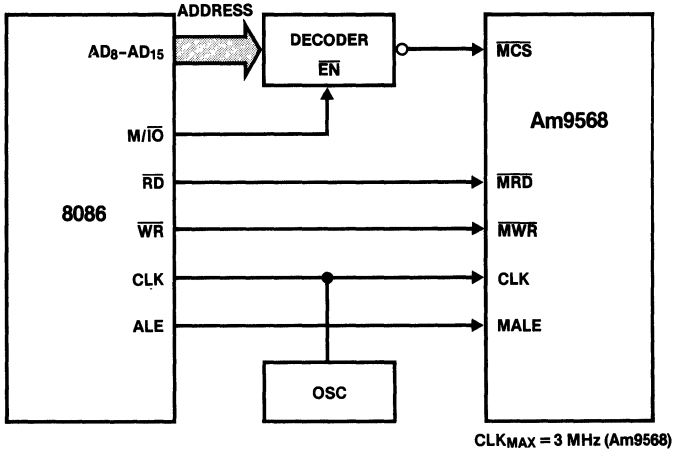
An 8086/8088 system with clock rates larger than the rates mentioned above requires more sophisticated interface logic. The DCP clock must not exceed 4 MHz (3 MHz for the Am9518), the Address Strobe width has to be satisfied, and the Data Strobes must be synchronous to the clock. The case in which the DCP clock is divided down by two from the CPU clock is discussed below.

An application where the DCP runs asynchronously from the 8086 clock is not discussed here. Ideas can be taken from the chapter on iSBX Bus to Am9568 interface.



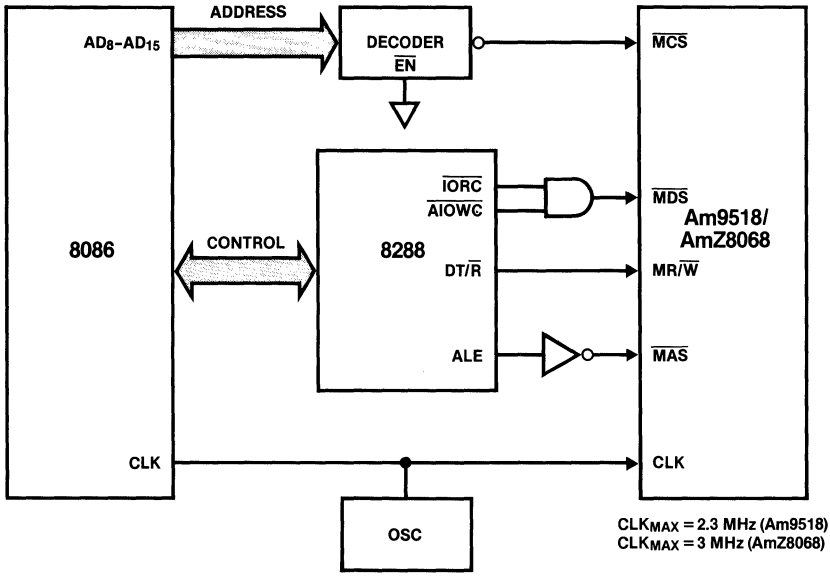
02188A-32

Figure 4-3.8a Direct Interface 8086-Am9568 (Maximum Mode)



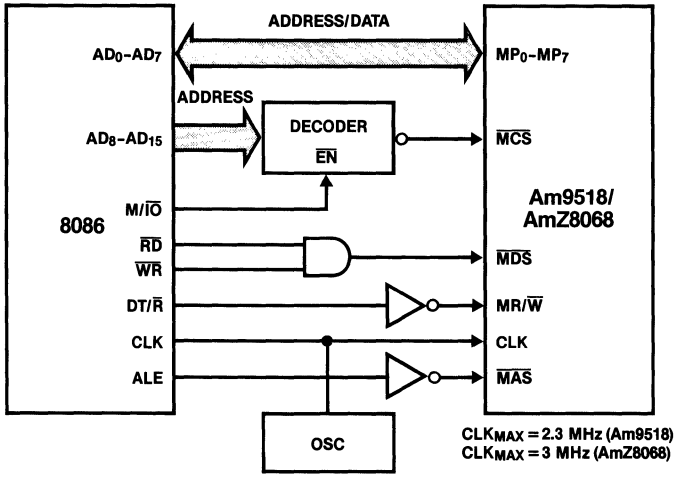
02188A-33

Figure 4-3.8b Direct Interface 8086-Am9568 (Minimum Mode)



02188A-34

Figure 4-3.8c Direct Interface 8086-Am9518/AmZ8068 (Maximum Mode)



02188A-35

Figure 4-3.8d Direct Interface 8086-Am9518/AmZ8068 (Minimum Mode)

8086/8088 - Am9518/AmZ8068 (FIGURES 4-3.8e AND 4-3.8f)

The Control/Key Mode input (C/$\overline{K}$) is wired Low to select the Multiplexed Control Mode. In this mode the address to the internal registers of the DCP, MP_1 and MP_2, is multiplexed with the data byte on the eight bidirectional lines of the Master Port bus. MP_1 and MP_2 are latched on the rising edge of $\overline{MAS}$ (Master Port Address Strobe), to select the internal register for subsequent data transfer cycles.

$\overline{MAS}$ is the inverted Address Latch Enable of the 8086 bus. The state of $\overline{MCS}$ (Master Port Chip Select) is aso latched at the rising edge of $\overline{MAS}$. In the Minimum Mode of the 8086 (MN/$\overline{MX}$=High) $\overline{MCS}$ may only go Low during Input/Output cycles (M/$\overline{IO}$=Low); therefore, M/$\overline{IO}$ enables the address decoder in Minimum Mode.

The Read/Write input (MR/$\overline{W}$) is connected to Data Transmit/Receive (DT/$\overline{R}$). DT/$\overline{R}$ satisfies the setup and hold time requirements of MR/$\overline{W}$.

Master Port Data Strobe ($\overline{MDS}$) is active if either Input/Output Read Control ($\overline{IORC}$) or Advanced Input/Output Write Control ($\overline{AIOWC}$) are active. The $\overline{AIOWC}$ has a wider Low width than $\overline{IOWC}$ (Input/Output Write Control) and so gives a wider margin in interfacing.

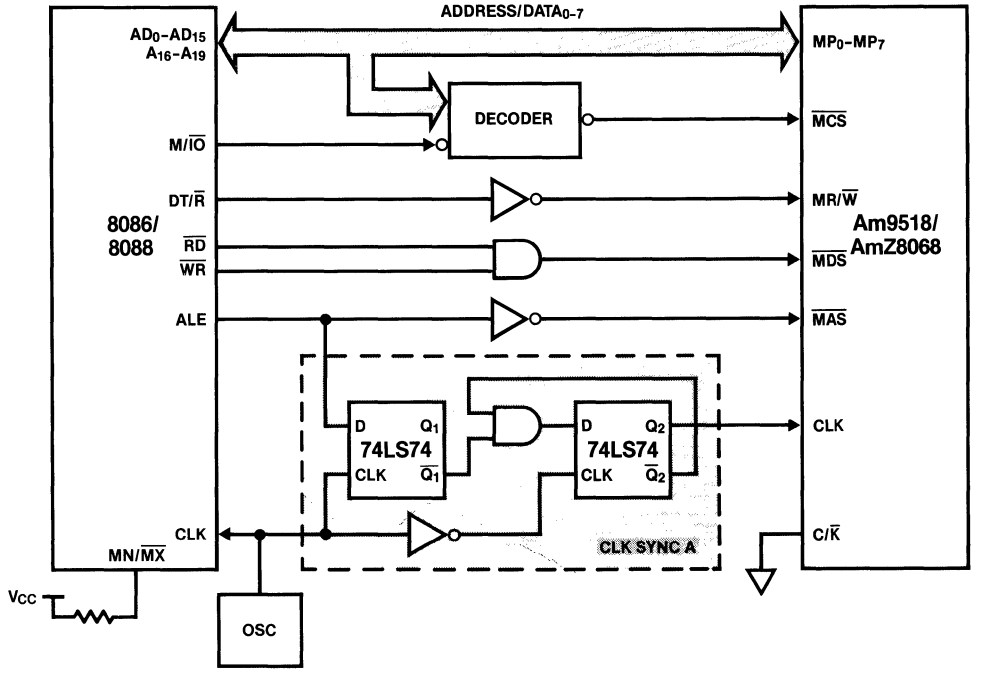
In Minimum Mode (Figure 4-3.8e), $\overline{RD}$ and $\overline{WR}$ are logical ORed to generate $\overline{MDS}$. The timing is the same as in Maximum Mode.

8086/8088 - Am9568 (FIGURE 4-3.8a)

CPU clock rates above 4.44 MHz (above 5.8 MHz for the AmZ8068) require use of the Am9568 instead of the Am9518, because TWA (Master Port Address Strobe width) becomes critical with increased clock rate, as shown below:

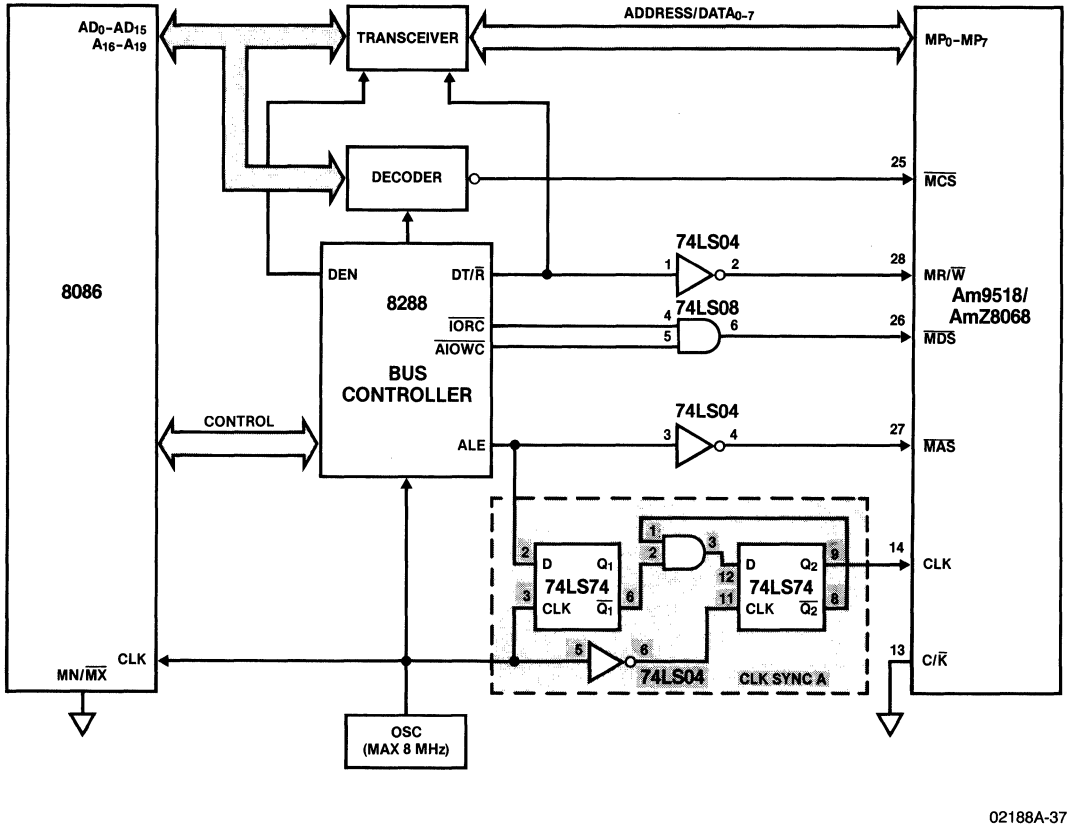
Am9518: TWA = 115 ns
AmZ8068: TWA = 80 ns
Am9568: TWA = 40 ns

8086/8088: TLHLL = 115 ns at 4.44 MHz
8086/8088: TLHLL = 80 ns at 5.80 MHz
8086/8088: TLHLL = 48 ns at 8.00 MHz



02188A-36

Figure 4-3.8e 8086/8088-Am9518/AmZ8068 Interface (Minimum Mode)



02188A-37

Figure 4-3.8f 8086/8088-Am9518/AmZ8068 Interface (No Wait State)

TLHLL is the Address Latch Enable width (ALE) of the 8086.

For CPU clock rates above 7 MHz, one Wait State has to be inserted during Control Register Reads (timing parameter 44). The interface block diagram is shown in Figure 4-3.8g.

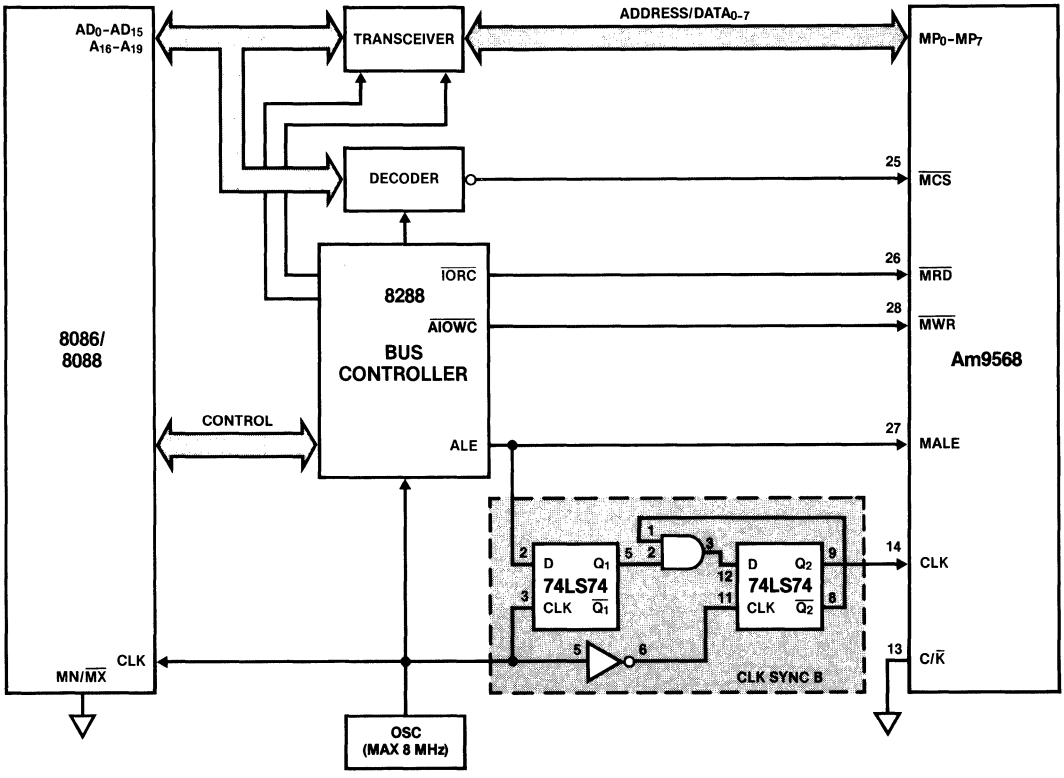
NOTE

In the interfaces shown, the number of Wait States must be the same for all read or write accesses to the DCP, because the Clock Synchronizer is designed for either an even or an odd number of Wait States.

CLOCK SYNCHRONIZATION

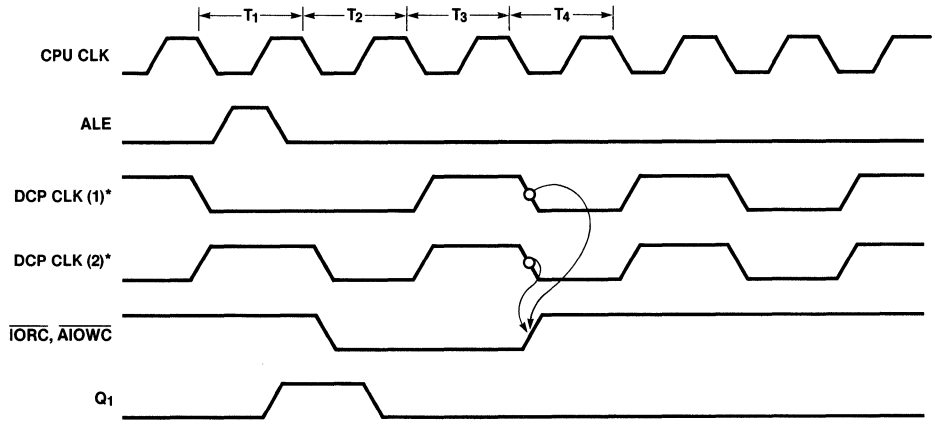
A very important factor in designing the interface to the 8086 is that the rising edge of $\overline{MDS}$ must be synchronous to the falling edge of the DCP clock (timing parameter 45).

In a system where the DCP runs at a divided system clock, a clock synchronizer is required. Without a synchronizer the rising edge of the Data Strobes ($\overline{MDS}$, $\overline{MRD}$ and $\overline{MWR}$) would be synchronous to either the falling or rising edge of the divided clock. Two simple Clock Synchronizers are used in these interfaces, one is designed for an even number, the other is designed for an odd number of Wait states. The DCP clock is synchronized to the Data Strobes at the falling edge of the CPU clock at the end of the CPU cycle T1 (Figures 4-3.8h and 4-3.8j). At this edge, the state of the DCP clock is forced to a Low (CLK SYNC A in Figure 4-3.8h) or to a High (CLK SYNC B in Figure 4-3.8j), depending on the number of Wait states inserted. DCP CLK 1 and 2 show the two possible phases of the DCP clock and how the Clock Synchronizer adjusts the phase.



02188A-38

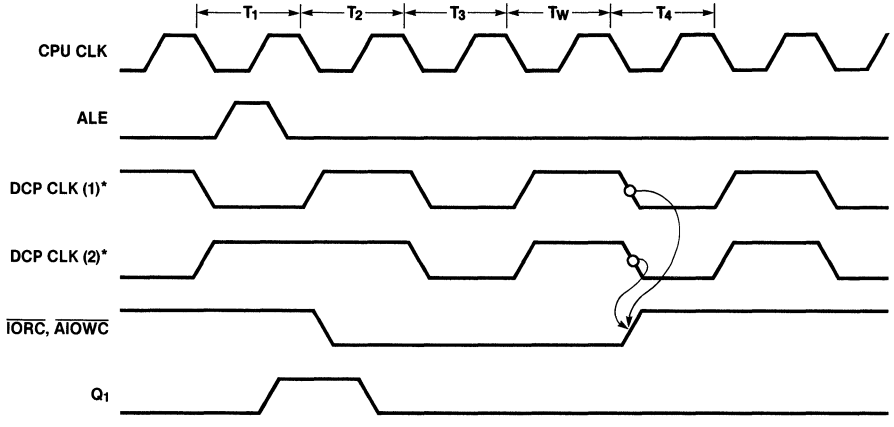
Figure 4-3.8g 8086/8088-Am9568 Interface (1 Wait State)



* DCP CLK (1) AND (2) SHOW TWO PHASES OF DCP CLK

02188A-39

Figure 4-3.8h DCP CLK Synchronization Timing (No Wait States, CLK SYNC A)



* DCP CLK (1) AND (2) SHOW TWO PHASES OF DCP CLK

02188A-40

Figure 4-3.8j DCP CLK Synchronization Timing (1 Wait State, CLK SYNC B)

DATA CIPHERING SPEED

The data ciphering speed of the DCP is limited by the byte transfer capability of the 8086 bus. A high-performance DMA like the AM9516 increases the throughput as shown in the following table:

8086 clock	DMA clock	DCP clock	N	T
8 MHz	4 MHz	4 MHz	36	0.78 MByte/s
6 MHz	6 MHz	3 MHz	18	1.05 MByte/s
8 MHz	no DMA	4 MHz	70	0.42 MByte/s

The formula for calculating the throughput is:

T = (8 * f) / (N + 5) MByte/s

T = Throughput in MByte/s

N = Number of clock cycles per 8 byte transfer

5 = Internal operation time (5 clocks per block)

f = DCP clock in MHz

8 = 8 data bytes per block

The first two cases in the table above are fast enough to encrypt and decrypt the data transferred to or from a 5 1/4-inch Winchester Disk Controller "on the fly" (5 MBit/s = 0.625 MByte/s).

TESTING

The interface of Figures 4-3.8f and 4-3.8g and both Clock Synchronizers were built and tested using the software described below.

— The DCP is reset by software writing "00_H" to the Command Register.

— The ciphering mode is selected by writing "18_H" into the Mode Register. Here the mode is: Master Port-only configuration, Electronic Code Book (ECB) and Encryption.

— The Clear Encryption key is loaded through the Master Port by issuing the command "11_H". After the command is entered, the Status Register content is read out. Only the Command Pending bit should be set (40_H). If other bits are set, the program sets the error flag "CODE" to FF_H and terminates. If the status is correct, eight bytes of key are strobed in through the Master Port in eight output instructions. The Key is "8001010101010101_H". The most significant byte is loaded first.

— The status of the DCP is checked, the Command Pending bit and the parity error bits should be reset (00_H).

— The encryption is started by entering the command "Start Encryption" (41_H).

— One block of data (8 bytes) is strobed into the Master Port. The source is the byte string "PLAIN". In this example, the plain text is: "0000000000000000_H".

— Loop 3 is executed until the Busy bit of the Status Register shows the encryption is done.

— One block of ciphered data is read out of the Master Port and transferred to the program location "CIPHER". The ciphered text should be: "$95A8D72813DAA94D_H$".

— The Status Register is checked; only the Start Entered bit should be set (80_H).

— The encryption session is stopped by issuing the command "Stop Encryption" ($E0_H$).

— After that the status should be 00_H; all flags are reset.

The program can be used to decrypt data, if two program locations are changed:

— The "Enter Key" command of location 0110_H has to be changed to 12_H ("Load Clear D-Key Through Master Port").

— The Start Command of location 0131_H has to be changed to 40_H ("Start Decryption"). After running the program, the error flag in "CODE" should be reset (00_H).

This test was performed to verify the communication between the 8086 and the DCP. By providing clear and encrypted data for the key shown, users should be able to verify operation of any variation to the design. The software was kept simple to avoid dependence on other hardwares in the system.

```
        ASM86 VER 1.0   SOURCE: APPL8068.ASM

+-------------------------------------------------
;                                          JUERGEN STELBRINK  4-12-83
;                                          ADVANCED MICRO DEVICES
;
;        8086 TO AM9518 (AMZ8068) INTERFACE TEST PROGRAM
;
;-------------------------------------------------------------------------
;
;
; ADDRESSES OF THE DCP (EVEN ADDRESSES)
;
FC00                            MPSEL   EQU     0FC00H    ; BASE ADDRESS OF MASTER PORT
FC02                            MPCOM   EQU     MPSEL+2   ; COMMAND REGISTER (WRITE ONLY)
FC02                            MPSTAT  EQU     MPSEL+2   ; STATUS REGISTER (READ ONLY)
FC06                            MPMODE  EQU     MPSEL+6   ; MODE REGISTER (READ AND WRITE)
FC00                            MPINP   EQU     MPSEL     ; INPUT REGISTER (WRITE ONLY)
FC00                            MPOUT   EQU     MPSEL     ; OUTPUT REGISTER (READ ONLY)
0018                            ECB     EQU     18H       ; ENCRYPT, MP ONLY, ECB
0080                            KEY1    EQU     80H       ; KEY: 80010101010101011H
0001                            KEY2    EQU     01H
                        ;
                                ORG     100H
                        ;
0100 BA 02 FC           BEGIN:  MOV     DX,MPCOM          ; DX: POINTER TO PORT ADDRESS
0103 B0 00                      MOV     AL,0              ; SOFTWARE RESET
0105 EE                         OUT     DX,AL
                        ;
0106 BA 06 FC                   MOV     DX,MPMODE
0109 B0 18                      MOV     AL,ECB            ; SELECT MODE
010B EE                         OUT     DX,AL
                        ;
010C BA 02 FC                   MOV     DX,MPCOM
010F B0 11                      MOV     AL,11H            ; LOAD CLEAR E-KEY THROUGH MP
0111 EE                         OUT     DX,AL
                        ;
0112 BA 02 FC                   MOV     DX,MPSTAT
0115 EC                         IN      AL,DX             ; READ STATUS
0116 3C 40                      CMP     AL,40H            ; 40= CP SET
0118 75 5F                      JNE     ERROR
                        ;
011A B0 80                      MOV     AL,KEY1           ; LOAD 1. KEY BYTE
011C BA 00 FC                   MOV     DX,MPINP
011F EE                         OUT     DX,AL             ; OUTPUT 1. KEY BYTE
0120 B9 07 00                   MOV     CX,7              ; LOAD COUNTER FOR NEXT 7 BYTES KEY
0123 B0 01                      MOV     AL,KEY2           ; FOLLOWING KEY DATA
0125 EE                 LOOP1:  OUT     DX,AL
0126 E2 FD                      LOOP    LOOP1
                        ;
0128 BA 02 FC                   MOV     DX,MPSTAT
012B EC                         IN      AL,DX             ; READ STATUS
012C 3C 00                      CMP     AL,0              ; FLAGS RESET?
012E 75 49                      JNE     ERROR
                        ;
0130 BA 02 FC                   MOV     DX,MPCOM
0133 B0 41                      MOV     AL,41H            ; START ENCRIPTION
0135 EE                         OUT     DX,AL
                        ;
0136 B9 08 00                   MOV     CX,8              ; 8 BYTES (1 BLOCK) OUTPUT
```

Figure 4-3.8k

```
ASM86 VER 1.0   SOURCE: APPL8068.ASM


  0139 BB 00 00                 MOV     BX,0                    ; INITIALIZE POINTER
  013C BA 00 FC                 MOV     DX,MPINP
  013F 2E 8A 87 81 01   LOOP2:  MOV     AL,PLAIN[BX]            ; LOAD DATA
  0144 43                       INC     BX                      ; INCREMENT POINTER
  0145 EE                       OUT     DX,AL                   ; WRITE PLAIN DATA
  0146 E2 F7                    LOOP    LOOP2
                        ;
  0148 BA 02 FC                 MOV     DX,MPSTAT
  014B EC               LOOP3:  IN      AL,DX                   ; WAIT UNTIL ENCRYPTION IS DONE
  014C 24 20                    AND     AL,20H                  ; TEST BUSY BIT
  014E 75 FB                    JNZ     LOOP3
                        ;
  0150 B9 08 00                 MOV     CX,8                    ; 8 BYTES (1 BLOCK) INPUT
  0153 BB 00 00                 MOV     BX,0                    ; INITIALIZE POINTER
  0156 BA 00 FC                 MOV     DX,MPOUT
  0159 EC               LOOP4:  IN      AL,DX                   ; READ ENCRYPTED DATA
  015A 2E 88 87 89 01           MOV     CIPHER[BX],AL           ; STORE DATA
  015F 43                       INC     BX                      ; INCREMENT POINTER
  0160 E2 F7                    LOOP    LOOP4
                        ;
  0162 BA 02 FC                 MOV     DX,MPSTAT
  0165 EC                       IN      AL,DX                   ; TEST STATUS REGISTER
  0166 3C 80                    CMP     AL,80H                  ; 80= START ENTERED
  0168 75 0F                    JNE     ERROR
                        ;
  016A BA 02 FC                 MOV     DX,MPCOM
  016D B0 E0                    MOV     AL,0E0H                 ; STOP ENCRYPTION
  016F EE                       OUT     DX,AL
                        ;
  0170 BA 02 FC                 MOV     DX,MPSTAT
  0173 EC                       IN      AL,DX                   ; TEST STATUS REGISTER
  0174 3C 00                    CMP     AL,0                    ; ALL BITS MUST BE RESET
  0176 75 01                    JNE     ERROR
  0178 C3                       RET
                        ;
  0179 B0 FF            ERROR:  MOV     AL,0FFH                 ; LOAD ERROR CODE
  017B 2E A2 80 01              MOV     CODE,AL
  017F C3                       RET
                        ;
  0180 00               CODE    DB      00H                                     ; ERROR CODE
  0181 00 00 00 00 00 00 PLAIN  DB      00H,00H,00H,00H,00H,00H,00H,00H         ; PLAIN TEXT
       00 00
  0189 12 23 34 45 56 67 CIPHER DB      12H,23H,34H,45H,56H,67H,78H,89H         ; CIPHER TEXT
       78 89
                        ;
                                END

END OF ASSEMBLY. NUMBER OF ERRORS:   0
```

Figure 4-3-8k (continued)

4.3.9 THE 8086 AND Am9519A INTERRUPT CONTROLLER INTERFACE

This interface illustrates the use of the Am9519A Interrupt Controller with the 8086 (Figure 4-3.9a). The Am9519A has many advantages over the 8259A. However, some external logic is required. The purpose of this logic is to convert the two interrupt acknowledge cycles from the 8086 into one. Those familiar with the Am9519A may question this since the Am9519A can be programmed to accept two interrupt acknowledge pulses. The reason for choosing this approach is faster response time. The Am9519A requires the first interrupt acknowledge pulse to be 900 nsec in order to resolve priority, therefore, Wait States would need to be inserted. However, the simple logic shown eliminates the need for Wait States and meets the Am9519A timing requirements.

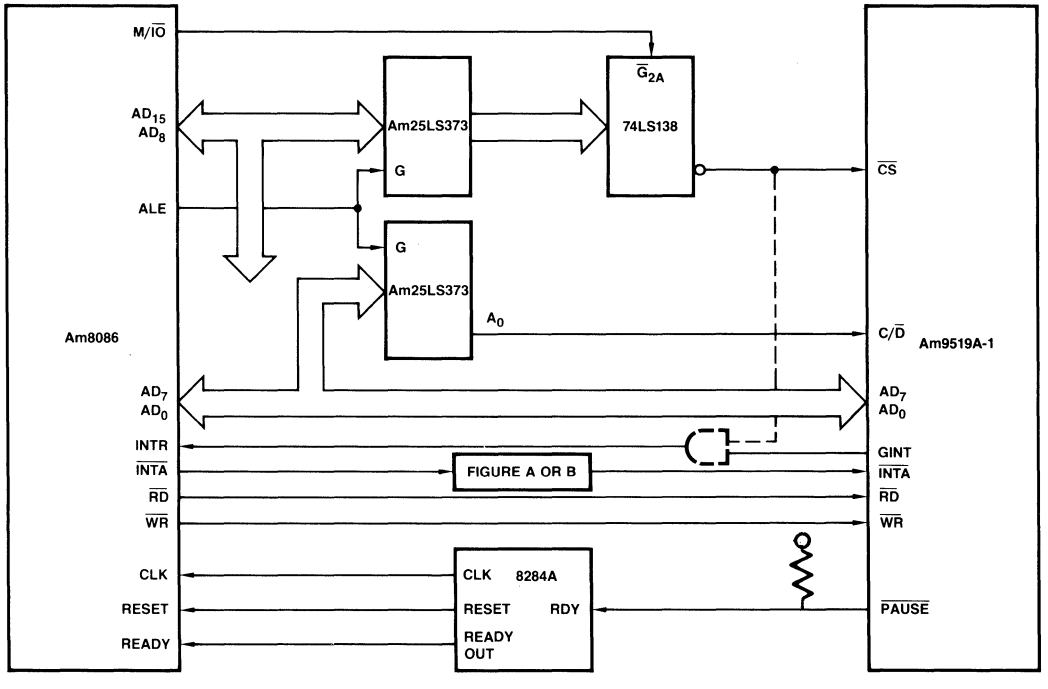
The gate shown by dotted lines is optional. Its purpose is to prevent interrupts when the CPU does a write to the Mask Register. This can be done in software by disabling interrupts whenever a write to the Interrupt Mask Register (IMR) is done.

The discussion so far has been centered on the Interrupt Acknowledge response. When programming the Am9519A, a careful timing analysis shows that, for a 5 MHz CPU, the Am9519A-1 should be used or a Wait State inserted. For the 8 MHz CPU, a Wait State will be required. This is due to the uncertainty of where the /RD pulse occurs with respect to the clock. This will result in the data-to-clock setup time not being met in some cases. The best resolution is to insert a Wait State for programmed I/O.

(option)

The Am9519A interface illustrates two options to this logic. The first is an AND gate inserted into the interrupt request line. This allows users to dynamically mask channels in the Am9519A without hanging the system up. Without this gate, it is necessary to disable CPU interrupts, change the mask register, and re-enable CPU interrupts. This hang up occurs if the CPU masks off a channel which is causing the GINT to be active. The CPU recognizes the interrupt before the mask removes the request. When the interrupt acknowledge cycle starts, and since the channel was masked, the Am9519A will assert pause indefinitely.

The second option is an alternate way to convert the processor's two acknowledge pulses into one acknowledge pulse. The timing diagram illustrates the difference. User should choose the one that eliminates Wait States.



02188A-41

Figure 4-3.9a

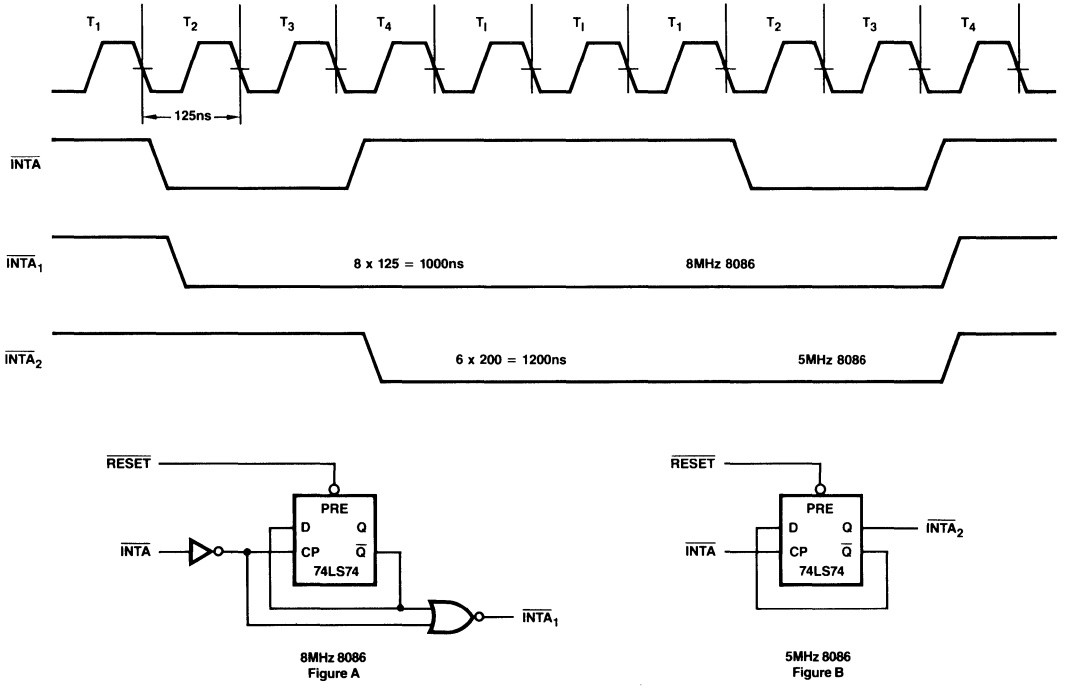


Figure 4-3.9b

4.3.10 iAPX286 TO Am9568 DATA CIPHERING PROCESSOR INTERFACE

This interface is designed for an 8-MHz CPU where the DCP is synchronously operating at the maximum clock rate of 4 MHz. Block diagram for the interface is shown in Figure 4-3.10a. The Am9568 requires a narrower width of address strobe than the Am9518. This works comfortably with the 60 ns address strobe width of an 8-MHz CPU.

The Multibus* Mode Select input of the Bus Controller 82288 is tied Low to optimize the command and control signals for short bus cycles. The Command Delay (CMDLY) becomes active High for one 16-MHz clock cycle whenever the DCP is selected to delay the Read and Write strobes by 125 ns. This satisfies the timing requirement of the minimum delay between ALE inactive and Read or Write strobe active of the DCP. An open collector gate must be added to allow other peripherals to drive this input.

The ALE, $\overline{IORC}$ and $\overline{IOWC}$ outputs of the 82288 are wired directly to the DCP. ALE strobes a D-Flip-Flop to store the state of Chip Select for the entire cycle.

$\overline{Q}_3$ and the latched Chip Select CSL are ANDed externally to generate the Synchronous Ready for the 82284. The 82284 samples the line at the falling edge of the clock. The registered output $\overline{Q}_3$ is clocked with the rising edge of the same clock, thus satisfying the setup and hold time requirements of the 82284. Two Wait States are inserted.

Half of the PAL device operates as a bidirectional Address/Data Multiplexer. During the Address Latch Enable active phase, the state of A_1 and A_2 is transferred to the AD_1 and AD_2 pin of the PAL device. The DCP latches this two bit-address with the falling edge of ALE.

When $\overline{IORC}$ and CSL are active, the states of AD_1 and AD_2 are passed to D_1 and D_2 respectively. The DCP Register can be read. If $\overline{IOWC}$ and CSL are active, the data path is turned around: D_1 and D_2 are inputs, AD_1 and AD_2 are outputs.

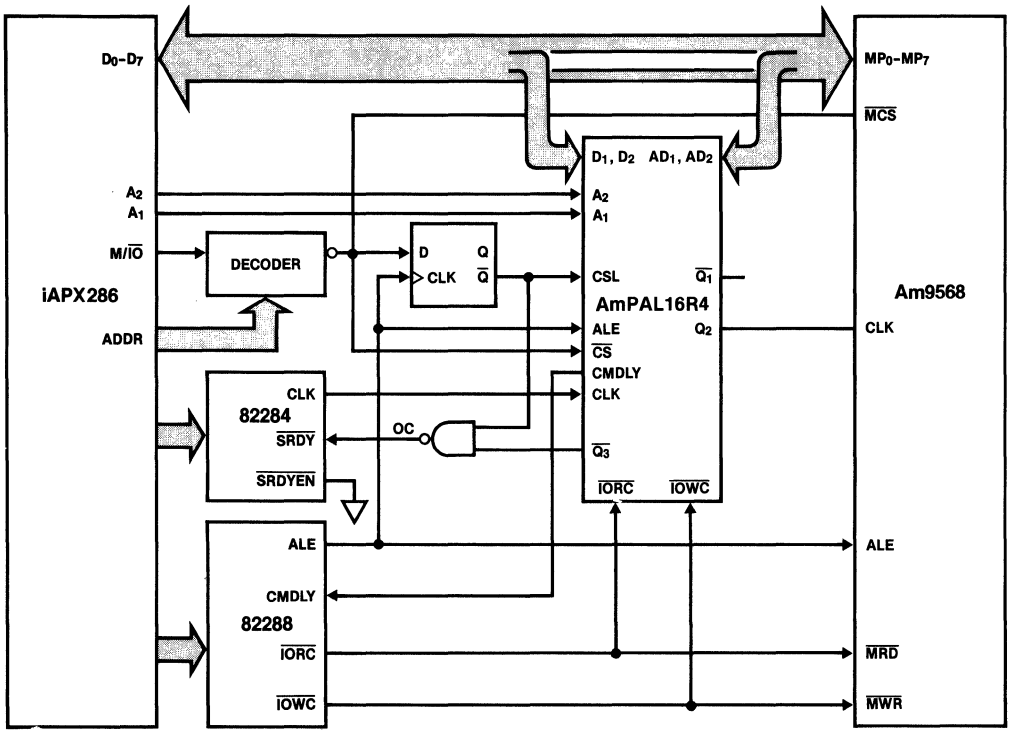
The Address Hold Time of the PAL device is sufficient because the address information is passed to AD_1 and AD_2 whenever IORC*CSL or IOWC*CSL are not true, i.e. whenever data is not transferred between the CPU and the DCP.

The Read Data Hold Time requirement of 5 ns of the Am9568 is satisfied by the propagation delay of the PAL device.

The Read Data Hold Time requirement of 5 ns of the iAPX286 is also satisfied by the PAL device.

The Master Port Chip Select ($\overline{MCS}$) input of the DCP is connected to the unlatched address decoder output.

*MULTIBUS is a registered trademark of Intel Corporation.



02188A-43

Figure 4-3.10a iAPX286-Am9568 Interface

THE DCP CLOCK

The PAL device synchronizes the DCP clock to the Data Strobes $\overline{IORC}$ and $\overline{IOWC}$ (Figure 4-3.10b). It also divides the 16-MHz system clock (8-MHz CPU clock) down to the maximum DCP clock rate of 4 MHz. At this clock rate, the Data Strobe Delay to the DCP clock must be 0-30 ns. The Bus Controller is specified to generate a Data Strobe timing of 3-15 ns to the falling edge of CLK (16 MHz). Because of the higher propagation delay of a standard PAL device, the registered outputs are toggled at the rising edge of CLK before the Data Strobes become inactive. This gives additional 32.5 ns for the DCP clock signal path.

Q_1 to Q_3 are three outputs of the PAL device state machine. The registered output are clocked with the rising edge of the 16-MHz 82284 clock. Whenever ALE and $\overline{CS}$ are active, Q_1 to Q_3 are set to the initial state. Q_1 to Q_3 are outputs of a 3-bit down counter, with Q_3 as the most significant bit.

Q_3 is used to generate the $\overline{SRDY}$ signal for the 82284 as mentioned above.

Q_2 is the DCP clock. This design must guarantee that the minimum DCP clock High or Low time is at least 115 ns or two 16-MHz clock cycles. This is done by toggling Q_2 only during phase 2 cycles of the CPU. The CPU design guarantees that there is always a phase 1 cycle between two phase 2 cycles.

Assuming a typical PAL device propagation delay of 25 ns, timing parameter TCDS (Time Clock Data Strobe) is 10.5 to 22.5 ns (3 + 32.5 - 25 ns to 15 + 32.5 - 25 ns). This satisfies the 0 to 30 ns requirement.

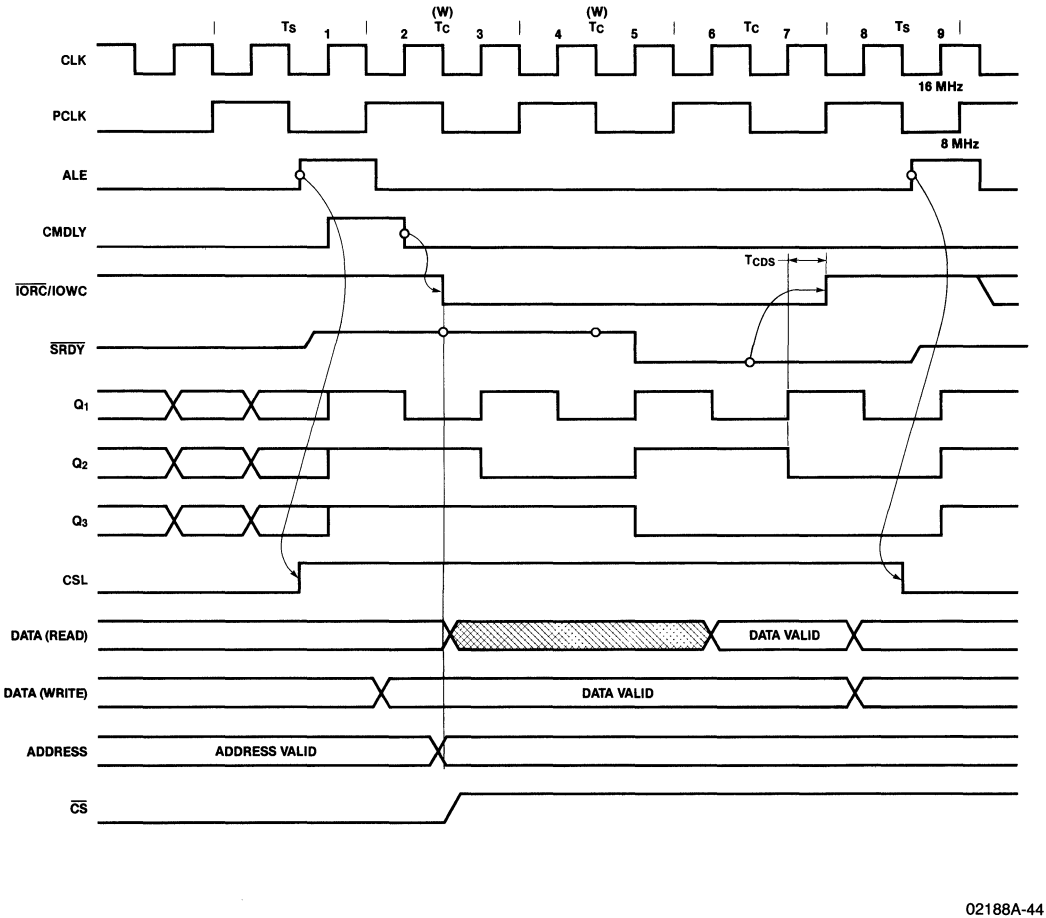
The AmPAL16R4 has active Low outputs. But one output, Q_2, should be active High. The equation for Q_2 was derived to be

$$Q_2 = ALE * CS\ Q_1 * Q_2\ \overline{Q}_1 * \overline{Q}_2$$

To compensate for the inversion in the PAL device, either de Morgan Theorem or Karnaugh-Veitch diagrams can be used to convert it to the form shown in PAL device Design Specification.

IMPROVEMENTS

The DCP needs two Wait States only when the Control Registers are read. Data Register read or writes and Control Register reads can be executed with only one Wait State, which improves the Data Ciphering speed of this interface. The more sophisticated Wait control logic and the two external TTL gates can be integrated into one AmPAL22V10 device.



02188A-44

Figure 4-3.10b Timing Diagram

```
PAL16R4                                          PAL DESIGN SPECIFICATION
DCP043                                           JUERGEN STELBRINK  8-23-83
iAPX286 - Am9568 (DCP) INTERFACE DEVICE
ADVANCED MICRO DEVICES

CLK  /CS  CSL  ALE  /IORC  /IOWC  A1     A2   NC   GND
/OE  D1   D2   /Q1  Q2     /Q3    CMDLY  AD1  AD2  VCC

Q1      :=  ALE*CS + /Q1

/Q2     :=  Q1*/Q2*/ALE + Q1*/Q2*/CS + /Q1*Q2*/ALE + /Q1*Q2*/CS

Q3      :=  ALE*CS + Q1*Q2*Q3 + /Q1*Q2*Q3 + Q1*/Q2*Q3 + /Q1*/Q2*/Q3

/CMDLY  :=  /ALE+/CS

IF(CSL*IORC) /D1  =  /AD1

IF(CSL*IORC) /D2  =  /AD2

IF(CSL*/IORC) /AD1 =  /A1*ALE + /D1*/ALE

IF(CSL*/IORC) /AD2 =  /A2*ALE + /D2*/ALE


FUNCTION TABLE

CLK /CS CSL ALE /IORC A1 A2 D1 D2 AD1 AD2 /Q1 /Q2 /Q3 CMDLY

;              /                                    C
;              I                                    M
; C  /  C  A   O                  A  A    /  /  /   D
; L  C  S  L   R    A  A  D  D    D  D    Q  Q  Q   L
; K  S  L  E   C    1  2  1  2    1  2    1  2  3   Y     COMMENT
-------------------------------------------------------------------
;
  C  L  H  H   H    L  L  Z  Z    L  L    L  L  L   H ; 1 (/CS ACTIVE)
  X  L  H  H   H    L  H  Z  Z    L  H    L  L  L   H
  X  L  H  H   H    H  H  Z  Z    H  H    L  L  L   H
  C  L  H  L   H    H  L  L  ↯    L  H    H  L  L   L ; 2 (WRITE CYCLE)
  X  H  H  L   L    L  L  L  H    L  H    H  L  L   L ;   (READ CYCLE)
  C  H  H  L   L    H  L  L  H    L  H    L  H  L   L ; 3
  C  H  H  L   L    H  L  L  L    L  L    H  H  L   L ; 4
  C  H  H  L   L    H  L  H  H    H  H    L  L  H   L ; 5
  C  H  H  L   H    H  L  H  H    H  H    H  L  H   L ; 6
  C  H  H  L   H    H  L  L  L    L  L    L  H  H   L ; 7
  C  H  H  L   H    H  L  H  L    H  L    H  H  H   L ; 8
;
  C  H  L  H   H    X  X  Z  Z    Z  Z    L  L  L   L ; 1 (NO /CS)
;
-------------------------------------------------------------------
```

DESCRIPTION:

INPUT SIGNALS:

CLK 16 MHZ SYSTEM CLOCK OF THE 82284 SYSTEM TIMING CONTROLLER. THIS CLOCKS TRIGGERS THE D-FLIP-FLOPS OF FOUR PAL OUTPUTS

/CS ACTIVE LOW UNLATCHED CHIP SELECT OF THE ADDRESS DECODER

CSL ACTIVE HIGH LATCHED CHIP SELECT. IT HAS TO BE ACTIVE TO THE RISING EDGE OF ALE OF THE NEXT CYCLE

ALE ADDRESS LATCH ENABLE OF THE 82288 BUS CONTROLLER

A1,A2 DEMULTIPLEXED ADDRESS INPUTS. THEY CARRY THE 2-BIT REGISTER ADDRESS FOR THE DCP

/IORC INPUT/OUTPUT READ CONTROL OF THE 82288

/IOWC INPUT/OUTPUT WRITE CONTROL OF THE 82288

OUTPUT SIGNALS:

/Q1 INTERNAL STATE SIGNAL. IT IS DIVIDED BY TWO FROM CLK AND SYNCHRONIZED TO ALE

/Q2 INTERNAL STATE SIGNAL. IT IS DIVIDED BY TWO FROM /Q1 AND SYNCHRONIZED TO ALE. IT IS THE INVERTED DCP CLOCK (4MHZ). THE RIGHT EDGE OF Q2 IS SYNCHRONOUS TO THE DATA STROBES /IORC AND /IOWC, IF TWO WAIT STATES ARE INSERTED.

/Q3 INTERNAL STATE SIGNAL. IT IS DIVIDED BY TWO FROM /Q2 AND SYNCHRONIZED TO ALE. IT IS USED TO GENERATE THE SYNCHRONOUS READY (/SRDY) FOR THE 82284. EXTERNALLY IT HAS TO BE LOGICALLY AND'ED WITH THE THE LATCHED CHIP SELECT (CSL).

CMDLY COMMAND DELAY GOES ACTIVE FOR ONE CLOCK WIDTH TO DELAY THE DATA STROBES. THE AM9568 REQUIRES A DELAY BETWEEN ALE INACTIVE AND DATA STROBE ACTIVE.

BIDIRECTIONAL SIGNALS:

D1,D2 DEMULTIPLEXED DATA BUS LINES TO 8086 CPU

AD1,AD2 MULTIPLEXED ADDRESS/DATA BUS LINES FOR THE DCP

5.0 INTERFACING TO THE 8088

5.1.0 8088 OVERVIEW

The 8088 CPU is an 8-bit processor designed around the 8086 internal structure. Most functions of the 8088 are identical to the equivalent 8086. The 8088 fetches and writes 16-bit words in two consecutive bus cycles. Both the 8086 and the 8088 handle the external bus the same way but the 8088 handles only 8-bits at a time; and both appear identical to the software engineer, with the exception of execution time.

The hardware interface of the 8088 contains the major differences between the two CPUs. The pin assignments are nearly identical, however, with the following functional changes:

A_8-A_{15} — These pins are only address outputs on the 8088. They are latched internally and remain valid throughout a bus cycle in a manner similar to the 8085 upper address lines.

$\overline{SS0}$ — Provides the $\overline{SO}$ status information in the minimum mode. This output occurs on pin 34 in minimum mode only.

DT/$\overline{R}$, IO/$\overline{M}$, and $\overline{SS0}$ provide the complete bus status in minimum mode.

IO/$\overline{M}$ has been inverted to be compatible with the MCS-85 bus structure.

ALE is delayed by one clock cycle in the minimum mode when entering HALT, to allow the status to be latched with ALE.

5.2.0 8088 AND Z8000 PERIPHERALS

Most of the interface design between the 8088 and Z8000 peripherals are similar to that of the previous chapter. The user is referred back to Chapter 4 of this manual when the interface is similar.

5.3.0 THE 8088 AND AMD PROPRIETARY PERIPHERALS

The evolution of chip design has taken the 8-bit environment into the 16-bit environment. While the new generation of peripheral devices are often 16 bits wide, the older, established 8-bit orientation of CPUs and peripherals are still significant. Interfacing a 16-bit peripheral with an 8-bit CPU often encounters data path incompatibility and involves bus control manipulation. This type of integration mainly involves separating the control and data paths from the new peripheral and the system.

The ability to mix different data path widths can improve system functionality, performance, and cost. It is less expensive to use an 8-bit bus in a new design because the memory requirements are generally cheaper. A designer can use this data path mixing to upgrade the existing system until a new system design is warranted, or the designer can simply improve on the existing design as new peripherals become available. AMD makes a number of proprietary peripherals and the following sections show users with 8-bit systems how to incorporate those AMD products into their designs.

5.3.1 8088 AND Am8052 CRT CONTROLLER INTERFACE

The interface technique between the Am8052 CRT Controller and an 8-bit microprocessor also applies to the 8088 and Am8052 interface.

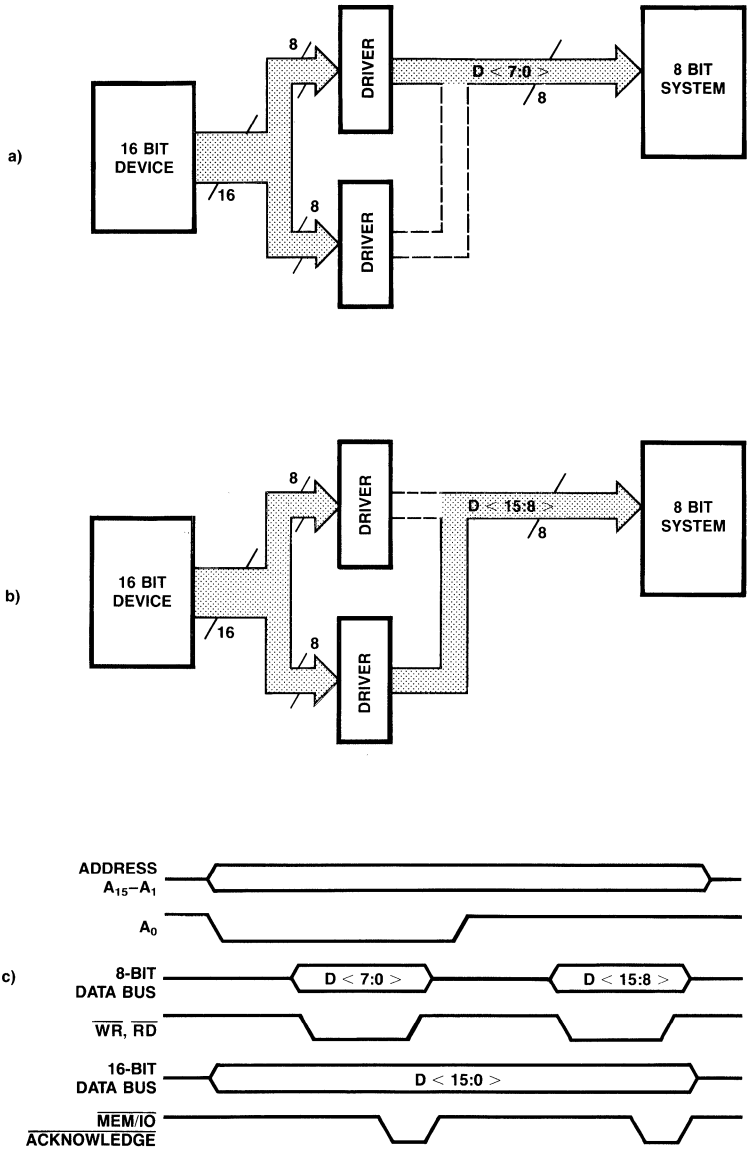
There are two fundamental issues associated with mixing devices that communicate over different-sized buses. The first problem is allowing the two devices to communicate on a "common" data bus. Consider, for example, a 16-bit system utilizing 8- and 16-bit peripherals. Overcoming the mismatched data paths requires some form of controlled multiplexing/demultiplexing of the different data paths. In addition, extra control signals for partitioning the 16-bit word into 8-, and 16-bit units may be required. Today, most of the 16-bit CPU based systems that use 8-bit peripherals usually use just the lower half of the data bus to transfer data to and from the peripheral. However, this scheme does not work when interfacing 16-bit peripherals to 8-bit CPUs, especially when these peripherals have bus master capability.

DATA FUNNELLING

When a 16-bit peripheral attempts to transfer data over an 8-bit bus (memory write cycle or slave read cycle), the 16-bit data bus has to be broken down into two bytes and transferred sequentially. First, the lower 8-bits are transferred out on the bus (Figure 5-3.1.1a), and then in the next transfer cycle the upper 8-bits of the 16-bit word are sent out (Figure 5-3.1.1b). The generalized bus timing for such an operation is shown in Figure 5-3.1.1c. Figures 5-3.1.2a, 5-3.1.2b, and 5-3.1.2c show the opposite case; a bus read operation from an 8-bit bus to a 16-bit peripheral. Here, the first byte read from the system must be latched. Once the second byte has been fetched, the 16-bit peripheral reads in the assembled 16-bit (2-byte) word. Additionally, provisions may need to be made for the case when the 16-bit peripheral accesses single bytes.

Interruptions of the two cycle transfer must be analyzed very carefully. Master transfers may not be interrupted by slave accesses while being in the middle of a two-cycle transaction. Similar, slave accesses may not be interrupted by master transfers. While the interface funnels the data, the current bus cycle needs to be stretched. When the peripheral is bus master, as shown in Figures 5-3.1.1a, 5-3.1.1b, and 5-3.1.1c, the 16-bit peripheral is holding its data available for what would normally be two complete bus transfer cycles. This stretch can be achieved by delaying the transfer acknowledge signal to the peripheral, causing it to wait ($\overline{WAIT}$ asserted).

In slave mode, the 8-bit CPU would have to make two consecutive read operations to examine a 16-bit peripheral status register. The peripheral must not become bus master in-between the first and second read operations since this invalidates the results of the first read operation. This function can be handled in two different ways: if the CPU has a bus lock instruction (for example, like the iAPX family of CPUs), then the programmer uses one of these before the CPU accesses the peripheral. Alternately, the CPU can disable the arbitration logic while it is performing the critical uninterruptible slave transfer.



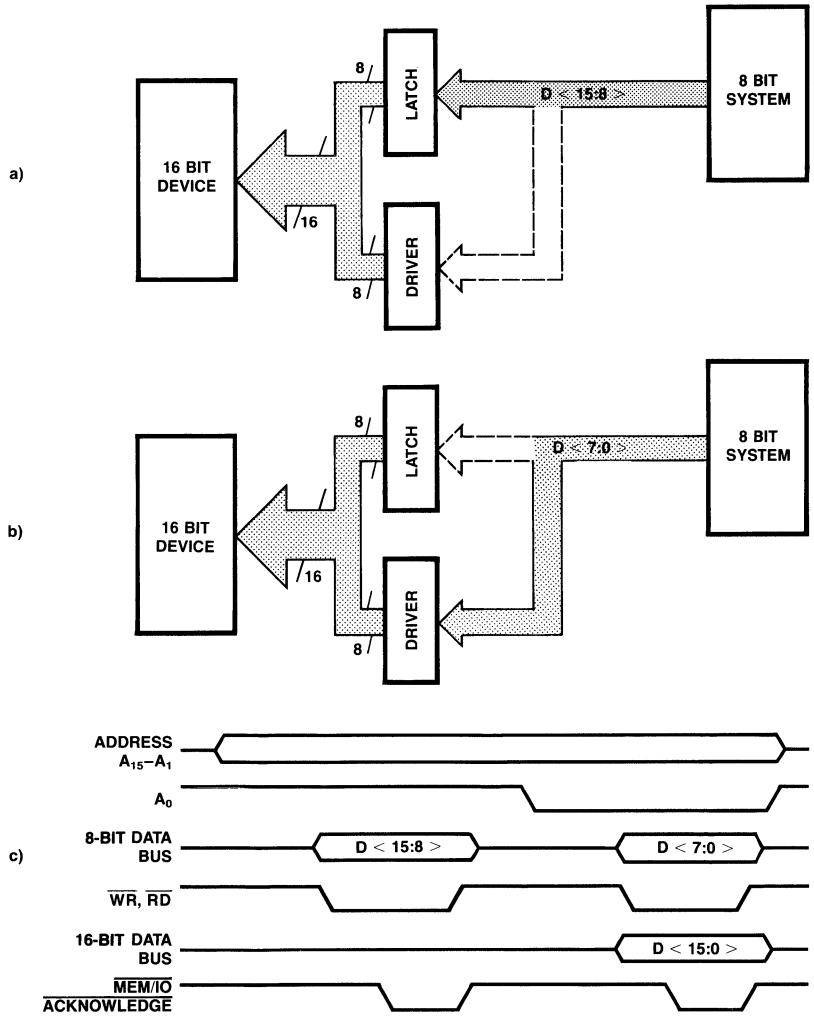
02188A-45

Figure 5-3.1.1 Bus Master Write or Slave Read Operation

DEVELOPING THE CONTROL AND DATA TRANSFER INTERFACE

Designing the control interface to allow mixing 8- and 16-bit peripherals requires an analysis of the data and control flow. The data flow automatically defines the data path design (see Figures 5-3.1.1 & 5-3.1.2). The bus master operation by the peripheral is relatively straightforward. During a write operation, the data is written out sequentially: the lower byte first and then the upper byte (or vice-versa). During a read operation, the data is fetched sequentially. The byte fetched first is latched, to hold the data until the peripheral can read it. In the second byte read cycle, the remaining byte is fetched, the 16-bit word is assembled from the two bytes, and the 16-bit word is loaded into the peripheral. Similarly, $\overline{WAIT}$ is asserted until the second byte read cycle can be terminated.

The slave mode of operation works almost identically to the peripheral bus master mode. The master read cycle is similar to the slave write cycle, and the master write cycle is similar to the slave read cycle. In general, if the peripheral puts data on the narrower system bus, the peripheral can keep the data active in both sequential system bus cycles. On the other hand, if data is loaded into the peripheral, the interface logic has to latch the data of the first fetch cycle, whereas the data of the second cycle can be loaded directly into the peripheral (no latching required).



02188A-46

Figure 5-3.1.2 Bus Master Read or Slave Write Operation

When defining the interface, the designer must make a conscious choice of which byte (upper or lower) to latch during peripheral read operations (or conversely, slave peripheral write operations). Once this decision has been made, the CPU must always access the latched data byte first (during a slave write) and then access the non-latched byte to complete the transfer. This restriction is a minor one with no extra software overhead; yet it could affect the ease of the programmer's coding if not handled properly. For example, if the programmer uses a compiler to generate the software for the system, extra care may be necessary to ensure that the compiler generates the correct addressing sequence. An alternative to this solution would be to latch both the upper and lower data bytes. In that case, the cost of the interface would be increased, as would the complexity, with no gain in performance.

The state diagram (Figure 5-3.1.3) illustrates the control sequence implemented in the 8/16-bit bus control logic. It also depicts how uninterrupted word transfers will occur and how the addresses for upper and lower bytes are generated. In addition, the specific bus timing of the peripheral and the data bus must be examined to quantify the state control flow and provide information on data latching, read/write control strobes, and addressing to and from the peripheral. The state control flow is broken down into three parts bus master read, slave read, and slave write operations.

The three control signals that must be generated by the 8/16-bit control unit are: Address bit 0 (A_0), peripheral hold ($\overline{WAIT}$), and bus read ($\overline{RD}$). The A_0 line is generated by the control logic to indicate which byte is to be transferred in bus master modes only. Otherwise, the A_0 generated by the system is used to indicate which byte is being accessed. The $\overline{WAIT}$ line holds up the peripheral during transfers. The $\overline{RD}$ line is required to indicate successive transfer cycles on the bus. The peripheral's control signals strobe active only once, because the two-cycle transfer must be kept hidden from the peripheral.

The slave transfer flow is almost identical, except that the CPU is generating the bus signals and the transfer directions are reversed, that is, a bus write goes into the peripheral.

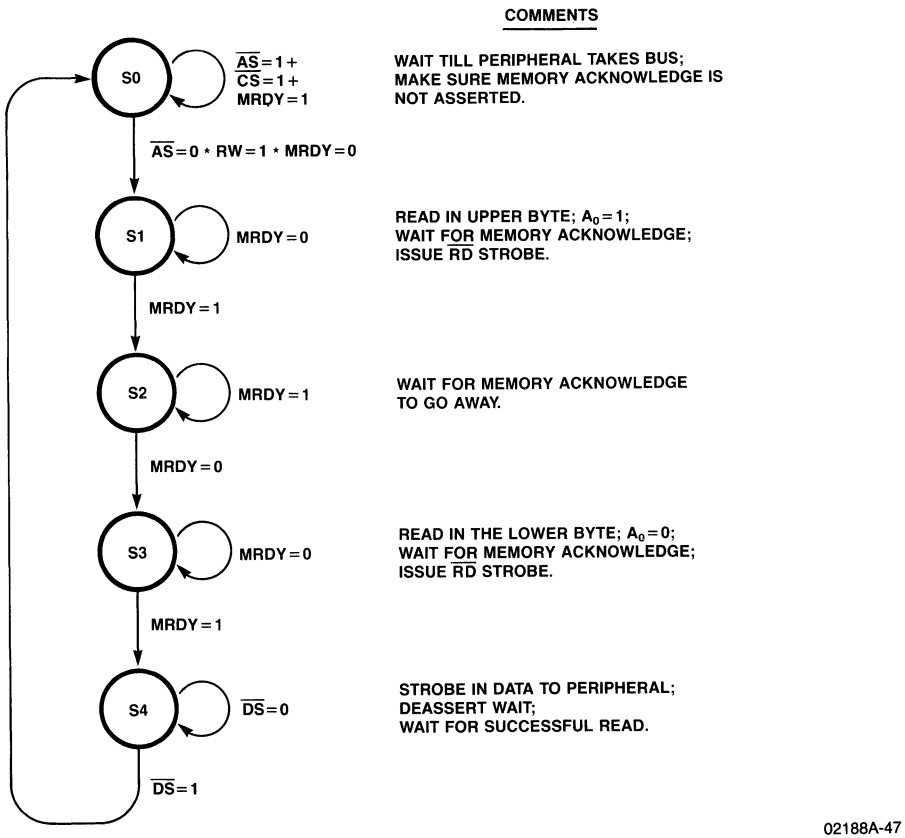
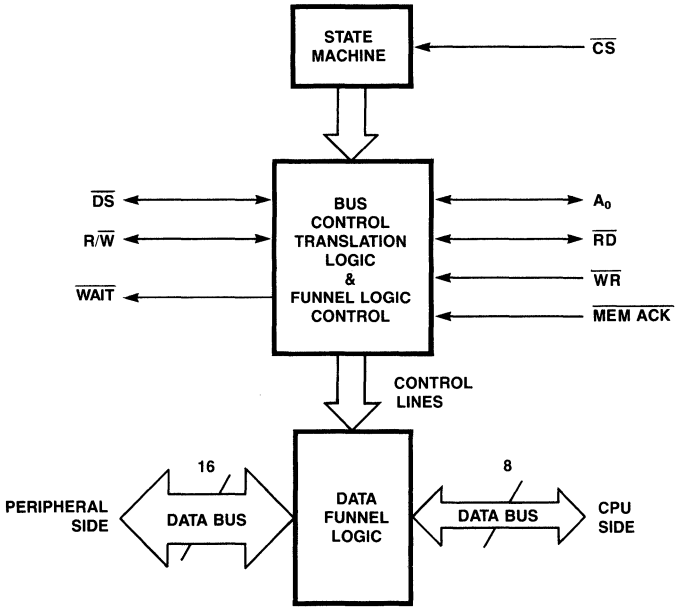


Figure 5-3.1.3 Bus Master Read State Flow-Control

The conceptual logic for the 16- to 8-bit data flow example is shown in Figure 5-3.1.4. The data on the upper byte is latched when data is being read (as a bus master) and read or written (as a bus slave). Although this interface must latch data coming from the 8-bit data bus into the peripheral, it also needs to act as transceiver when the peripheral is sending data out to the system. The ideal part to accomplish such an interface would be one that has a three-stated output, with an 8-bit wide latch, in one direction and a three-stated driver in the other direction. The Am2952 8-bit bidirectional I/O port combines the upper data bus latch and upper data driver chips into one IC. It provides two 8-bit clocked I/O ports, each with three-state output controls and individual clocks and clock enables. An Am2949 bidirectional bus transceiver completes the logic required to buffer the data path.

The state flow control requires logic capable of sequentially moving from state to state, holding in a particular state, and being reset or initialized back to a predefined state. This design integrates the state machine generator and the control signal logic into the same Programmable Array Logic (PAL) device.

A considerable amount of logic is required to generate the data-path flow logic and the bus control signals. This is especially true if the peripherals and CPUs have different signal conventions (for example, $\overline{AS}$, $\overline{DS}$, and R/$\overline{W}$ versus ALE, $\overline{RD}$, and $\overline{WR}$). Conversion between different signal conventions, signal polarity changes, and extra functions (such as generating A_0) requires quite a bit of logic and design effort. If the peripheral has bus master capability, additional information, such as bus arbitration controls, must be fed into the next state determination logic to decide what control sequence to follow.



02188A-48

Figure 5-3.1.4 Conceptual 16/8-Bit Conversion Logic

Figure 5-3.1.5 shows a typical 8/16-bit control interface which combines all the individual components discussed above. The state machine and the bus and latch controls have to be tightly coupled in order to transfer data between the 8-bit and 16-bit buses. The generalized machine is designed under the assumption that the peripheral has bus master capability. If this is not the case, the design can be greatly simplified.

Since the CRTC does not modify system memory, no provision for a bus master write operation is required. This is important because it eliminates the need to generate a system write control signal ($\overline{WR}$). In addition, the control and display information will always be aligned on word boundaries. This relieves the 8/16-bit control logic from worrying about funneling the bytes and performing odd/even byte transfers. It also saves control inputs from the Am8052 because all transfers are words; there is no need for upper and lower data strobes or byte high enable inputs/outputs.

The slave accesses by the CPU are either pointer writes (to select the desired control/status register) or 16-bit data read/write operations. The pointer write operation is really an 8-bit operation because only the lower 8 bits of the data form the register address. The three different transfer timings are shown in Figures 5-3.1.6, 5-3.1.7, and 5-3.1.8.

Two special conditions have been incorporated into the state flow diagrams whenever a transfer is first initiated. Before a new transfer cycle is attempted (that is, while the state machine is waiting in S0), the memory acknowledge must be inactive. This prevents any interference from the last transfer. The second special condition occurs when the Am8052 asserts the R/$\overline{W}$ line to indicate a write operation. Whenever the Am8052 updates the upper 8 bits of the 24-bit address latch, the R/$\overline{W}$ line indicates a write operation (in conjunction with $\overline{AS}$). The Am8052 is not actually performing a system data write, only an address latch update. Hence, the state flow reflects this fact by not starting a sequence if the R/$\overline{W}$ line is active Low from the Am8052.

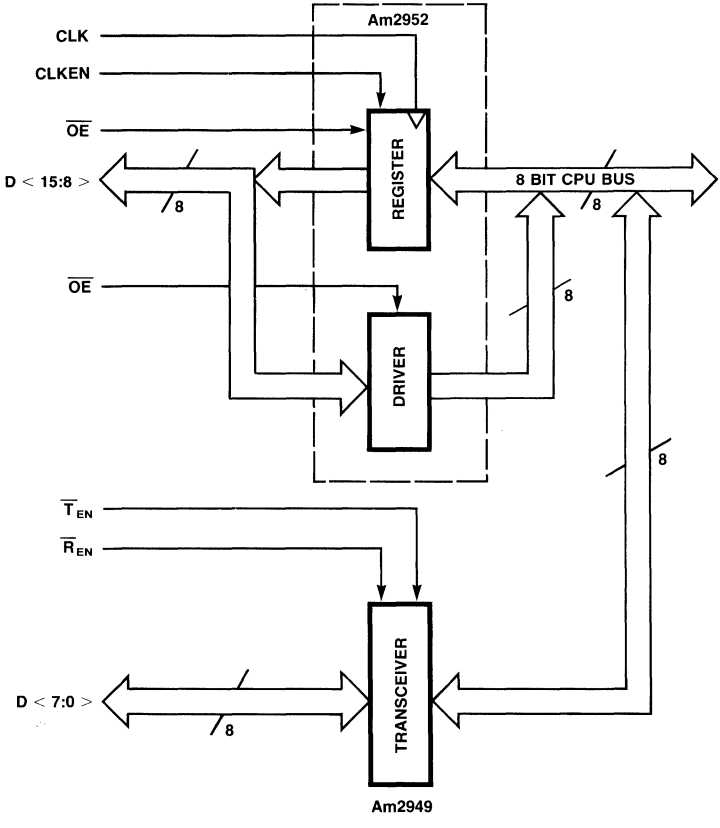


Figure 5-3.1.5 Data Funnel Logic

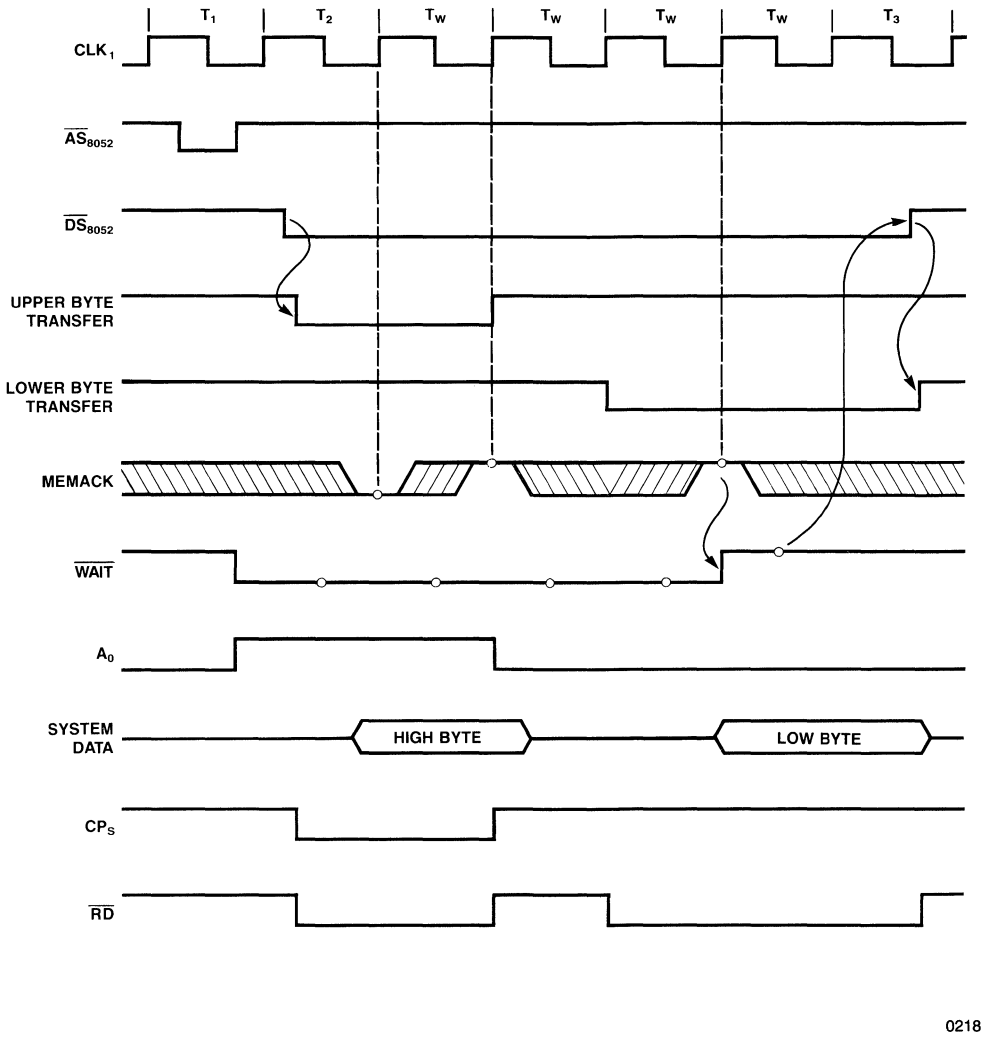
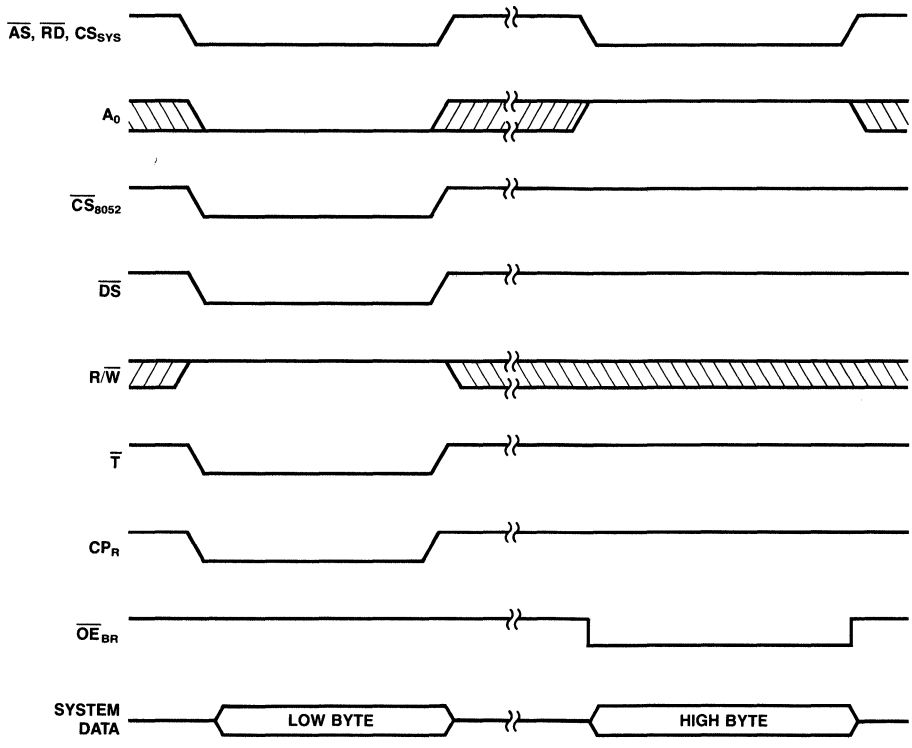


Figure 5-3.1.6 Bus Master Read Timing Diagram



02188A-51

Figure 5-3.1.7 Slave Read Timing Diagram

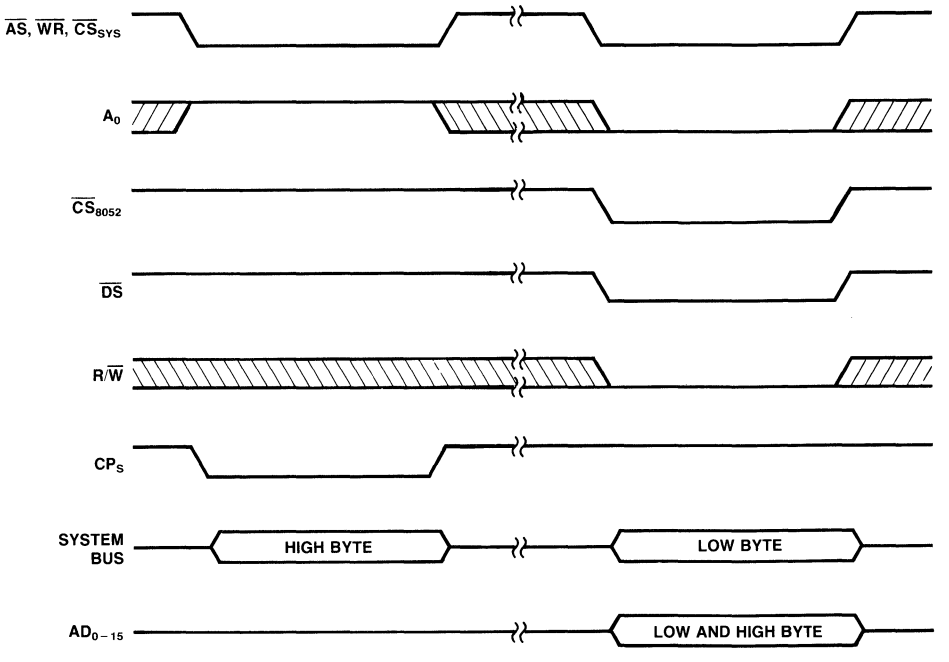


Figure 5-3.1.8 Slave Write Timing Diagram

These simplifications make it possible to combine the Am8052 to an 8-bit CPU control interface in a single AmPAL22V10 device (Figure 5-3.1.9) which also converts the bus control signals from $\overline{AS}$, $\overline{DS}$, and R/$\overline{W}$ to $\overline{RD}$ and $\overline{WR}$. Figure 5-3.1.9 shows the assembled control and data transfer logic for this interface. The minimum Am8052 and bus control signals that have to be generated are $\overline{RD}$, A_0, $\overline{DS}$, R/$\overline{W}$. Although $\overline{DS}$ and R/$\overline{W}$ are used as inputs during a bus master operation by the Am8052, the AmPAL22V10 must convert the CPU $\overline{RD}$ and $\overline{WR}$ signals to $\overline{DS}$ and R/$\overline{W}$ for slave I/O operations. The signals A_0 and $\overline{RD}$ are generated by the control logic when the Am8052 is performing a read access to the system. The $\overline{WAIT}$ (or not READY) signal to the Am8052 also needs to be generated by the control logic. Additionally, the four control signals of the bidirectional port and transceiver are generated.

TRADE-OFFS AND LIMITATIONS

In a design dramatically affecting the I/O of the system, a number of trade-offs and limitations should be noted. The most obvious limitation in using 16-bit peripherals on an 8-bit bus is that the 16-bit peripheral will be under-utilized. The speed of all I/O operations will be cut by 50%. Consequently, the bus utilization percentage will go up if the 16-bit peripheral represents a significant factor of the bus usage. A CRT controller like the Am8052 might use 5% to 10% of the bus bandwidth for display information when using 16-bit I/O. Converting to 8-bit I/O would double bus usage to 10% or 20%.

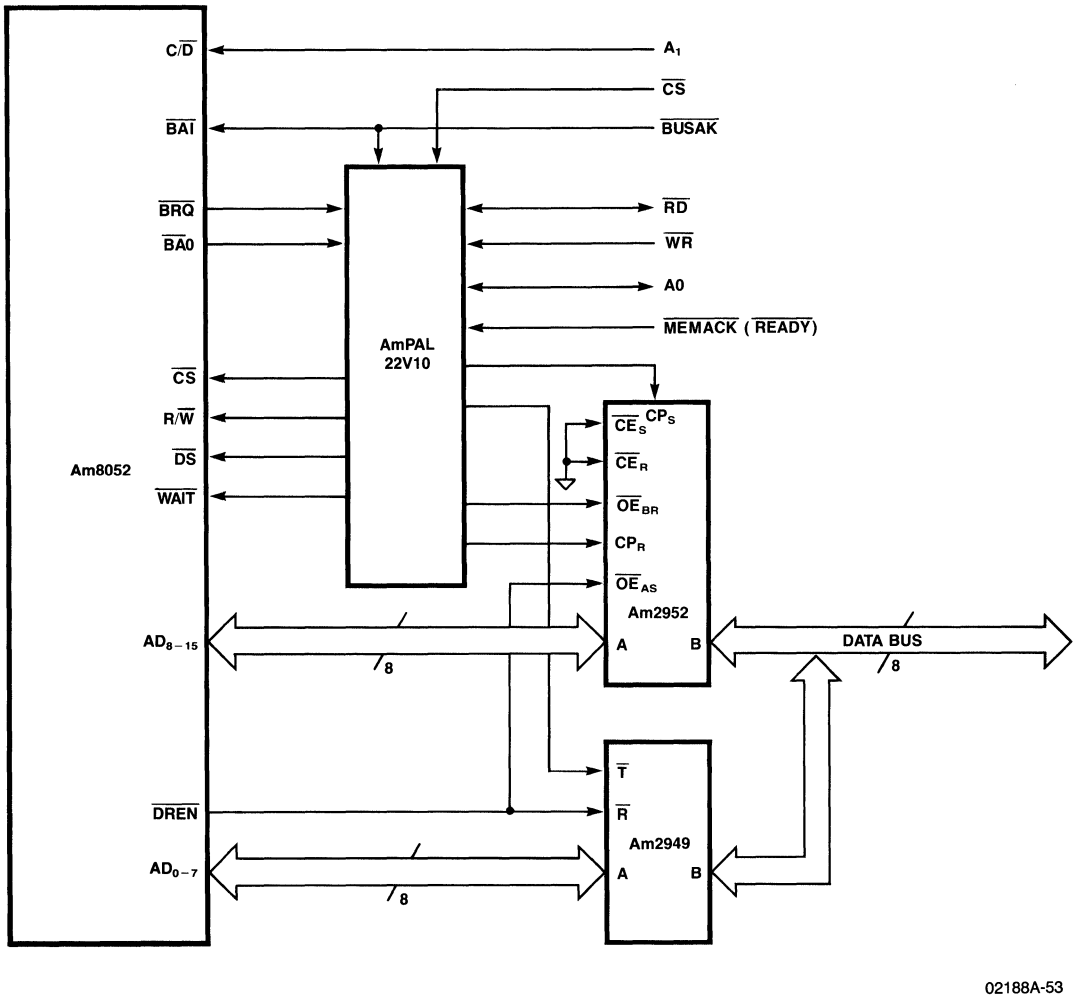


Figure 5-3.1.9 Am8052 8-Bit Interface

Another factor that might affect the bus usage is the efficiency of the 8- to 16-bit conversion control logic. If the state machine designed to perform the 8/16-bit conversion (or 16/32-bit) is improperly designed, then extra transfer overhead may be introduced. This could mean that a sequential transfer of two 8-bit values takes longer than two single 16-bit transfers. The system designer must weight the cost of the extra overhead on a case-by-case basis. Most interfaces outside a system's immediate family require some kind of extra interface logic anyway. Therefore, by optimizing the control signals and incorporating them into programmable logic devices such as the AmPAL22V10, the IC count can be dramatically reduced.

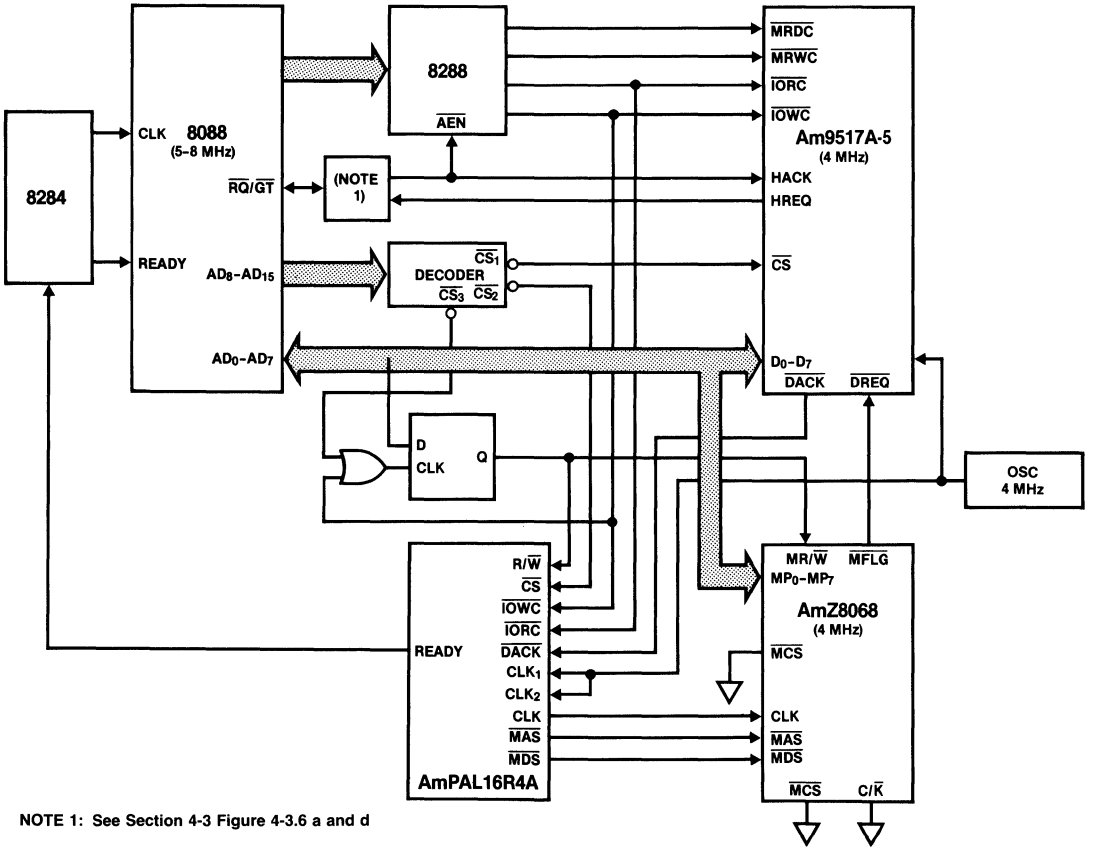
5.3.2 8088 AND AmZ8068 DATA CIPHERING PROCESSOR INTERFACE

Figure 5-3.2a shows the CPU-DMA interface. The CPU is operating in Maximum Mode. The bus arbitration handshake of the DMA controller (HREQ and HACK) must be translated into the Bus Request/Grant handshake of the 8088 CPU.

If the CPU is programmed to operate in Minimum Mode, both devices have the same bus arbitration handshake. The HREQ and HACK of the DMA controller can be connected directly to the corresponding pins of the CPU (HREQ to HACK).

The central part of this interface is a PAL device, which has been programmed for the 8088 CPU timing. The PAL equation for this interface is shown at the end of this section. The Chip Select 2 ($\overline{CS}_2$) input of the PAL device must be stable during the entire I/O transfer. This is guaranteed by decoding $\overline{CS}_2$ from the latched address/data bus of the 8088 (A_0-A_{15} in Figure 5-3.2a).

Master Port Read/Write is latched in the D Flip-Flop. It is clocked in an output operation with $\overline{CS}_3$ active. One of the data lines is latched in to define the status on the MR/$\overline{W}$ input. This is necessary because the DCP requires a set-up time of 100 ns of MR/$\overline{W}$ to the Data Strobe. Generation of MR/$\overline{W}$ for each cycle of a high-speed data transfer session of the DMA controller would extend each cycle and slow down the maximum throughput. This logic cannot be integrated into the PAL device because of the flip-flop's asynchronous clock.



NOTE 1: See Section 4-3 Figure 4-3.6 a and d

02188A-54

Figure 5-3.2a 8088-Am9517-AmZ8068 Interface

Before executing an access to the DCP, the CPU must latch the MR/$\overline{W}$. The transfer itself is evaluated in a two-cycle operation.

Master Port Address Strobe ($\overline{MAS}$) is only generated if the CPU executes an output instruction to a specific I/O address ($\overline{CS}_2$ active, A_0=Low) (Figure 53.2b) Address Latch Enable of the CPU (ALE) cannot be used for the generation of $\overline{MAS}$ because the CPU must set up the DCP for data transfer before a DMA transfer session is started. The DCP is set up by putting out a 00_H (data register address) to the I/O address mentioned above.

Figures 5-3.2c and 5-3.2d show data read and write cycles. Figure 5-3.2e shows DMA data read and writes cycles.

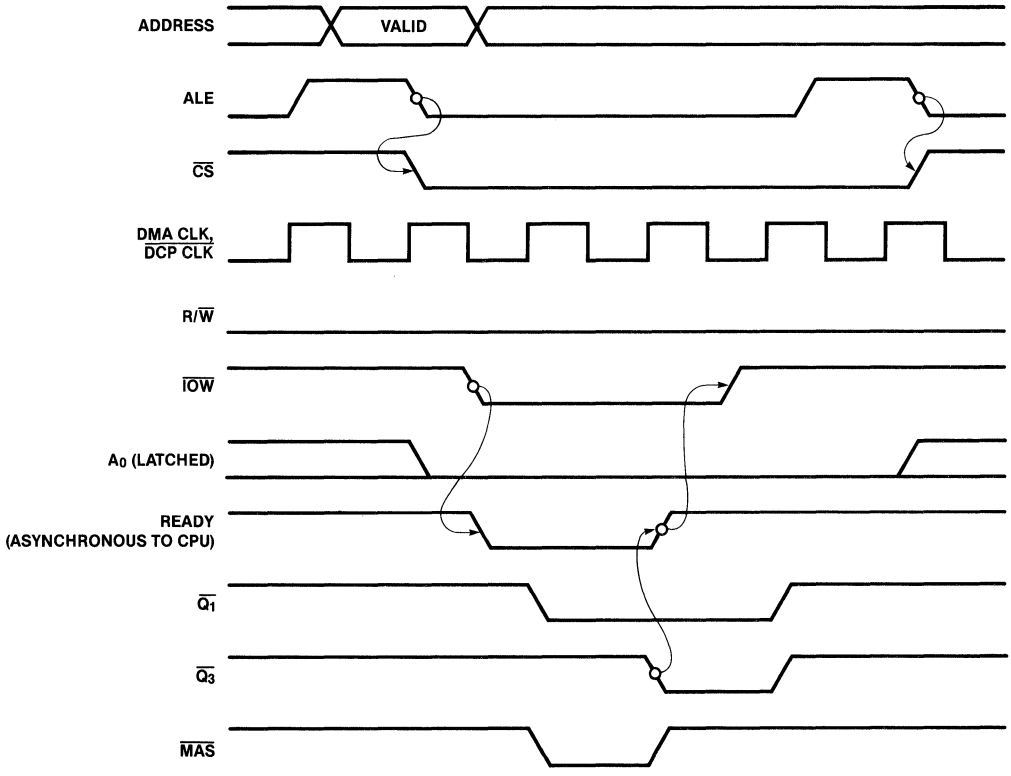
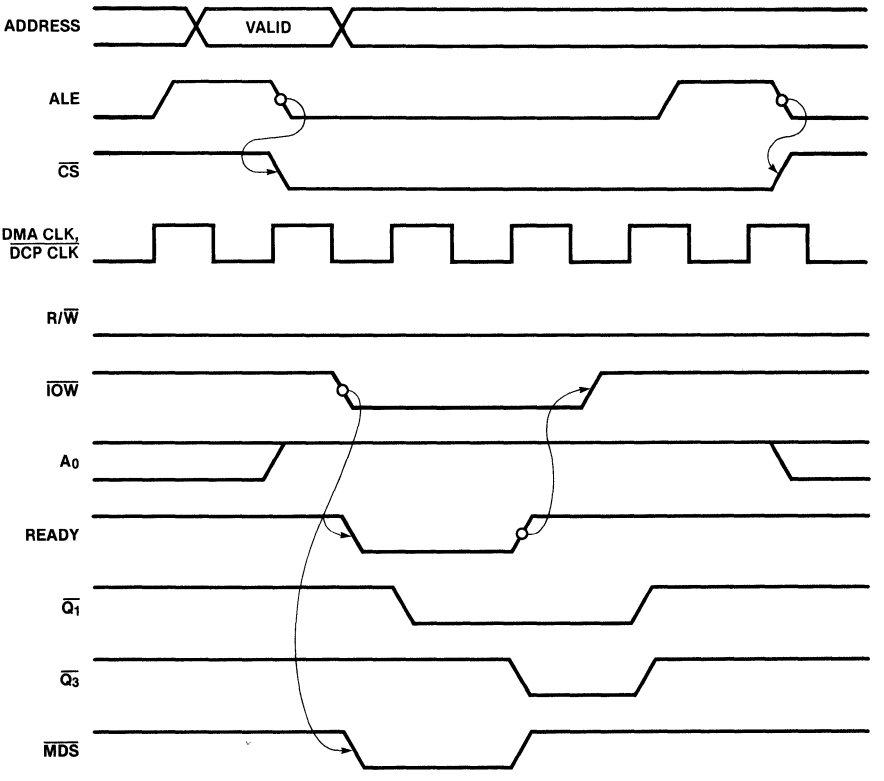
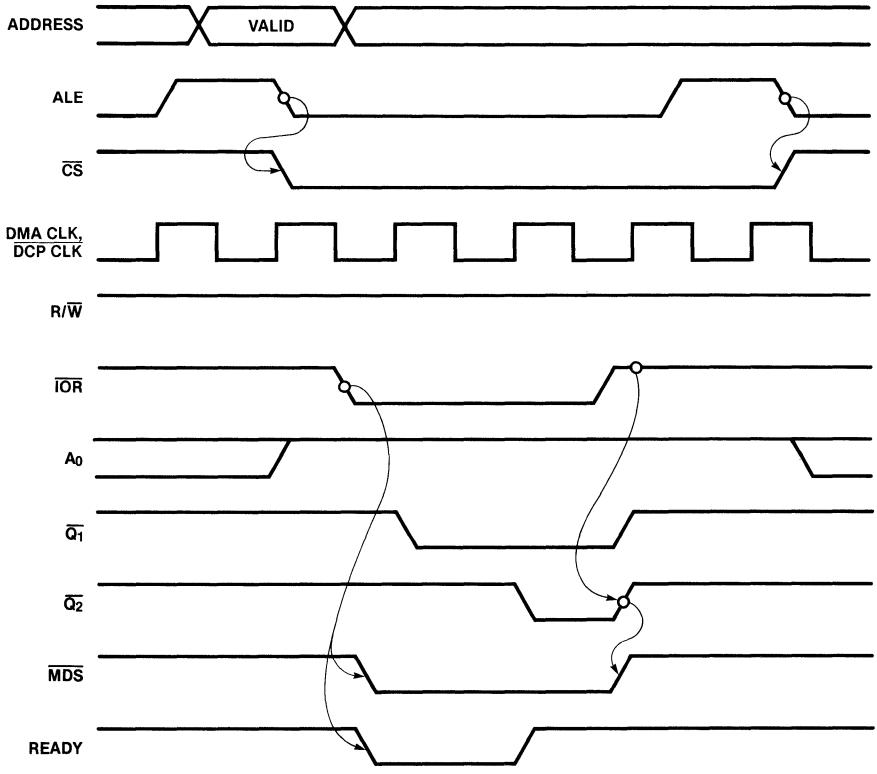


Figure 5-3.2b Address Latch Cycle Timing (CPU-DCP)



02188A-56

Figure 5-3.2c Data Write Cycle Timing (CPU-DCP)



02188A-57

Figure 5-3.2d Data Read Cycle Timing

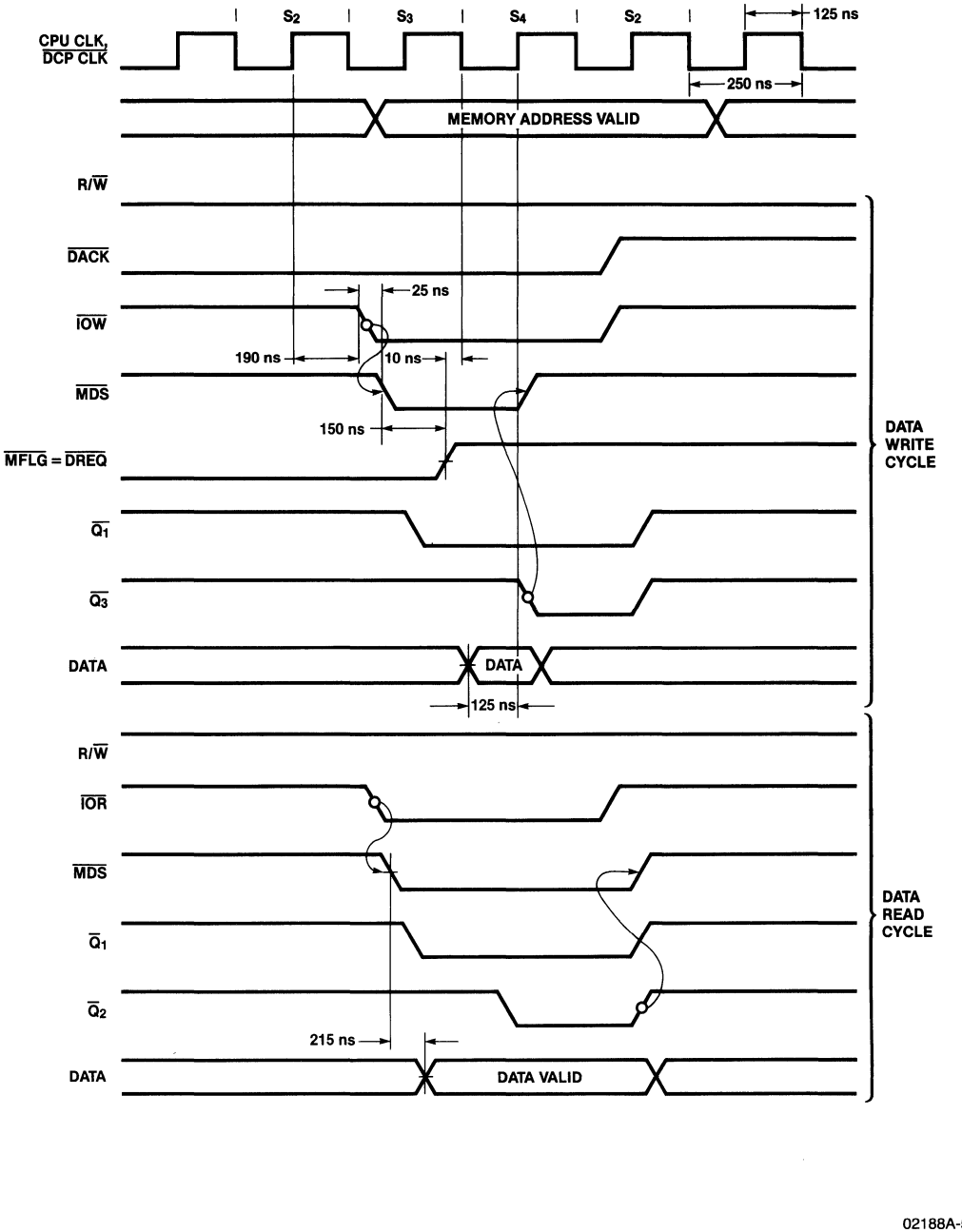


Figure 5-3.2e DMA-DCP Timing Diagram

```
PAL16R4                                        PAL DESIGN SPECIFICATION
DCP049                                         JUERGEN STELBRINK  8-12-83
8088- AM9517(DMA)- AMZ8068(DCP)  INTERFACE DEVICE
ADVANCED MICRO DEVICES

CLK1   CLK2   /CS    /IOR   /IOW   A0      RW      /DACK   NC    GND
/OE    /MDS   READY  /Q1    /Q2    /Q3     /MAS    NC      CLK   VCC

MAS    :=   IOW*/IOR*CS*/A0*/Q3*/MAS    ; MASTER PORT ADDRESS STROBE

Q1     :=   CS*IOR*/IOW*RW*/Q2    +
            CS*IOW*/IOR*/RW*/Q3   +
            DACK*IOR*/IOW*RW*/Q2  +
            DACK*IOW*/IOR*/RW*/Q3

Q2     :=   CS*IOR*/IOW*RW*Q1     +
            CS*IOR*/IOW*RW*Q2     +
            DACK*IOR*/IOW*RW*Q1   +
            DACK*IOR*/IOW*RW*Q2

Q3     :=   CS*IOW*/IOR*/RW*Q1    +
            CS*IOW*/IOR*/RW*Q2    +
            DACK*IOW*/IOR*/RW*Q1  +
            DACK*IOW*/IOR*/RW*Q2

MDS     =   CS*A0*IOR*/IOW*RW      +    ; MASTER PORT READ
            DACK*IOR*/IOW*RW       +
            Q2*A0                  +
            CS*A0*IOW*/IOR*/RW*/Q3+     ; MASTER PORT WRITE
            DACK*IOW*/IOR*/RW*/Q3

/READY =   CS*/A0*IOW*/IOR*/RW*/Q3+    ; ADDRESS LATCH CYCLE
            CS*A0*IOW*/IOR*/RW*/Q3 +    ; DATA WRITE CYCLE
            CS*A0*IOR*/IOW*RW*/Q2

/CLK   =   CLK2                        ; DCP CLOCK


FUNCTION TABLE

CLK1 CLK2 /CS /IOR /IOW /DACK A0 RW   CLK /MAS /MDS READY /Q1 /Q2 /Q3

;                  /                      R
; C  C     /  /  D           /  /  E
; L  L  /  I  I  A        C  M  M  A  /  /  /
; K  K  C  O  O  C  A  R  L  A  D  D  Q  Q  Q
; 1  2  S  R  W  K  0  W  K  S  S  Y  1  2  3      COMMENT
-------------------------------------------------------------------
; CLOCK GENERATION
  X  L  X  X  X  X  X  X    H  X  X  X  X  X  X
  X  H  X  X  X  X  X  X    L  X  X  X  X  X  X
;  ADDRESS LATCH
  C  X  H  H  H  H  L  L    X  H  H  H  H  H  H   ; CPU
  X  X  L  H  L  H  L  L    X  H  H  L  H  H  H
```

```
  C  X  L  H  L  H  L  L     X  L  H  L  L  H  H
  C  X  L  H  L  H  L  L     X  H  H  H  L  H  L
  C  X  H  H  H  H  L  L     X  H  H  H  H  H  H
; READ DATA
  X  X  H  H  H  H  H  H     X  H  H  H  H  H  H    ; CPU
  X  X  L  L  H  H  H  H     X  H  L  L  H  H  H
  C  X  L  L  H  H  H  H     X  H  L  L  L  H  H
  C  X  L  L  H  H  H  H     X  H  L  H  L  L  H
  C  X  L  L  H  H  H  H     X  H  L  H  H  L  H
  C  X  H  H  H  H  H  H     X  H  H  H  H  H  H
  X  X  H  L  H  L  X  H     X  H  L  H  H  H  H    ; CYCLE S3 (DMA)
  C  X  H  L  H  L  X  H     X  H  L  H  L  H  H
  C  X  H  L  H  L  X  H     X  H  L  H  L  L  H    ; CYCLE S4
  C  X  H  H  H  H  X  H     X  H  H  H  H  H  H    ; CYCLE S2
; WRITE DATA
  X  X  L  H  L  H  H  L     X  H  L  L  H  H  H    ; CPU
  C  X  L  H  L  H  H  L     X  H  L  L  L  H  H
  C  X  L  H  L  H  H  L     X  H  H  H  L  H  L
  C  X  H  H  H  H  H  L     X  H  H  H  H  H  H
  X  X  H  H  L  L  H  L     X  H  L  H  H  H  H    ; CYCLE S3 (DMA)
  C  X  H  H  L  L  H  L     X  H  L  H  L  H  H
  C  X  H  H  L  L  H  L     X  H  H  H  L  H  L    ; CYCLE S4
  C  X  H  H  H  H  H  L     X  H  H  H  H  H  H    ; CYCLE S2
; INVALID CYCLES
  X  X  L  L  L  H  H  H     X  H  H  H  H  H  H
  X  X  L  L  H  H  H  L     X  H  H  H  H  H  H
  X  X  L  H  L  H  H  H     X  H  H  H  H  H  H
-----------------------------------------------------------------------
```

DESCRIPTION:

THIS PAL GENERATES ALL NECESSARY BUS CONTROL SIGNALS, TO INTERFACE A 8088 CPU AND A AM9517 DMA CONTROLLER TO THE AMZ8068 DATA CIPHERING PROCESSOR. THE MAXIMUM SYSTEM CLOCK FOR THE DMA CONTROLLER AND THE DCP IS 4 MHZ, THE SYSTEM CLOCK OF THE CPU CAN BE UP TO 8 MHZ. THE DEVICES ARE WORKING ASYNCHRONOUSLY.

INPUT SIGNALS:

CLK1, CLK2 — DMA CLOCK

/CS — CHIP SELECT FOR THE DCP, GENERATED BY A DECODER LOGIC

/IOR — INPUT/ OUTPUT READ

/IOW — INPUT/ OUTPUT WRITE

A0 — LEAST SIGNIFICANT BIT OF THE Z80 ADDRESS BUS TO SELECT THE TYPE OF OPERATION:
A0 = LOW SELECT DCP REGISTER FOR NEXT DATA CYCLES (ADDRESS LATCH)
A0 = HIGH READ OR WRITE INTERNAL REGISTER

(DATA TRANSFER TO CONTROL, MODE, INPUT OR OUTPUT REGISTER)

/DACK DMA ACKNOWLEDGE FROM DMA CONTROLLER, TREATED AS /CS=LOW AND A0=HIGH

RW READ/ WRITE SIGNAL STORED IN A EXTERNAL LATCH, TO ALLOW A DMA OPERATION WITHOUT WAIT STATES. THIS SOLVES THE PROBLEM OF THE SETUP TIME OF MR/W OF THE MASTER PORT TO MDS GOING ACTIVE. THE STATUS OF THIS SIGNAL MUST AGREE WITH /IOR OR /IOW OR THE PAL GENERATES NO STROBES.

OUTPUT SIGNALS:

CLK INVERTED DMA CLOCK FOR THE DCP

/MAS MASTER PORT ADDRESS LATCH ENABLE, ACTIVE DURING ADDRESS LATCH CYCLES TO LATCH THE REGISTER ADDRESS ON MP1 AND MP2 (2 LINES OF THE MASTER PORT BUS) AND THE STATE OF /MCS IN. THE DCP STORES INTERNALLY THE ADDRESS AND CHIP SELECT TO THE NEXT ADDRESS LATCH CYCLE

/MDS MASTER PORT DATA STROBE, TO TIME DCP DATA TRANSFERS

/Q1, /Q2, /Q3 INTERNAL USED STATE SIGNALS (DO NOT CONNECT). Q1 IS ACTIVE 2 CLOCK CYCLES IN ALL CYCLES. IT IS USED TO GENERATE THE DELAYED Q2 AND Q3. Q2 IS ACTIVE IN A DATA READ CYCLE. IT ALLOWS /MDS TO BE ACTIVE UNTIL /IOR HAS GONE INACTIVE. Q3 IS ACTIVE IN AN ADDRESS LATCH OR DATA WRITE CYCLE. Q3 DISABLES READY AND /MDS IN THE SECOND HALF OF THE CYCLE.

5.3.3 8088 AND Am9513A SYSTEM TIMING CONTROLLER

This interface is very similar to the 8086 to Am9513A interface except that the Data bus is 8-bit wide instead of 16-bit wide. Figure 5-3.3a shows the block diagram for this interface. The dip-switch at the B-port of the Am29809 allows the designer to decode any I/O address for the application. If a fixed address decode is required, the B port lines can be selectively grounded or left open, (as the part has internal pull-up). Note that Word Writes should not be used for this interface as they will violate Am9513A recovery time specification.

5.3.4 8088 TO Am9516 UDC INTERFACE

Figure 5-3.4a shows the Data bus and Control Interface between an 8088 microprocessor and an Am9516. This interface is accomplished by using an AmPAL22V10, a 74LS161 counter, an Am2952 bidirectional I/O port, and an Am2949 bidirectional transceiver. Figures 5-3.4c and 5-3.4d show the PAL device timing for both the Bus Master and the Bus Slave cases.

In this design, certain simplifying constraints have been made. Word operations are only allowed during command chaining operations when the Am9516 is Bus Master. During Slave Write operations, the first byte output to the Am9516 must have an odd address, and the following second byte an even address. Conversely, during a Slave Read cycle, the first byte read from the Am9515 must be at an even address and the second at the next higher odd address. Furthermore, for both slave and master operations, the system must use the latched A_O (LA_O) from the AmPAL22V10 as its sole A_O. That is, this LA_O must circumvent any address latching to memory and so is to be used directly.

As can be seen from the figures, the AmPAL22V10 manipulates the control signals to the transceivers and latches so that bytes can be funneled into words during Am9516 Command Chaining operations and Slave Writes, and words can be funneled into bytes during Am9516 Slave Reads. Also, LAO is alternately toggled in order to provide the dual address (16-bits) during chaining operations.

Figure 5-3.4e shows the PLPL equations and resulting sum-of-products equations for the design (PLPL is a higher level "C-like" language for PAL devices). For those not familiar with "C" or Pascal, the intermediate sum-of-products form shows the equations in Boolean form.

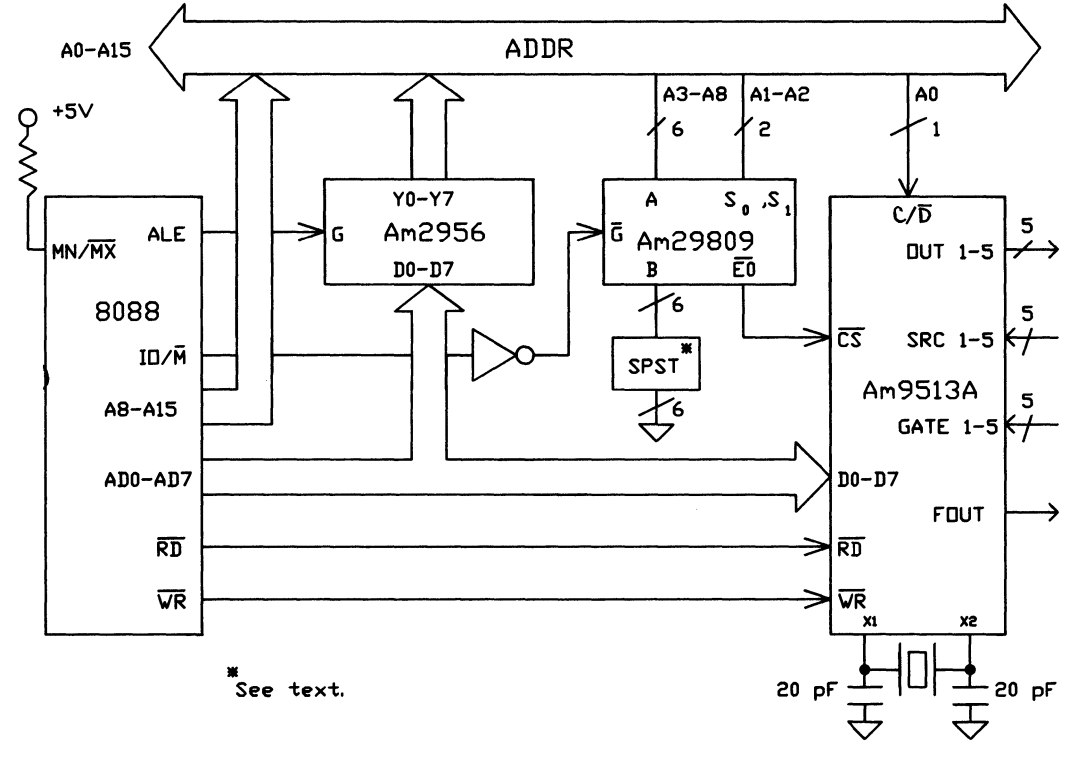
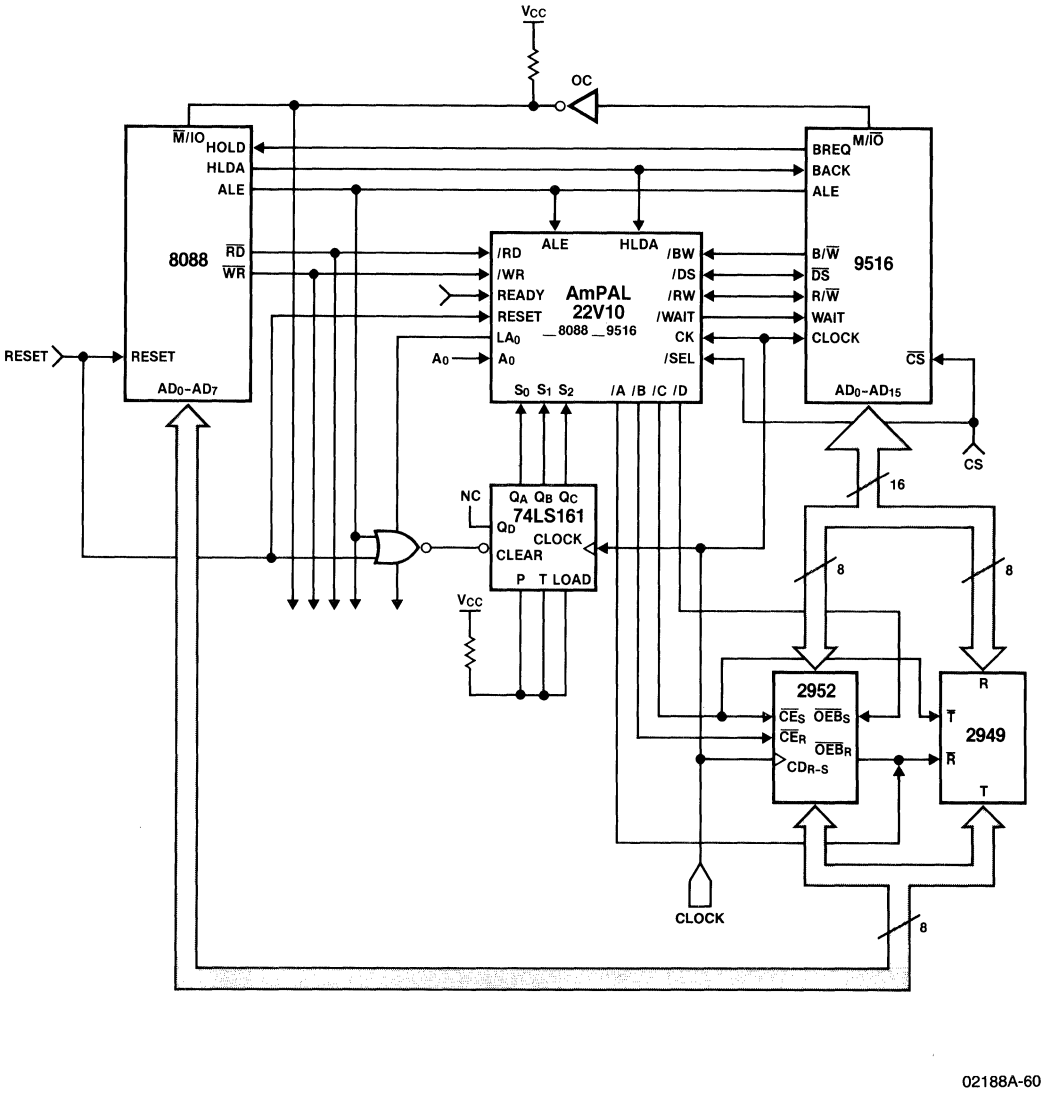


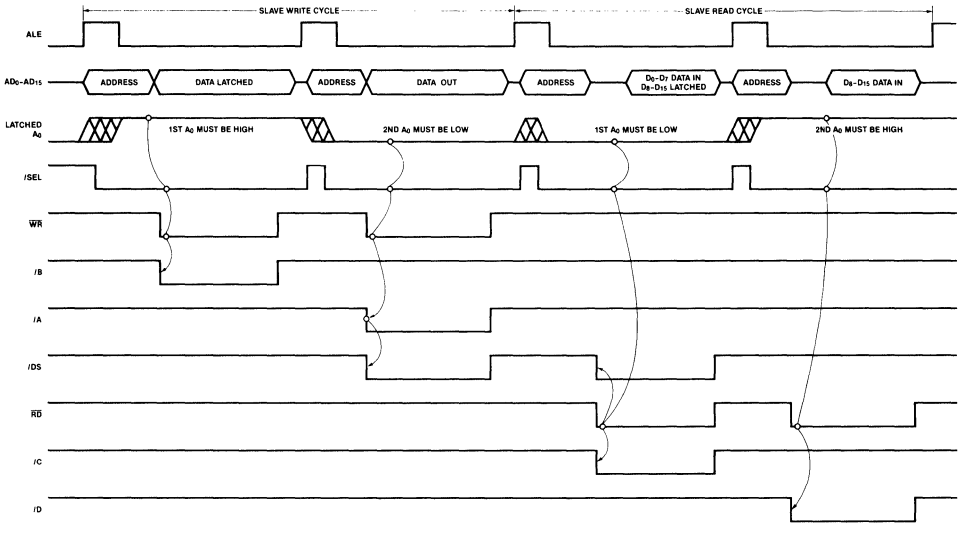
Figure 5-3.3a

02188A-59



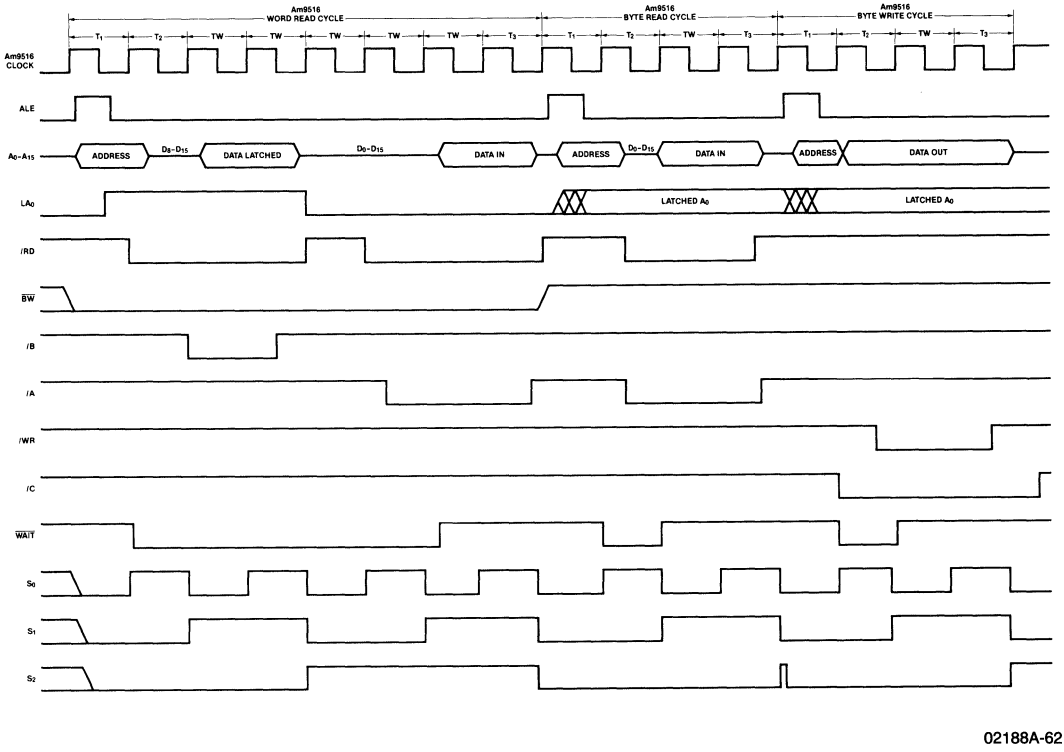
02188A-60

Figure 5-3.4a The Am9516 UDC to 8088 CPU Data Bus and Control Interface



02188A-61

Figure 5-3.4c The Am9516 UDC to 8088 CPU Bus Slave Timing



02188A-62

Figure 5-3.4d The Am9516 UDC to 8088 CPU Bus Master Timing

```
DEVICE _8088_9516 (pal22V10)

"This is a PLPL file to implement an interface between the 8-bit
8088 and the 16-bit Am9516.  This design assumes that the Am9516
is programmed for byte operations and so only accesses 16-bit
words during chaining.                          -James Williamson"

PIN

        CK      = 1                     /RD    = 23
        S[0:2]  = 2:4                   /WR    = 22
        A0      = 5                     LA0    = 21
        /SEL    = 6                     /DS    = 20
        ALE     = 7                     /RW    = 19
        HLDA    = 8                     /WAIT  = 18
        /BW     = 9                     /A     = 17
        READY   = 10                    /B     = 16
        RESET   = 11                    /C     = 15
                                        /D     = 14;
BEGIN

     IF (RESET) THEN ARESET();
"================================================================="
"This section defines the wiggles when the Am9516 is Bus Master"
"================================================================="
     IF (HLDA)  THEN ENABLE();
     IF (/S[2] * HLDA) THEN BEGIN
          IF (/S[1] * /S[0]) THEN
                    LA0  =  /CK  * BW + /BW * A0 * ALE +
                            /BW  * LA0 * /ALE                   ;
          ELSE
                    LA0  =  BW  + /BW * A0 * ALE +
                            /BW * LA0 * /ALE                    ;

     END;
     IF (HLDA) THEN
          CASE (S[2:0])
               BEGIN
               1) BEGIN
                    RD   = /RW * DS                             ;
                    A    = /BW * /RW * /CK                      ;
                    WR   = /BW * RW  * DS                       ;
                    C    = /BW * RW                             ;
                    WAIT = 1                                    ;
               END;

               2) BEGIN
                    RD   = /RW * DS                             ;
                    B    = BW                                   ;
                    A    = /BW * /RW                            ;
                    WR   = /BW * RW * DS                        ;
                    C    = /BW * RW                             ;
                    WAIT = BW                                   ;
               END;
```

Figure 4-3.4e

```
            3) BEGIN
                 RD    = /RW * DS * B                    ;
                 B     = BW * CK                         ;
                 A     = /BW * RD                        ;
                 WR    = /BW * RW * DS                   ;
                 C     = /BW * RW                        ;
                 WAIT  = BW                              ;
            END;

            5) BEGIN
                 RD    = /RW * DS                        ;
                 A     = /BW * /CK                       ;
                 WAIT  = BW                              ;
            END;

            6) BEGIN
                 RD    = /RW * DS                        ;
                 A     = BW                              ;
            END;

            7) BEGIN
                 RD    = /RW * DS                        ;
                 A     = /RD                             ;
            END;

       END;

"================================================================="
"This  section  defines the wiggles when the 8088 is Bus  Master"
"================================================================="

     BEGIN

          LA0 = A0 * ALE * SEL +
                LA0 * /ALE * SEL                         ;
          B   = LA0 * WR * SEL                           ;
          A   = /LA0 * WR * SEL                          ;
          DS  = A +
                /LA0 * RD * SEL                          ;
          C   = /LA0 * RD * SEL                          ;
          D   = LA0 * RD * SEL                           ;
     END;

END.
```

Figure 4-3.4e (continued)

5.3.5 8088 TO Am9517A INTERFACE

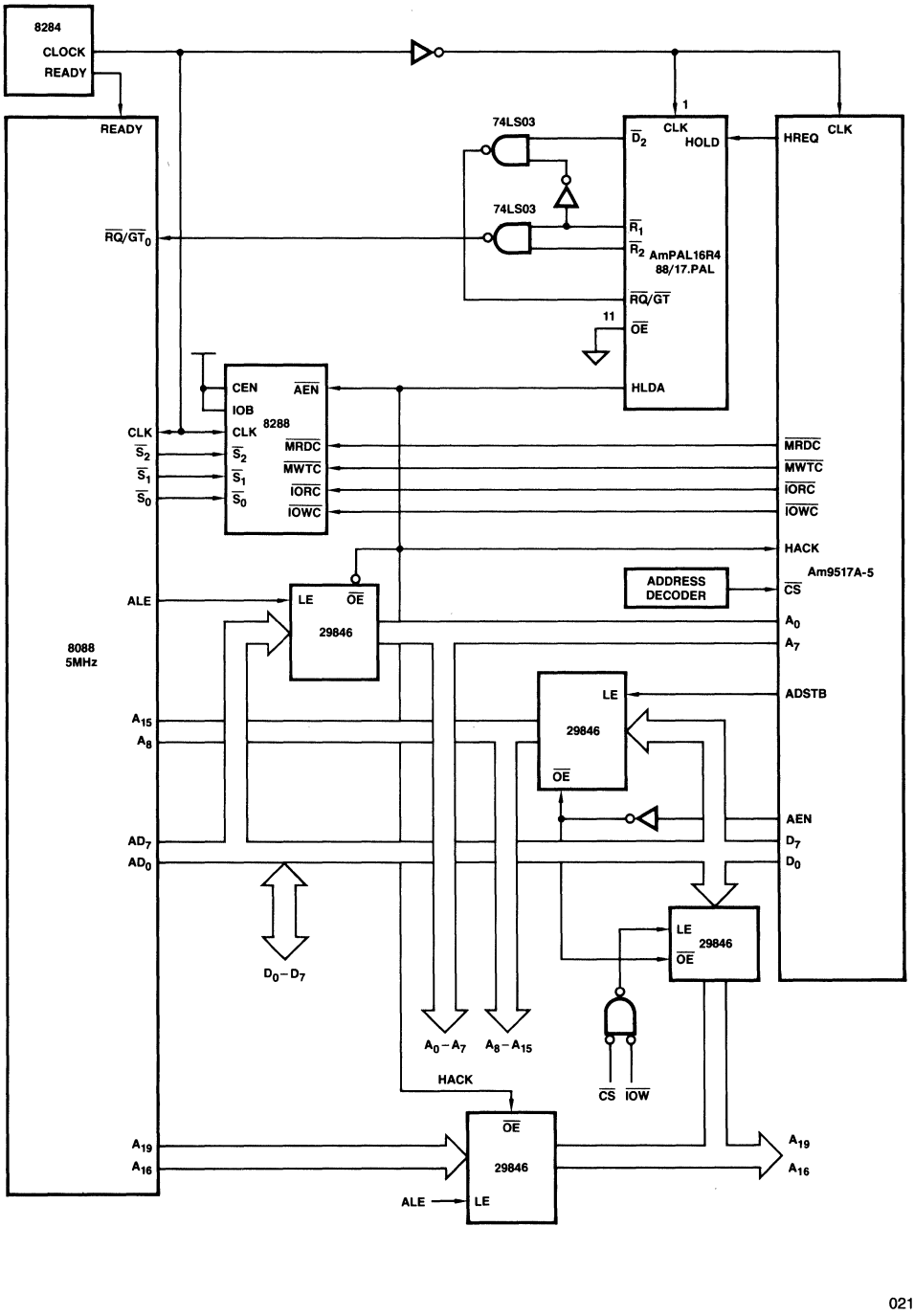
This interface assumes the Am9517A to be closely coupled to the 8088 (i.e. on the same board). Additional buffers may be necessary for larger systems. The upper address latch is really a page register, which the CPU loads. This provides the upper address when the Am9517A is bus master. This page register is necessary since the Am9517A only supplies 16 bits of address compared to the 8088's 20 bits.

Diagrams in Figures 5-3.5a and 5-3.5b show the 8088 to Am9517A interface in both Max. and Min. modes. When operating the 8088 in Max. mode, the 8288 can be used to produce MRDC, MWTC, IORC, and IOWC without the additional logic required in the Min. mode. A problem with Max. mode operation is the need to convert HREQ and HACK into the RQ/GT protocol. Figure 5-3.5c shows the PAL equation for the PAL16R4 used in the Max. mode.

Note that even if both the 8088 and Am9517A are 5 MHz parts, a common clock cannot be used since the 8088 clocks duty cycle does not meet Am9517A requirements. Sometimes adding another oscillator can be saved, as in the case of an 8 MHz CPU. By dividing the clock by 2, the DMA can be operated at 4 MHz.

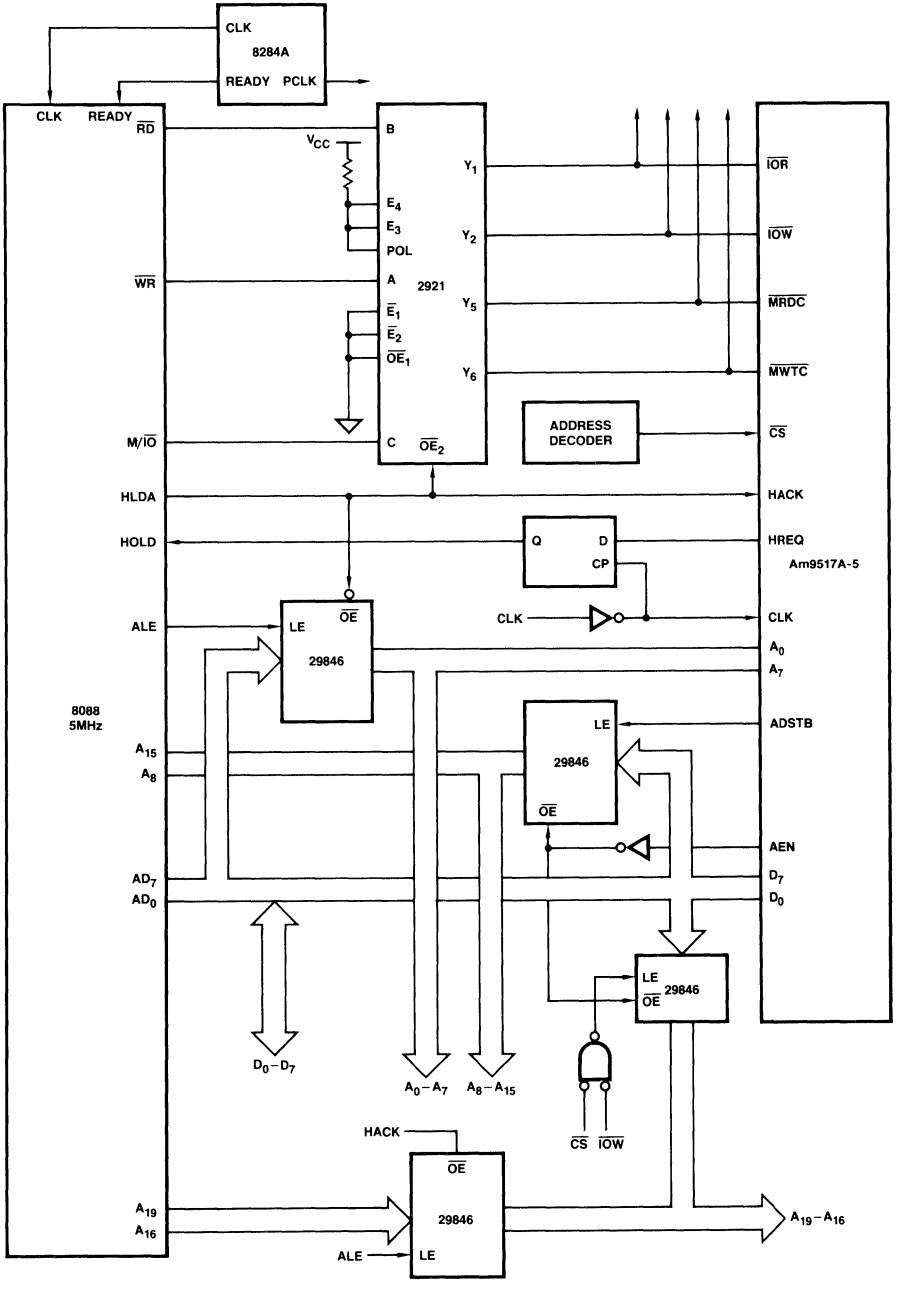
5.3.6 8088 AND Am9519A INTERRUPT CONTROLLER INTERFACE

See Chapter 4



02188A-63

Figure 5-3.5a



02188A-64

Figure 5-3.5b

Type Am9517 PAL
PAL16R4

8088 to Am9517A Interface

CLK /RQGT HOLD NC NC NC /NC /NC /NC GND
/OE /NC /NC /HLDA /D_2 /R_2 /R_1 /NC /NC V_{CC}

R_1: =HOLD
R_2: =/R_1
D_2: =R_1
/HLDA: =/R_1+/D_2*/HLDA+/RQGT*/HLDA

Figure 5-3.5c

6.0 INTERFACING TO THE 68000

6.1 68000 OVERVIEW

The 68000 has an asynchronous, 16-bit, bidirectional, data bus. Data types supported by the 68000 are: bit data, integer data of 8, 16, or 32 bit, 32 bit addresses and binary coded decimal data. It can transfer and accept data in either words or bytes. The $\overline{DTACK}$ input indicates the completion of a data transfer. When the processor recognizes $\overline{DTACK}$ during a read cycle, the data is latched and the bus cycle terminates. When $\overline{DTACK}$ is recognized during a write cycle, the bus cycle also terminates. An active transition of $\overline{DTACK}$ indicates the termination of a data transfer on the bus. All control and data lines are sampled during the 68000's clock high time. The clock is internally buffered, which results in some slight differences in the sampling and recognition of various signals. The 68000 mask sets prior to CC1 and allows $\overline{DTACK}$ to be recognized as early as S2, and all devices allow $\overline{BERR}$ or $\overline{DTACK}$ to be recognized in S4,S6, etc., which terminates the cycle. If the required setup time is met during S4, $\overline{DTACK}$ will be recognized during S5 and S6, and data will be captured during S6. $\overline{DTACK}$ signal is internally synchronized to allow for valid operation in an asynchronous system. If an asysnchronous control signal does not meet the required setup time, it is possible that it may not be recognized during that cycle. Because of this, synchronous systems must not allow $\overline{DTACK}$ to precede data by more than 40 to 240 nanoseconds, depending on the speed of the particular processor. I/O is memory-mapped, i.e., there are no special I/O control signals, any peripheral is treated as a memory location.

DMA: This Bus is requested by activating the $\overline{BR}$ input of the 68000. Bus Arbitration is started by the $\overline{BG}$ output going active. The Bus is available when $\overline{AS}$ becomes inactive. The requesting device must acknowledge bus mastership by activating the $\overline{BGACK}$ input to the CPU.

The 23-bit address (A_1...A_{23}) is on a unidirectional, three-state bus, and can address 8 M words (16 M bytes) of memory or I/O. It provides the address for bus operation during all cycles, except the interrupt cycles. During interrupt cycles, address lines A1, A2 and A3 provide information about the level of interrupt being serviced. Instead of A_0 and BYTE/$\overline{WORD}$, there are two separate data strobe lines for the two bytes in a word. A note of caution here, the 68000 treats the MSB of the lower byte as an even byte, or word address. The same goes with processors such as the Z8000. Processors such as the 8086 treats the lower byte as the odd byte.

Interrupt is requested by activating any combination of the interrupt inputs to the 68000 (IPL0...2), indicating the encoded priority level of the interrupt requester (inputs at or below the current processor priority are ignored). The 68000 automatically saves the status register, switches to supervisor mode, fetches a vector number from the interrupting device, and displays the interrupt level on the address bus. For interfacing with old 68000 peripherals, the 68000 issues an Enable signal at one-tenth of the processor clock frequency. There are a number of AMD proprietary third generation peripherals that can be interfaced to the 68000 CPU, to improve system performance. This chapter deals mainly with the interfacing of the 68000 and some of the AMD proprietary peripherals.

6.2.0 THE 68000 AND AmZ85XX PERIPHERALS

The Am8500 series of peripherals, as well as others, require $\overline{CS}$ to be valid before $\overline{RD}$ goes LOW. The timing of the 68000 does not guarantee this; therefore, it is necessary to delay the falling edge of $\overline{LDS}$ to meet this requirement. We are using the AmZ8530 here to illustrate the interface between the 68000 and an AmZ85XX device. The AmZ8530 Serial Communication Controller is an I/O device that offers user-programmable features for high end applications. It supports all advanced protocols: HDLC, SDLC, and IBM Bisync. It has a transmit rate of up to 1.5 Mbps. It has a programmable baud-rate generator which reduces chip count and cuts down on board real estate. Other features include built-in digital phase-lock-loop, auto echo, and local loopback.

The Am29809 Comparator and the Am29806 Comparator/Decoder provides high-speed address selection as well as an open collector acknowledge driver. This allows memories and peripherals to be conveniently wire ORed to the processor's $\overline{DTACK}$ pin. This interface does not provide a hardware reset therefore, it is necessary to do a dummy read to the control port before issuing a software reset. This insures the internal state machine is in the proper state to sequence through the Address Cycle and then the Data Cycle.

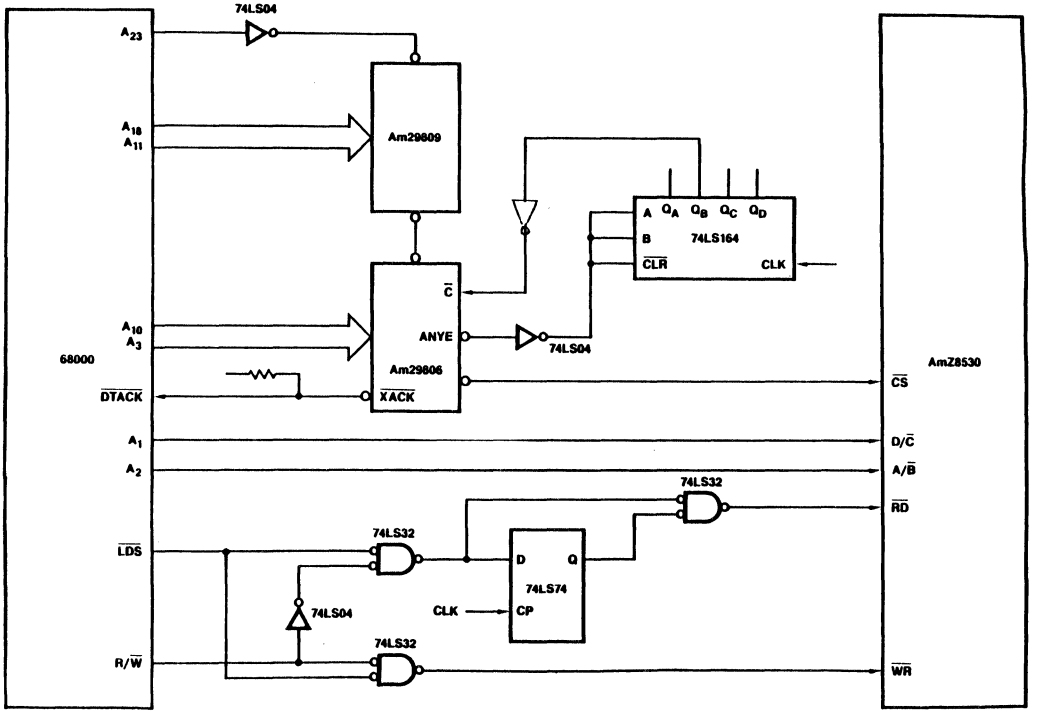
6.2.1 THE 68000 AND AmZ8530 WITHOUT INTERRUPTS

Implementation of an interface without interrupt is straightforward. $\overline{INTACK}$ must be tied High when not in use and the shift register provides a means for inserting Wait States.

Figure 6-2.1 shows the interface between the 68000 and the AmZ8530. The AmZ8530 SCC is the fastest available serial I/O in the market today. It supports all advanced protocols and has a number of user programmable features that provide design flexibility.

The data bus between the 68000 and AmZ8530 are directly linked together. The Am29806 decodes the address and generates CS for the peripheral and produces $\overline{ANYE}$, which is used by the 74LS164, to generate Wait States. The 74LS164 controls the number of wait states by the "$\overline{C}$" input which in turn controls the $\overline{DTACK}$ signal to the CPU. This allows $\overline{RD}$ and $\overline{WR}$ to obtain the required 400 ns width. A_1 generates the C/$\overline{D}$ input to the AmZ8530 while the remaining address lines are connected to the Am29809 address comparator.

The Am29809 Comparator and the Am29806 Comparator/Decoder provides high-speed address selection as well as an open collector acknowledge driver. This allows memories and peripherals to be conveniently wire-ORed to the processor's $\overline{DTACK}$ pin. This interface does not provide a hardware reset; therefore, it is necessary to do a dummy read to the control port before issuing a software reset.



02188A-65

Figure 6-2.1

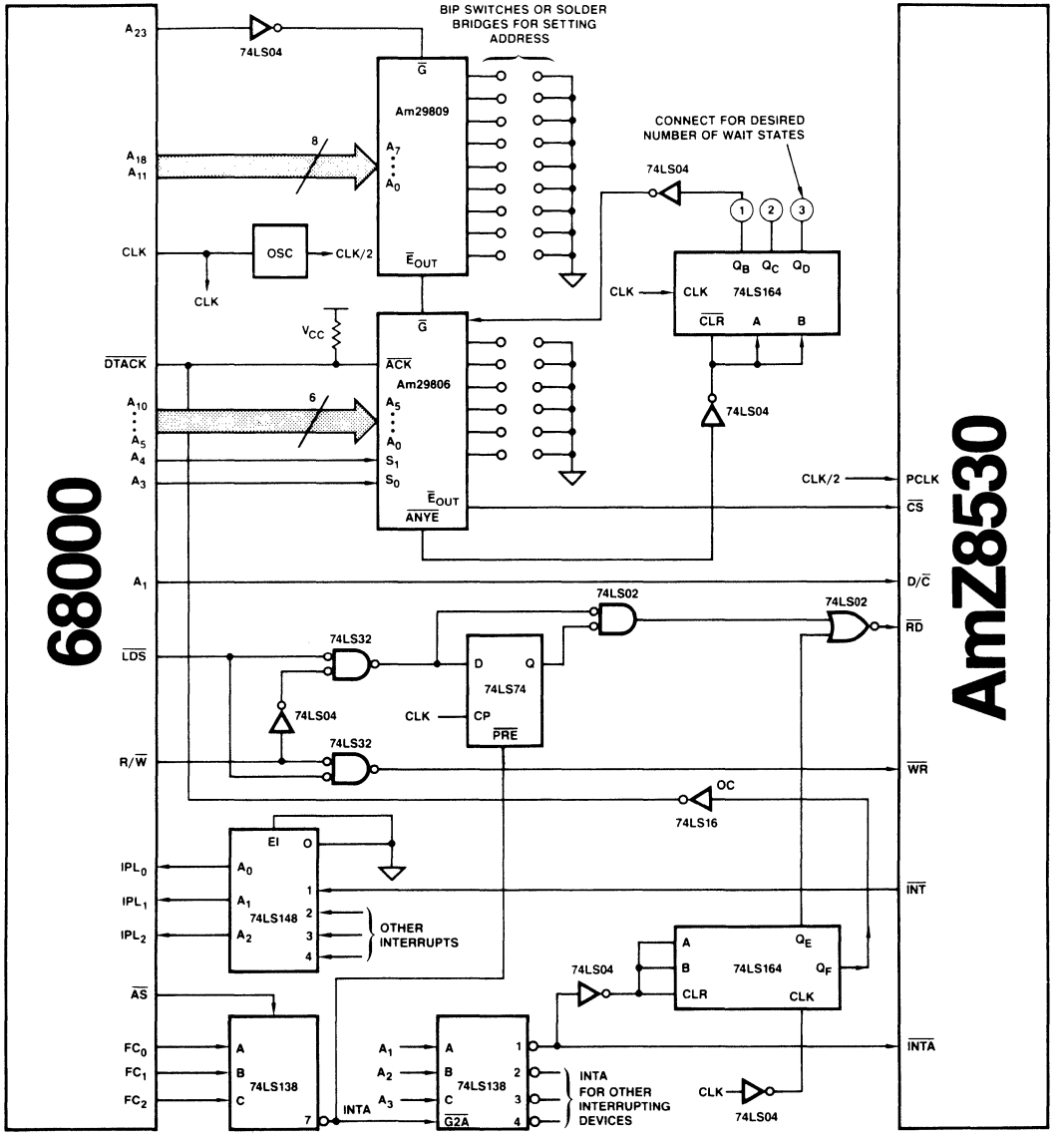
6.2.2 THE 68000 AND AmZ8530 WITH INTERRUPTS

This example addresses the problems of interfacing the 68000 and AmZ8530 with interrupts. The circuit configuration is basically the same as the design without interrupts.

The 74LS148 and the two 74LS138s assume there are other interrupting devices which are not compatible with the interrupt daisy-chain of the 85XX Family. The 74LS148 and one of the 74LS138 can be eliminated if this is not the case.

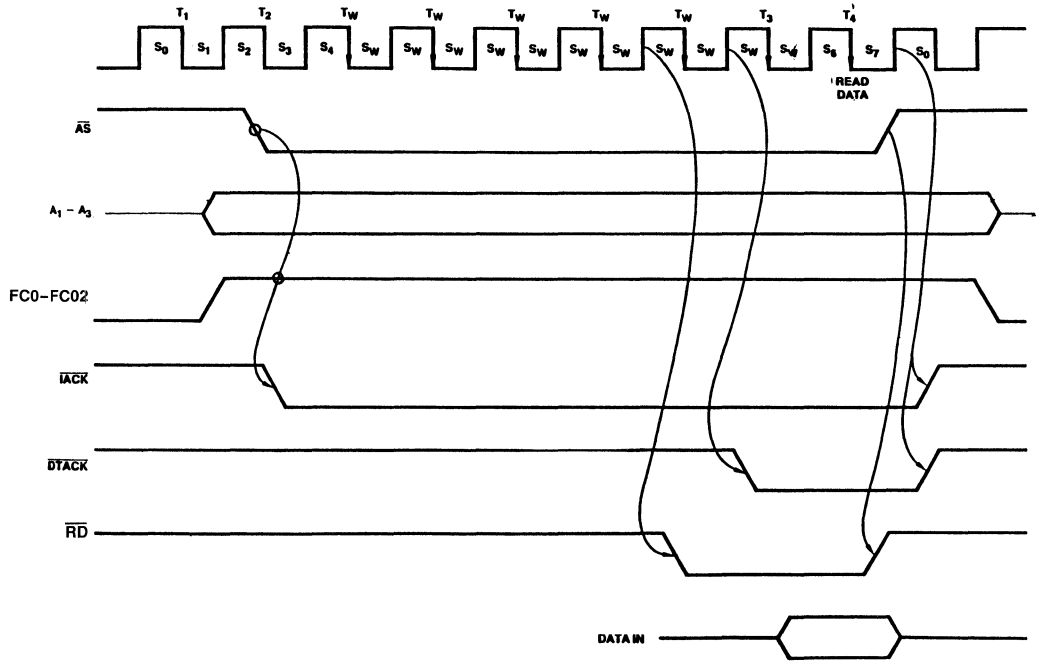
The first 74LS138 acts as a status decoder; it is gated with $\overline{AS}$ to de-glitch the outputs. The second 74LS138 decodes the Interrupt Acknowledge priority level, allowing a two-dimensional priority scheme. Daisy-chain can be used to resolve priority at any given priority level while the CPU resolves priority between levels.

The 74LS164 is added to generate the correct timing during an Interrupt Acknowledge cycle. It allows 5 CPU clocks for the daisy-chain to settle before it generates $\overline{RD}$ to put the vector onto the bus. The daisy-chain is implemented by using the IEI, IEO pins (not shown in the diagram) on the 8500 peripherals. The time allowed for the daisy-chain to settle is a function of the number of devices in the chain; thus, the allowance of 5 clocks used here is arbitrary. The 74LS164 also generates $\overline{DTACK}$. A block diagram of this interface is shown in Figure 6-2.2a. Timing diagram is followed in Figure 6-2.2b. It is more straightforward to use the Am9519A Interrupt Controller instead of the on-chip interrupt features. However, this approach does not allow the programmer to take advantage of some of the 85XX family's time-saving features.



02188A-66

Figure 6-2.2a Am8530 to 68000 Connection with Interrupts Using SSI and MSI



02188A-67

Figure 6-2.2b Interrupt Acknowledge Timing

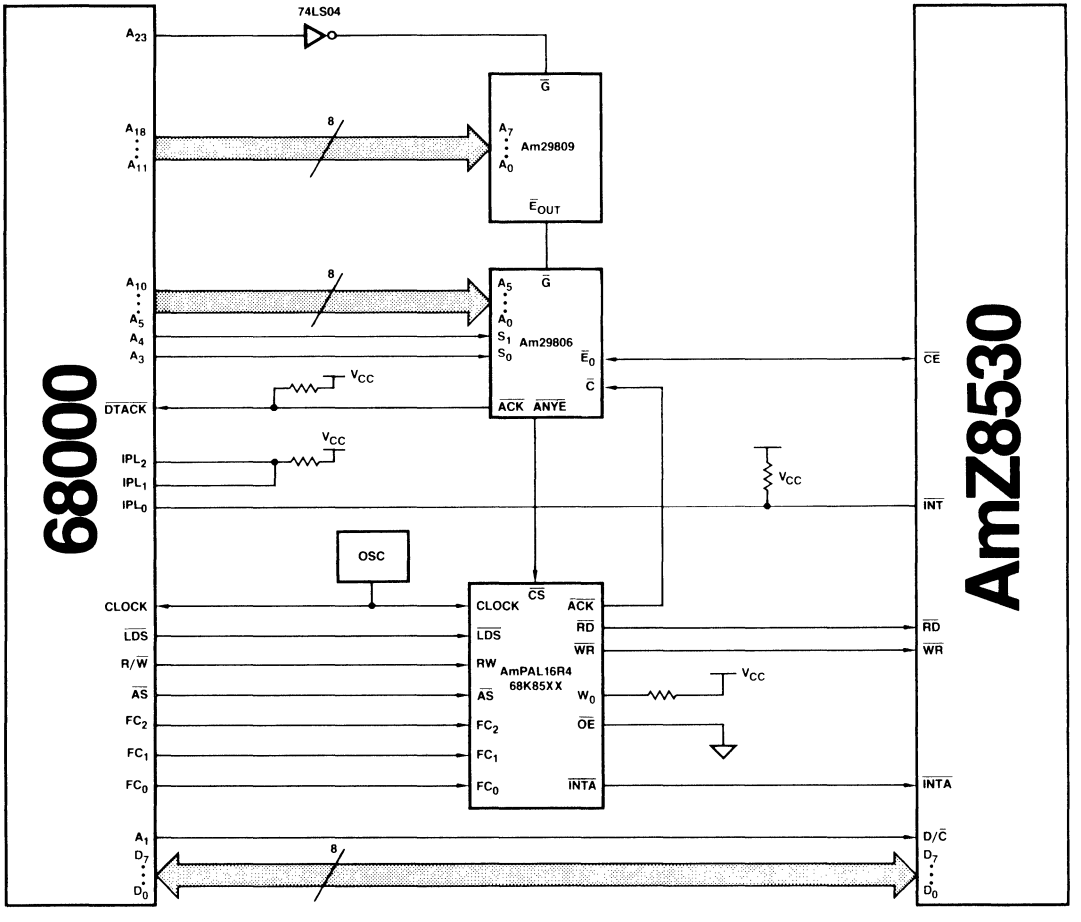
6.2.3 THE 68000 AND AmZ8530 WITH INTERRUPTS VIA A PAL DEVICE

This example shows how a Programmable Array Logic (PAL) device simplifies the task of interrupt generation compared to the MSI implementation. The block diagram for the interface via a PAL device is shown in Figure 6-2.3a. The timing diagram (Figure 6-2.3b) illustrates the Interrupt Acknowledge cycle. As in the other designs, $\overline{RD}$ is generated during Interrupt Acknowledge to place the vector on the bus.

The timing during register programming is not shown. The PAL device allows selection of one or two Wait States by making W_0 High or Low respectively. The table below shows the appropriate number of Wait States as a function of CPU speed.

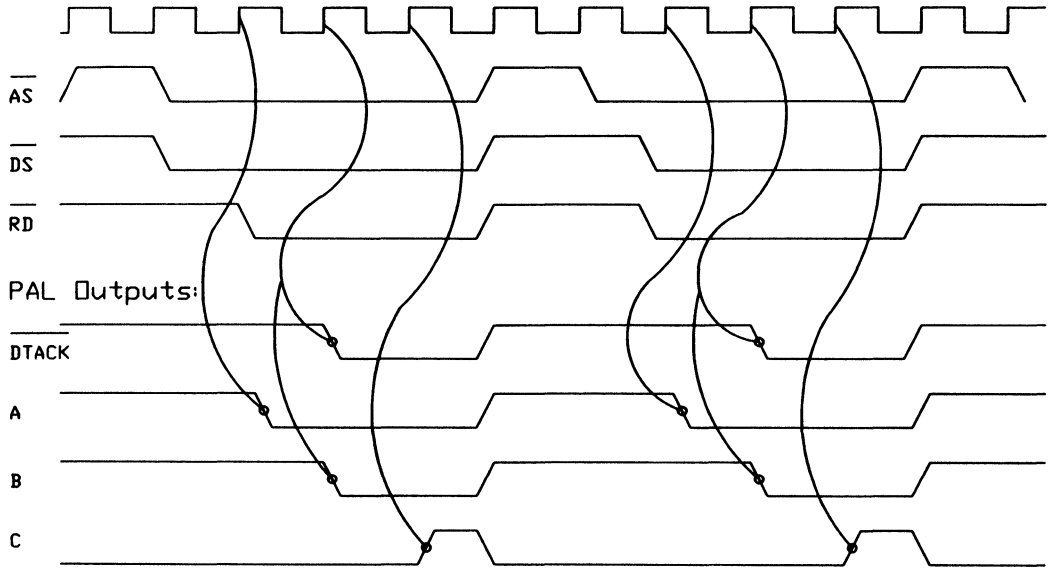
Part	CPU Speed	Wait States
85XX	4 MHz	1
85XX	6 MHz	2
85XXA	8 MHz	2
85XXA	10 MHz	2

PAL device equations are shown in Figure 6-2.3c.



02188A-68

Figure 6-2.3a 68000 to AmZ8530 Connection Using a PAL



02188A-69

Figure 6-2.3b

```
PAL16R4                    PAL DESIGN SPEC
PAT 002                    JOE BRCICH 9 SEPT 83
68000 TO 8500 OR 9500 PERIPHERALS
ADVANCED MICRO DEVICES

CLOCK  /CS  RW  /LDS  /W0  /AS  FC0  FC1  FC2  GND
/OE   /INTA  /ACK   /C  /B  /A  /DLDS  /RD  /WR  VCC

A := A*/B + B *C + /AS

B := A*/C + /A*C +
     /AS

C := /A*/B*AS + B*C*AS

DLDS := LDS

RD = LDS*DLDS*RW*/INTA + A*C*INTA*AS + A*/B*INTA*AS

WR = LDS * /RW

INTA = FC0*FC1*FC2*AS

ACK = /INTA*/A*/B*/C*/W0 + /INTA*/A*/B*C*W0 +
      INTA*/B*A  +  ACK * LDS

DESCRIPTION
        THIS PAL DEVICE INTERFACES 85XX TYPE PERIPHERALS TO THE
68000 MICRO PROCESSOR. IT INSERTS 1 OR 2 WAIT STATES AS
SELECTED BY /W0=0 IS ONE AND /W0=1 IS TWO WAIT STATES. FOUR
WAIT STATES ARE INSERTED DURING INTERRUPT ACKNOWLEDGE CYCLES.
ALSO THE RD OUTPUT GENERATED DURING INTA IS A FUNCTION OF THE
INTERNAL STATE MACHINE AND NOT A FUNCTION OF LDS. OE CAN BE
LEFT OPEN SINCE THE FLIP FLOP OUTPUTS ARE NOT USED DIRECTLY.
THE FALLING EDGE OF RD IS DELAYED IN ORDER TO GUARENTEE
THE CS TO RD SETUP TIME REQUIREMENTS.
```

Figure 6-2.3c

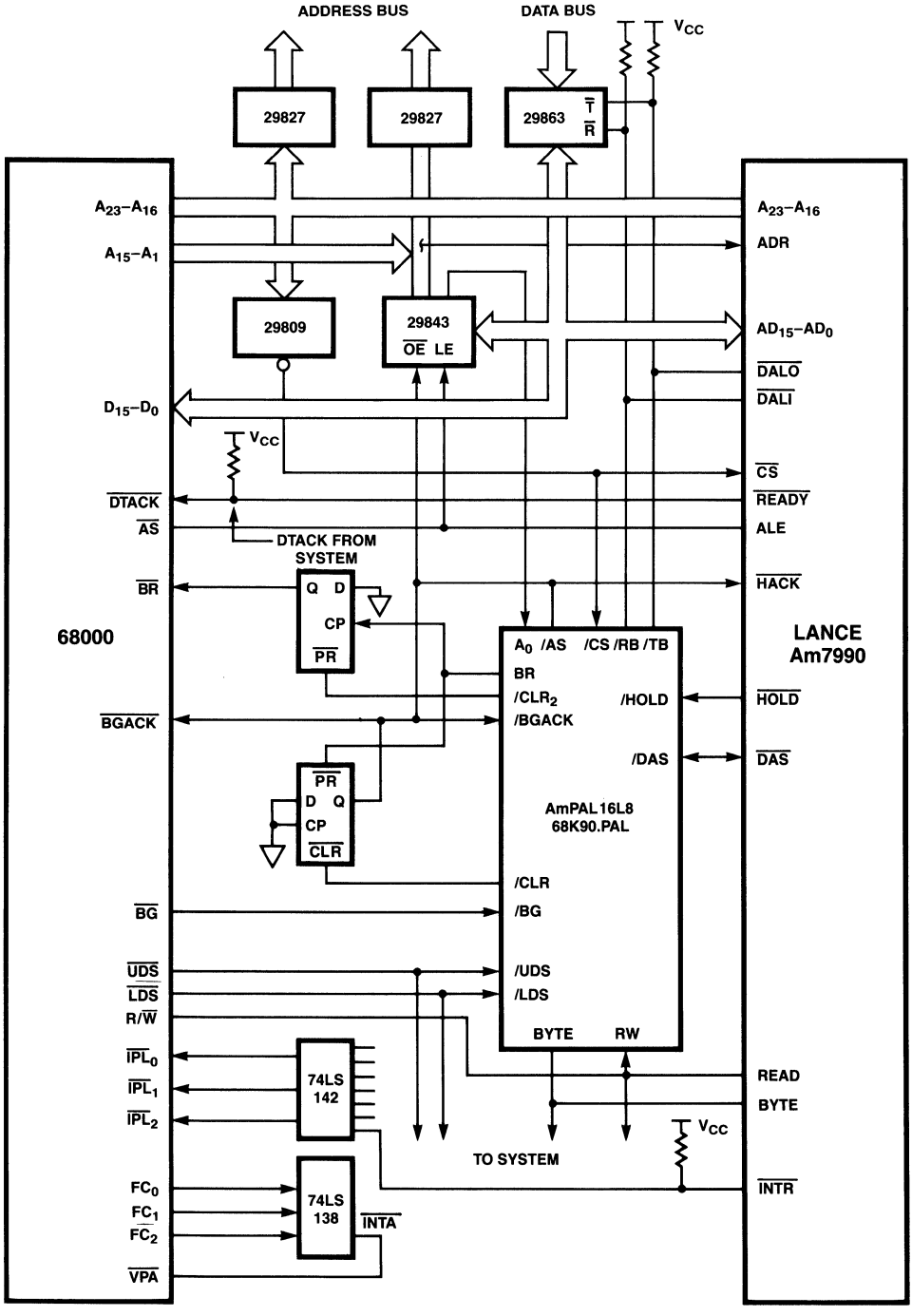
6.3.0 68000 AND AMD PROPRIETARY PERIPHERALS INTERFACE

AMD manufactures a large number of microprocessor peripherals, and they can be interfaced with a number of CPU types, the user is advised to verify that the interface specification is met. Two of the important parameters are set-up and hold times to insure that peripherals will work with both fast and slow CPU's. In some cases the insertion of a Wait State is all that is required. In the following sections, the interface between a number of the AMD proprietary peripheral products with the 68000 are discussed.

The Am29809 Comparator and the Am29806 Comparator/Decoder provide high-speed address selection as well as an open collector acknowledge driver. This allows memories and peripherals to be conveniently wire ORed to the processor's $\overline{\text{DTACK}}$ pin.

6.3.1 68000 AND Am7990 LANCE INTERFACE

The design of the LANCE has made it easier for the user to interface the device with demultiplexed buses. The example shown here is an interface to be compatible with an 8 MHz or faster 68000 (Figure 6-3.1a). The two flip-flops are needed to adapt the LANCE bus request handshake to the 68000.



02188A-70

Figure 6-3.1a

Autovectoring is used since the Am7990 does not return a vector during interrupt acknowledge cycles. The BYTE and $\overline{DAS}$ signals of LANCE are used to generate the $\overline{UDS}$ and $\overline{LDS}$ when LANCE is in Bus Master mode; the $\overline{UDS}$ and $\overline{LDS}$ is used to generate the $\overline{DAS}$ when LANCE is in Bus Slave mode. It takes two latches to demultiplex the LANCE address/data lines to adapt to the 68000 address bus. The flip-flops can be replaced by an Am22V10 to minimize parts count. Equations for the AmPAL16L8 are shown in Figure 6-3.1b.

6.3.2. 68000 AND Am8052 CRT CONTROLLER INTERFACE

One of the designer's most challenging tasks is to interface two generically different Bus Masters. An example would be the 68000 microprocessor and the Am8052 CRT Controller. The Am8052 CRT Controller is a general purpose interface device for raster scan CRT displays. The CRTC provides efficient manipulation of complex character formats and screen structures to allow sophisticated text display without undue CPU overhead. The diagram for the 68000 and Am8052 interface is shown in Figure 6-3.2a. Both Bus Masters support a 16-bit-wide data bus and a 24-bit linear addressing space (if the Linear/Segmented bit in the Am8052 Mode Register 1 is set to "1"). The control bus signals of the Am8052, however, differ from that of the 68000's and need to be translated bidirectionally. An easier approach (not shown here) is to convert the 68000 signals to Am8052 signals. Since the 68000 is never a slave, the logic is simple. This approach is feasible only if most are not 68000 peripherals.

The Am8052 provides two basic slave modes: the latched mode for systems with multiplexed address/data buses and the unlatched mode for systems with demultiplexed address/data buses. In this interface application, the Am8052 operates in the unlatched mode because the address and data buses of the 68000 are demultiplexed. In the unlatched slave mode, Address Strobe ($\overline{AS}$) is kept asserted throughout the entire bus cycle, making the internal latches for Chip Select ($\overline{CS}$) and Control/$\overline{Data}$ (C/$\overline{D}$) transparent. $\overline{AS}$ is driven Low by an open collector inverter connected to $\overline{BAI}$. This forces $\overline{AS}$ to go Low whenever the Am8052 is not in control of the bus.

```
PAL16L8                         JOE BRCICH
PAT002                          2 FEB 84
68000 TO LANCE INTERFACE
ADVANCED MICRO DEVICES

/AS   RW   BYTE   /HOLD   NC   /BG   A0   NC   /BGACK   GND
/CS   /TB   /UDS   /DAS   /CLR1   BR   /CLR2   /LDS   /RB   VCC

IF (/BGACK) RB = CS*RW*UDS + CS*RW*LDS

IF (/BGACK) TB = CS*/RW

IF (BGACK) UDS = DAS*/A0*BYTE + /BYTE*DAS

IF (BGACK) LDS = DAS*A0*BYTE + /BYTE*DAS

IF (/BGACK) DAS = UDS*LDS

CLR1 = /AS*BG

CLR2 = BGACK          ;DELAY

/BR = /HOLD

DESCRIPTION
        THE GOAL OF THIS INTERFACE WAS TO BE COMPATIBLE
WITH 8 MHZ AND FASTER 68000'S WHILE MINIMIZING PARTS COUNT.
THE AM22V10 COULD BE USED TO ELIMINATE THE TWO FLIP-FLOPS
SHOWN.  AUTOVECTORING IS USED SINCE THE 7990 DOES NOT RETURN
A VECTOR DURING INTERRUPT ACKNOWLEDGE CYCLES.
NOTE PROGRAM BSWP, BCON TO 1 ,AND ACON TO 0 IN CSR3 REG.
```

Figure 6-3.1b

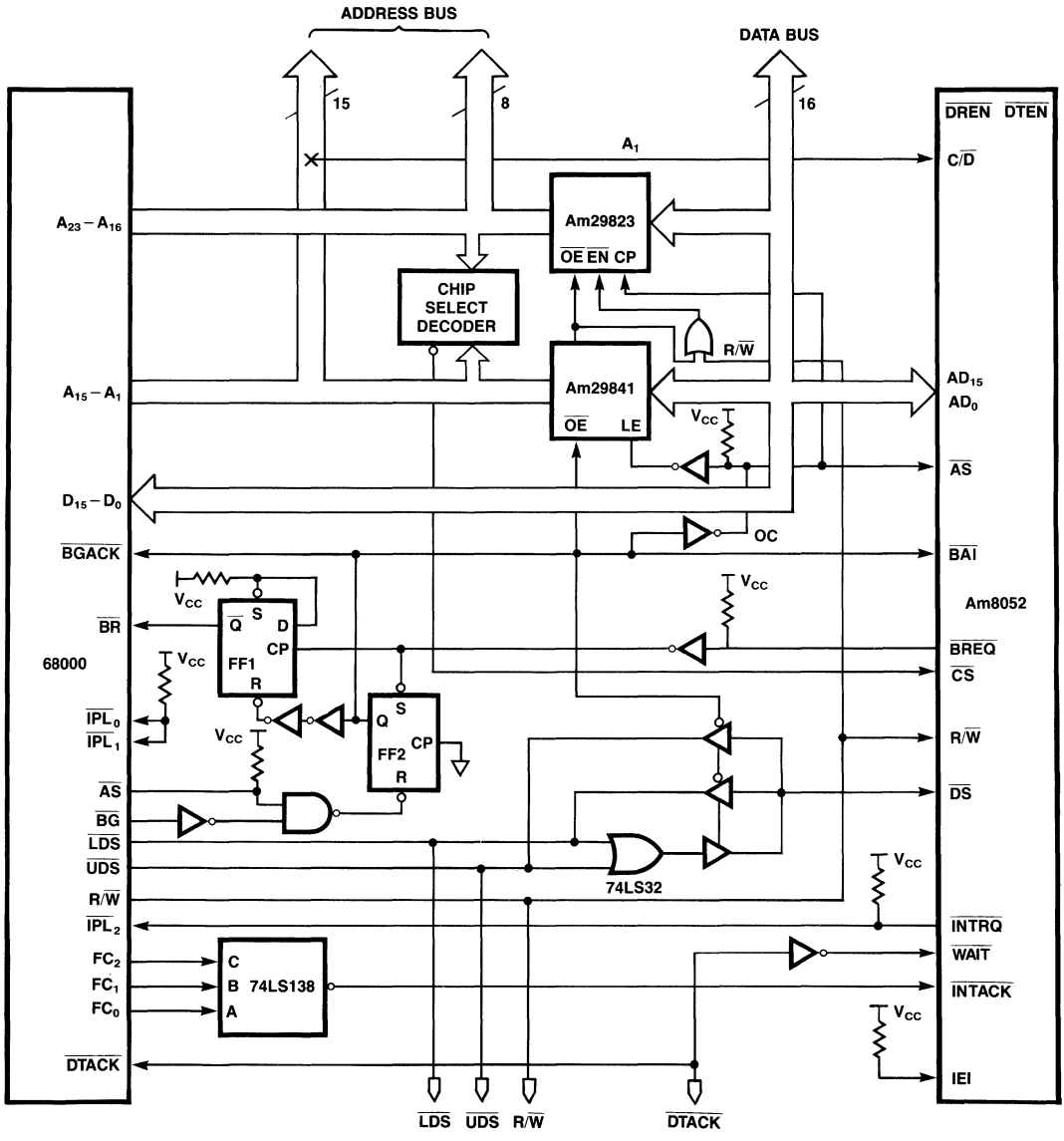


Figure 6-3.2a 68000–Am8052 Interface

The Am8052 in slave mode can only be accessed as a 16-bit peripheral (word transfers only). This means that both Data Strobes of the 68000 ($\overline{LDS}$ and $\overline{UDS}$) must be active simultaneously. It is only then that the OR gate asserts $\overline{DS}$ for the Am8052. The driver is enabled when the 68000 is Bus Master ($\overline{BAI}$ High). In Master Mode, both data strobes are driven by the Am8052 since the Am8052 does only word transfers.

After the Am8052 is initialized and the display is enabled, the Am8052 asserts Bus Request ($\overline{BREQ}$) to request the system bus. To avoid bus contention at the end of Bus Master read cycle, the data bus transceiver (not shown) must be turned off before the Am8052 starts driving the address for the next cycle. Timing Parameter 11 allows a turn-off time of 25 ns which is sufficient for the Am29863 transceiver.

The 68000 CPU supports a three-wire bus arbitration mechanism. A peripheral requesting bus mastership asserts a Bus Request ($\overline{BR}$), see Figure 6-3.2b. The CPU, in response, asserts a Bus Grant ($\overline{BG}$). At the end of the current bus cycle, the requesting peripheral goes on the bus. The end of the current CPU bus cycle is signaled by the Address Strobe going inactive. The combination of Bus Grant active and Address Strobe inactive asynchronously resets FF2 (see Figure 6-3.2a), thereby asserting $\overline{BAI}$ for the Am8052 and Bus Grant Acknowledge ($\overline{BGACK}$). Resetting FF2 also resets FF1 asynchronously, which deactivates $\overline{BR}$. In response to $\overline{BR}$ becoming inactive, the 68000 deactivates $\overline{BG}$. Note that $\overline{BR}$ must be Low for at least 20 ns after $\overline{BGACK}$ to prevent rearbitration. The inverters and the delay through FF1 must meet this requirements. $\overline{BGACK}$ and $\overline{BAI}$ stay asserted until the Am8052 terminates its DMA burst and releases $\overline{BREQ}$. At that time FF2 is asynchronously set and $\overline{BGACK}$ and $\overline{BAI}$ are deactivated, and the 68000 resumes operation.

The bus arbitration mechanism does not yet support DMA preemption. However, Am8052 DMA preemption by external devices can simply be supported by setting FF2 when preemption is desired. The preempting DMA can get the bus after the Am8052 has released the bus. This is indicated when the Am8052 deactivates $\overline{BREQ}$. In this case, $\overline{BAI}$ being Low is no longer sufficient to flag that the Am8052 has been granted the system bus. For proper DMA preemption support, the data strobe drivers and the open collector driver for $\overline{AS}$ must be controlled by a signal which flags that the Am8052 is on the bus. (Note: For the time between preemption ($\overline{BAI}$ High) and bus release ($\overline{BREQ}$ High), the Am8052 is still in control of the system bus. The Am8052 supports vectored interrupts if the No Vector bit in Mode Register 2 is disabled (NV=0). The vector is put out in Interrupt Acknowledge cycles ($\overline{INTACK}$ Low, IEI High, and $\overline{DS}$ Low).

All transactions to the Am8052 must be in words. All transactions between the Am8052 and memory are word operations; thus, the Am8052 does not have a B/$\overline{W}$ line and will always produce both an $\overline{LDS}$ and $\overline{UDS}$. Since the design goal of the Am8052 restricted the package to 68 pins, an external latch is required to get a full 24 bits of address. This latch is updated whenever a 64K boundary is crossed by a special write cycle.

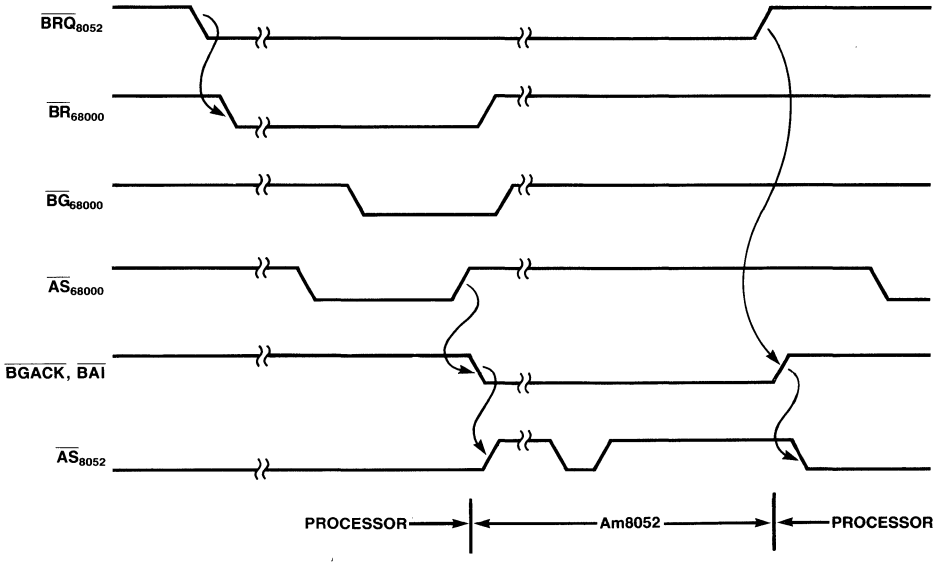


Figure 6-3.2b Bus Exchange Timing

6.3.3 68000 TO AmZ8068 DATA CIPHERING PROCESSOR INTERFACE

Figures 6-3.3a and 6-3.3b show the 68000-DCP interface and the interface timing. This interface provides a two-chip solution to add high speed data ciphering to a 68000 based system. About 500 kbyte/sec are possible in a CPU controlled transfer. The ciphering rate can be increased with a sophisticated DMA controller, or with several DCPs operating in parallel. The CPU operates at 8 MHz and the DCP operates synchronously at 4 MHz. The Interface Controller, a PAL device, generates the Address and Data Strobes for the DCP and the Data Acknowledge for the CPU. It also divides the CPU clock by two and synchronizes it to the Data Strobes.

The main features of this interface are:

— Multiplexed Control Mode

— Demultiplexed address and data bus

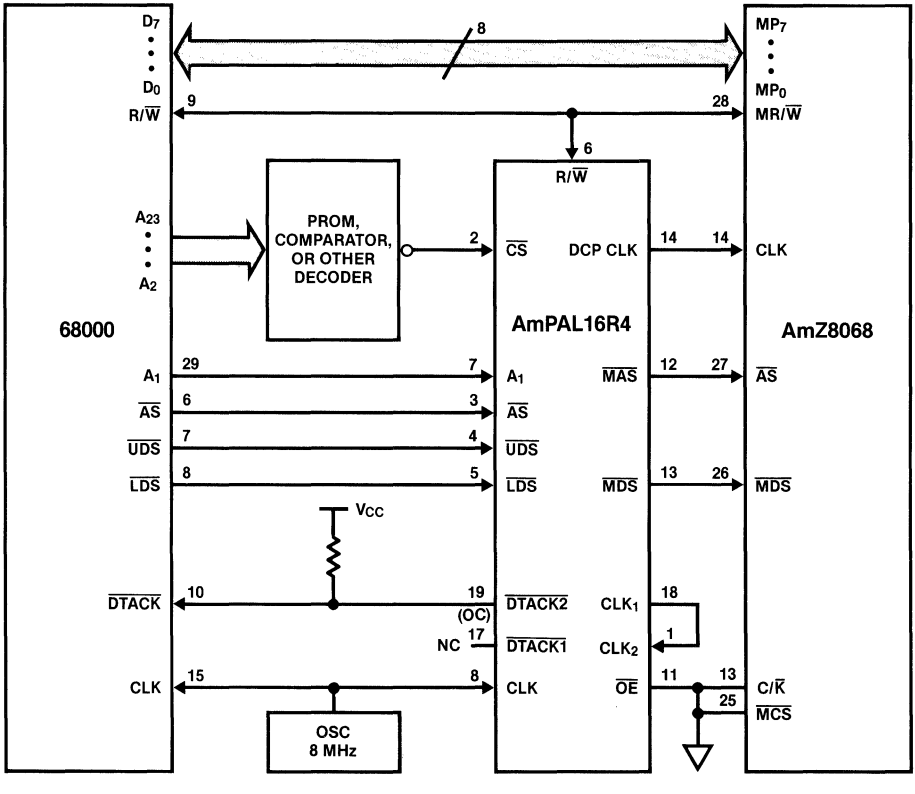
— Two-Cycle Operation

— Clock Synchronization with two Low Cycles after the Data Strobes

— About 500 kbyte/sec ciphering speed

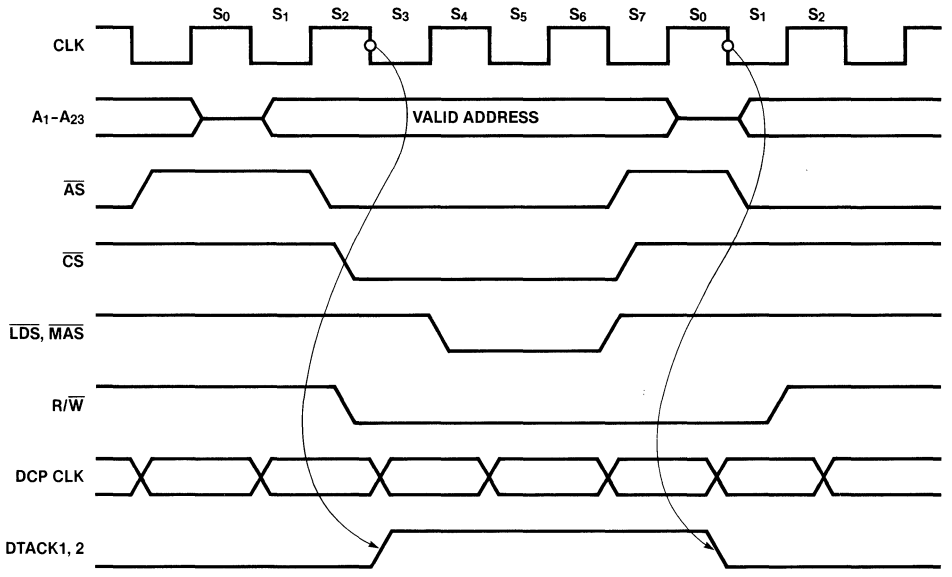
Data transfers between the CPU and the DCP are accomplished by a two-cycle operation. First the address of an internal register is latched in, then the data is transferred. This causes a small overhead in the initialization phase, but improves the ciphering rate in a high-speed data ciphering session. The rate of 500 kbyte/sec can be reached only if a high-speed peripheral device is connected to the Slave Port and the DCP is programmed for dual port configuration.

The PAL device is programmed to allow only DCP transfers to the DCP. The PAL device equations are shown in Figure 6-3.3e. A_0 must be odd to make the CPU transfer the data on the Low byte of the data bus. A "0" on A_1 indicates an Address Latch Cycle, whereas a "1" on A_1 indicates a Data Transfer Cycle. A_0 must be "1" in both cycles.



02188A-73

Figure 6-3.3a AmZ8068 to 68000 Connection Using a PAL



02188A-74

Figure 6-3.3b 68000–AmZ8068 Address Latch Cycle (A_1 = Low)

An address decoder generates the Chip Select for the DCP. The Address Strobe indicates a valid address. The PAL device is only activated if the Lower Data Strobe becomes active while the Upper Data Strobe stays inactive. This means that data is transferred in MOVE.B instructions with an odd peripheral address.

The PAL device provides two Data Acknowledge outputs. $\overline{DTACK}_1$ is an active Low TTL output. $\overline{DTACK}_2$ has the same timing as $\overline{DTACK}_1$, but is an Open Collector output. (The Open Collector output is realized by a three-state output which has only two states, Low or Floating.)

ADDRESS LATCH CYCLE

In this cycle only a Master Port Address Strobe ($\overline{MAS}$) is generated. Master Port Chip Select ($\overline{MCS}$) is tied to Low. $\overline{LDS}$ is sent to the $\overline{MAS}$ output. The minimum pulse width of $\overline{LDS}$ is 115 ns; 80 ns are required for the AmZ8068.

$\overline{DTACK}$ is activated with the falling edge of the CPU clock after cycle S_2. The CPU inserts no Wait states. $\overline{DTACK}$ is deactivated with the first edge of CLK after $\overline{AS}$ becomes inactive.

DATA READ CYCLE: (FIGURE 6-3.3c)

The generation of $\overline{MDS}$ in a Data Read Cycle is similar to the Data Write Cycle. Because the CPU activates $\overline{LDS}$ one cycle earlier, there is no need for a Wait State. The minimum pulse width of $\overline{LDS}$ is 240 ns; the DCP requires 200 ns for a Status Register read. $\overline{DTACK}$ is activated using the same logical condition as in the Data Write cycle. Because of the earlier activation of $\overline{LDS}$, $\overline{DTACK}$ becomes active earlier and the CPU inserts no Wait states.

DATA WRITE CYCLE: (FIGURE 6-3.3d)

A Data Write Cycle is performed when A_0 is High, $\overline{AS}$, $\overline{CS}$ and $\overline{LDS}$ are Low. The minimum pulse width of $\overline{LDS}$ is not sufficient for the DCP which requires at least 125 ns. One Wait state or a slower system clock will satisfy this parameter. In this interface, one Wait state is inserted by activating $\overline{DTACK}$ at the end of S_4.

The DCP clock is synchronized in Data Read or Write Cycles by forcing it Low when DTACK becomes active. This guarantees that the DCP clock has a falling edge just before $\overline{LDS}$ ($\overline{MDS}$) rises. The delay of the DCP clock to CLK is typically 8 ns for a normal speed PAL device. The delay of $\overline{LDS}$ to $\overline{MDS}$ is typically 12 ns. The delay of $\overline{LDS}$ to the system clock is 0-70 ns for the 8 MHz version. This results in a delay of 4-74 ns of $\overline{MDS}$ to the DCP clock. The DCP requires 0-50 ns when operating at the maximum clock rate.

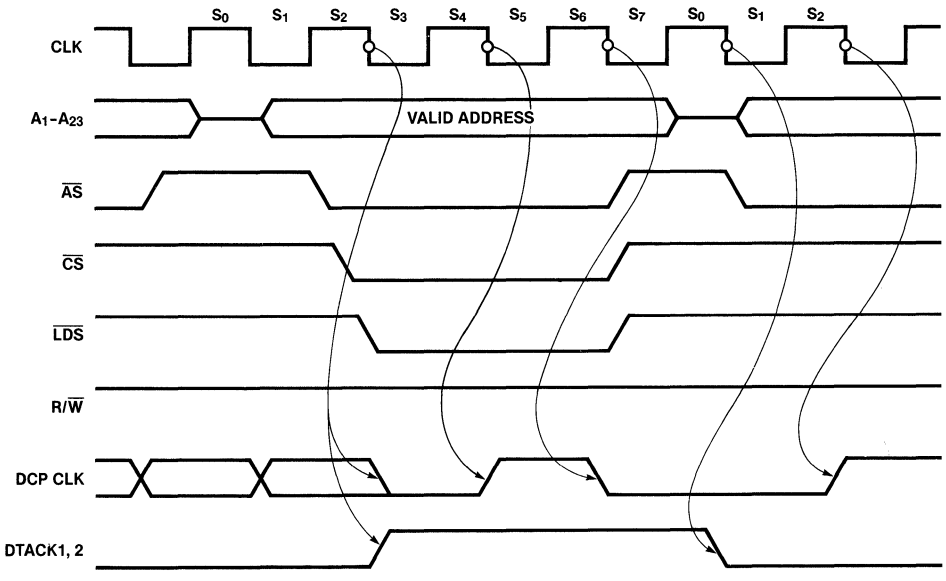
This problem is solved by stretching the clock for one cycle. The DCP clock stays Low for two cycle in the end of a transfer cycle. This is done automatically by the PAL device (see Timing Diagram).

A SAMPLE DATA CIPHERING SESSION

The interface was built on a Motorola 68000 evaluation board. The instructions are shown below. These instructions were executed with the "MACSBUG 1.32" evaluation board monitor.

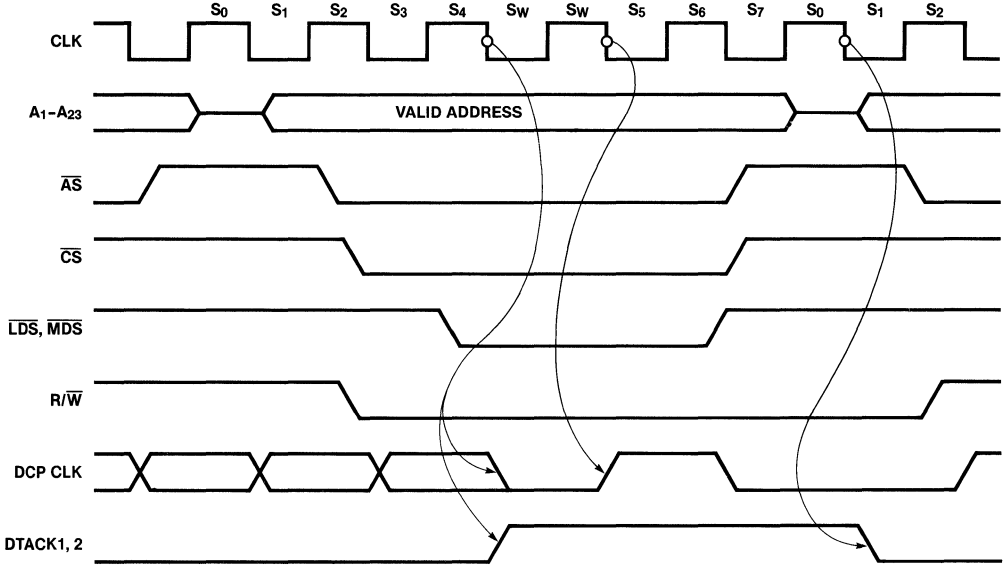
Address Register:

A_0 ..... Address Latch Address (A_1=0,A_0=1)
A_1 ..... Data Transfer Address (A_1=1,A_0=1)



02188A-75

Figure 6-3.3c 68000–AmZ8068 Data Read Cycle (A_1 = High)



02188A-76

Figure 6-3.3d 68000–AmZ8068 Data Write Cycle (A_1 = High)

```
PAL16R4                                          PAL DESIGN SPECIFICATION
DCP044                                           JUERGEN STELBRINK  8-24-83
68000 - AmZ8068 (DCP) INTERFACE DEVICE
ADVANCED MICRO DEVICES

CLK2  /CS   /AS   /UDS    /LDS  RW   A1        CLK   NC        GND
/OE   /MAS  /MDS  DCPCLK  NC    NC   /DTACK1   CLK1  /DTACK2   VCC

/CLK1   =  CLK           ; INVERT CLOCK TO TRIGGER THE REGISTERED
                         ; OUTPUTS WITH THE FALLING EDGE OF CLK

MAS     =  AS*LDS*/UDS*/RW*/A1*CS

MDS     =  AS*LDS*/UDS*A1*CS

/DCPCLK := DCPCLK +                    ; DIVIDE BY TWO
           /DTACK1*CS*AS*LDS*/UDS +
           DTACK1*/AS*/LDS*/UDS        ; TWO CLOCKS LOW IN
                                       ; THE END OF A DATA CYCLE

DTACK1  := AS*LDS*/UDS*A1*CS +         ; DATA TRANSFER CYCLE
           AS*/RW*/A1*CS               ; ADDRESS LATCH CYCLE

IF (DTACK1*AS*CS) DTACK2 = DTACK1


FUNCTION TABLE

CLK2 CLK CLK1 /CS /AS /LDS /UDS RW A1
DCPCLK /MAS /MDS /DTACK1 /DTACK2

;                                          /  /
;                                     D    D  D
;                                     C    T  T
; C     C        /  /          D  /  /  A  A
; L  C  L  /  /  L  U          C  M  M  C  C
; K  L  K  C  A  D  D  R  A    L  A  D  K  K
; 2  K  1  S  S  S  S  W  1    K  S  S  1  2     COMMENT
-------------------------------------------------------------------
; CLOCK INVERT
  X  L  H  X  X  X  X  X  X    X  X  X  X  X
  X  H  L  X  X  X  X  X  X    X  X  X  X  X
; DATA WRITE CYCLE
  C  X  X  L  H  H  H  H  H    X  H  H  H  Z  ; S0
  C  X  X  L  L  H  H  L  H    X  H  H  H  Z  ; S2
  C  X  X  L  L  L  H  L  H    L  H  L  L  L  ; S4
  C  X  X  L  L  L  H  L  H    H  H  L  L  L  ; SW (1 WAIT STATE)
  C  X  X  L  L  L  H  L  H    L  H  L  L  L  ; S6
  X  X  X  L  H  H  H  L  H    L  H  H  L  Z  ; S7
  C  X  X  H  H  H  H  L  X    L  H  H  H  Z  ; S0
  C  X  X  H  L  X  H  X  X    H  H  H  H  Z  ; S2
; DATA READ CYCLE
  C  X  X  H  H  H  H  H  H    X  H  H  H  Z  ; S0
  C  X  X  L  L  L  H  H  H    L  H  L  L  L  ; S2
```

Figure 6-3.3e

```
  C  X  X  L  L  L  H  H  H     H  H  L  L  L ; S4
  C  X  X  L  L  L  H  H  H     L  H  L  L  L ; S6
  X  X  X  L  H  H  H  H  H     L  H  H  L  Z ; S7
  C  X  X  L  H  H  H  H  H     L  H  H  H  Z ; S0
  C  X  X  X  H  H  H  H  H     H  H  H  H  Z ; S2
; ADDRESS LATCH CYCLE
  C  X  X  L  L  H  H  L  L     X  H  H  L  L ; S2
  C  X  X  L  L  L  H  L  L     X  L  H  L  L ; S4
  C  X  X  L  L  L  H  L  L     X  L  H  L  L ; S6
  X  X  X  L  H  H  H  L  L     X  H  H  L  Z ; S7
  C  X  X  X  H  H  H  L  L     X  H  H  H  Z ; S0
;
```

DESCRIPTION:

INPUT SIGNALS:

CLK2 CLOCK FOR THE REGISTERED OUTPUTS OF THE PAL. IT IS CONNECTED TO CLK1

CLK 8 MHZ 68000 SYSTEM CLOCK

/CS CHIP SELECT FOR DCP (A2-A23 ARE RELEVANT)

/AS ADDRESS STROBE

/LDS LOWER DATA STROBE USED TO TIME THE MASTER PORT DATA STROBE

/UDS UPPER DATA STROBE HAS TO BE INACTIVE DURING ALL TRANSFERS

A1 ADDRESS BIT 1 DISTINGUISHES BETWEEN ADDRESS LATCH AND DATA TRANSFER CYCLES

A1=LOW ADDRESS LATCH
A1=HIGH DATA TRANSFER

RW READ/ WRITE CONTROL

OUTPUT SIGNALS:

/MAS MASTER PORT ADDRESS STROBE

/MDS MASTER PORT DATA STROBE

CLK1 INVERTED CLOCK CLK

/DTACK1 LOW ACTIVE DATA ACKNOWLEDGE FOR 68000
ONE WAIT STATE IS INSERTED IN A DATA WRITE CYCLE

/DTACK2 LOW ACTIVE DATA ACKNOWLEDGE FOR 68000 (OPEN COLLECTOR)

DCPCLK 4 MHZ DCP CLOCK, IT IS SYNCHRONIZED TO THE MASTER PORT DATA STROBE. IN A DATA TRANSFER CYCLE DCPCLK STAYS TWO CLK CYCLES LOW TO DELAY THE FIRST RISING EDGE OF THE

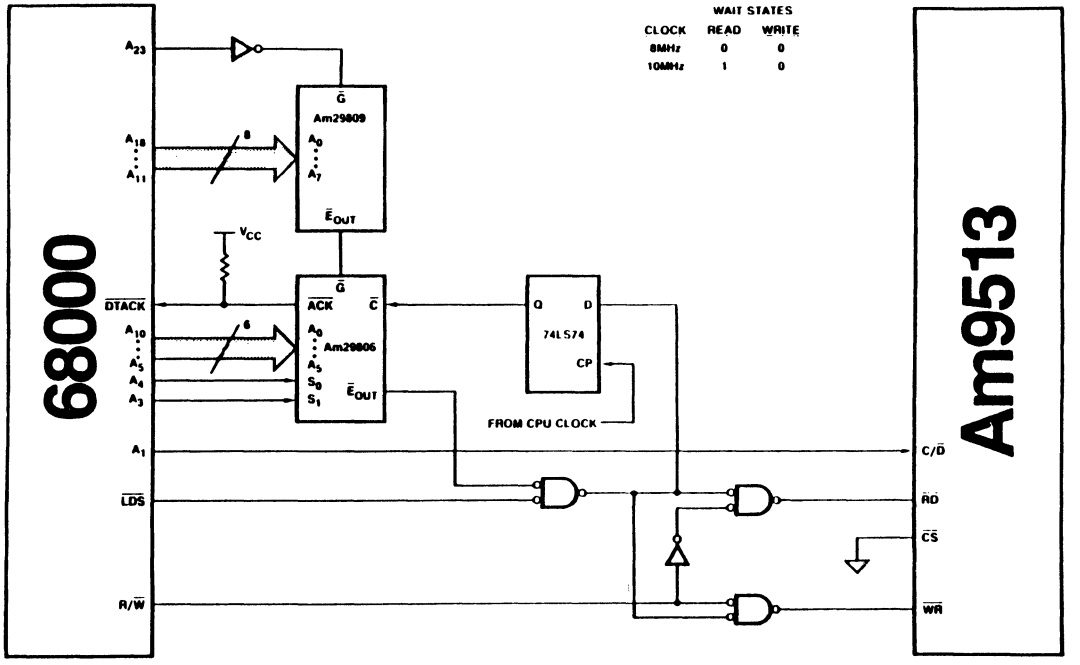
Figure 6-3.3e (continued)

6.3.4 68000 AND Am9513A SYSTEM TIMING CONTROLLER INTERFACE

The Am9513A System Timing Controller is an LSI circuit designed to service many types of counting, sequencing and timing applications. It provides the capability for programmable frequency synthesis, high resolution programmable duty cycle waveforms, retriggerable digital one-shots, time-of-day clocking, coincidence alarms, complex pulse generation, high resolution baud rate generation, frequency shift keying, stopwatching timing, event count accumulation, waveform analysis and many more. A variety of programmable operating modes and control features allow the Am9513A to be personalized for particular applications as well as dynamically reconfigured under program control.

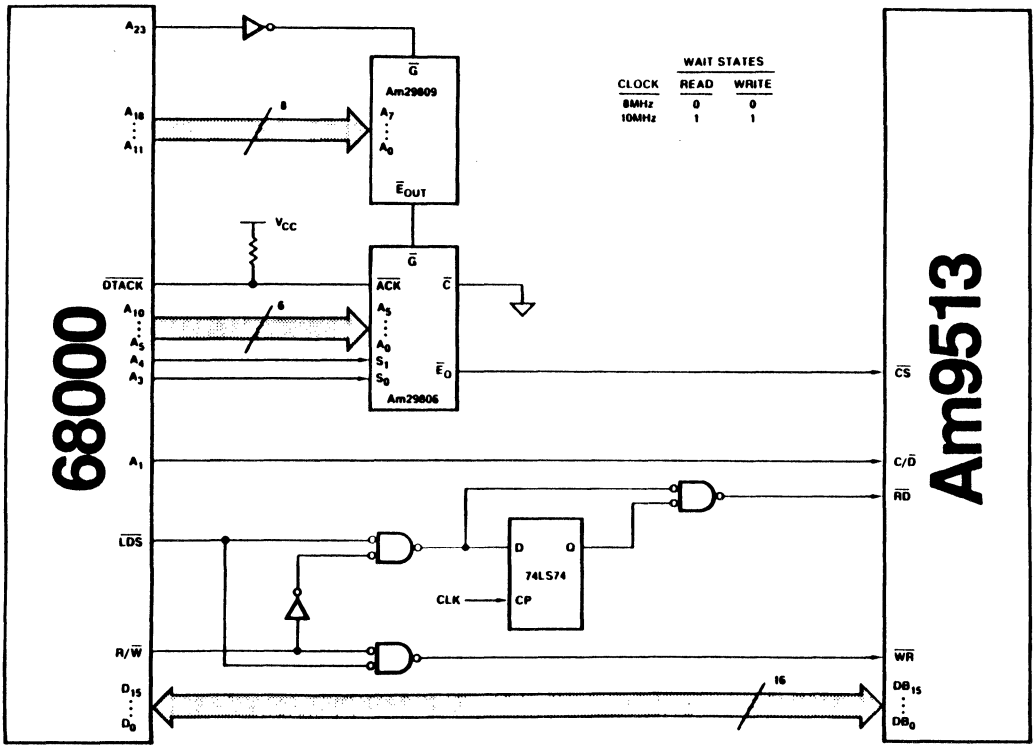
Figure 6-3.4a shows the block diagram of the 68000-Am9513A interface. This interface was contributed by Carlos Manamer from Wang. The trick here is that $\overline{CS}$ is tied low and $\overline{RD}$ and $\overline{WR}$ is gated through only when the proper address is selected. This design inserts one Wait State for write and none for read. This design assumes the CPU to be an 8 MHz device. It should also work with 10 MHz and 12 MHz CPU's as well. The key thing to observe, when interfacing the Am9513A to the 68000, is $\overline{CS}$ setup time with respect to the falling edge of $\overline{RD}$.

Without tying $\overline{CS}$ low, the falling edge of $\overline{RD}$ would have to be delayed to meet this requirement. This technique is shown in the second interface (Figure 6-3.4b). The flip-flop does the delay while the OR gate prevents the rising edge from being delayed, as this may cause bus contention with other devices. Note that this second technique requires more Wait States than the first.



02188A-77

Figure 6-3.4a Am9513 to 68000 Connection Using SSI/MSI



02188A-78

Figure 6-3.4b Am9513 to 68000 Connection Using SSI/MSI

6.3.5 68000 AND A SINGLE Am9516 DMA CONTROLLER INTERFACE

The Am9516 Universal DMA Controller (UDC) is a high performance peripheral interface circuit for 8086 and 68000 CPUs. The UDC was designed to interface with non-multiplexed address and data bus systems. However, because it is basically a modified 8016, several incompatabilities remain. ALE is more like the 8086 than the 68000. Although the timing of ALE closely matches the $\overline{AS}$ of the 68000, it does not tristate when the 9516 is not the bus master. The major obstacle in this design was generating the proper Data Hold time for a slave write; the 8 MHz and 10 MHz CPU provides 30 ns and 20 ns, respectively, while the 9516 requires 40 ns. The newer, 8 and 10 MHz versions only require 10 ns Hold time, thus simplifying the design.

The interface design shown here (Figure 6-3.5a) uses an AMPAL16R4 to insert a Wait State and truncate $\overline{DS}$ to the 9516 early. PAL device equations are shown in Figure 6-3.5b. In this case the shortened $\overline{DS}$ strobes data into the 9516 while the CPU holds data valid for almost 1 clock cycle afterwards. This Wait State is inserted only during slave write. This Wait State is actually a small penalty to pay, because most of the registers are being loaded by the UDC itself by means of a linked-list control structure. This results in very few writes being done directly to the UDC.

Another interface problem to be addressed is the bus exchange protocol. The BREQ and BACK handshake of the UDC is converted to the three-wire handshake of the 68000 by the PAL device and a 7403 package (open collector NAND Gate). This is done by a state machine internal to the PAL device. The state assignment was done to minimize external logic. This is very different from the discrete design in the Am9516 data sheet because the PAL doesn't have preset and clear on its flip-flops.

The UDC can support the bus error function by means of its $\overline{EOP}$ pin. If a bus error occurs during a DMA transfer, an $\overline{EOP}$ will cause the DMA operation to stop and interrupt the CPU. The CPU can read status, address, word count, or anything else it may need to take corrective action. The UDC may then be restarted or reprogrammed appropriately.

The control signals $\overline{LDS}$ and $\overline{UDS}$ are generated by the PAL device when the UDC is the bus master. This is a straightforward combination of $\overline{DS}$, A0, and B/$\overline{W}$.

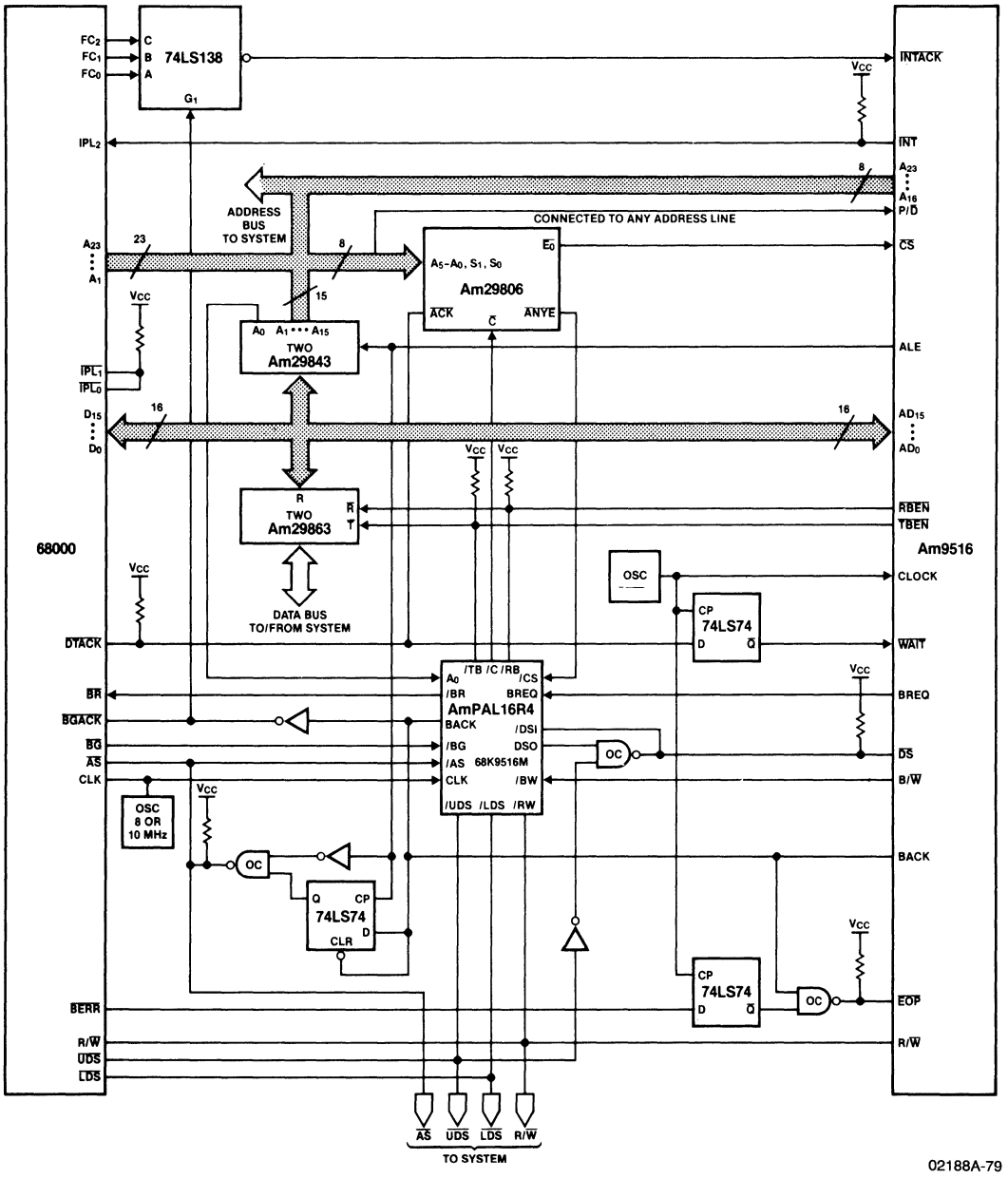


Figure 6-3.5a The Am9516 UDC to 68000 CPU Interface

Note that the address latch and data transceiver, along with the address decode logic, are present in any case. The 68450 and 68440 also multiplex address and data, thus the PAL device is the only extra component required to make the 9516 a low cost alternative. Additionally, the 6 MHz 9516 has the same data throughput as the 8 MHz 68450.

For those wary of or unfamiliar with PAL devices, these interfaces can be built with standard gates and flip-flops. The PAL device is simply a convenient means to collapse many SSI packages into one and conserve board space.

```
PAL16R4                  PAL DESIGN SPECIFICATION
PAT006                     JOE BRCICH 9/01/83
68000 TO Am9516 INTERFACE WITH DATA HOLD CORRECTION
ADVANCED MICRO DEVICES

CLK  RW   A0  BREQ  /BG  /DSI  /AS  /BW  /CS  GND
/OE /LDS /UDS DSO   /C   BACK  /BR  /TB  /RB  VCC

IF(/BACK) RB = /CS * RW * UDS     +
               /CS * RW * LDS
IF(/BACK) TB = /CS * /RW

IF(BACK) UDS = DSI * /A0 * /BW    +
               BW * DSI
IF(BACK) LDS = DSI * A0 * /BW     +
               BW * DSI

BR := BREQ * BG * BR * AS         +
      BREQ * /BG * /BACK

/BACK := /BREQ                    +
         /BREQ * /BG              +
          /BREQ * AS              +
         /BREQ * /BACK            +
         /BG * /BACK              +
         AS * /BACK

C := UDS * /BACK                  +
     LDS * /BACK

/DSO := BACK                      +
        /BACK * /RW * C
```

DESCRIPTION

IF BREQ*BACK IS TRUE THE Am9516 HAS THE BUS, OTHERWISE THE 68000 HAS THE BUS. THIS PAL CONNECTS THE Am9516 TO THE 68000 WITH ONE WAIT STATE DURING WRITES WHILE SHORTENING /DS TO ACHIEVE PROPER DATA HOLD TIME. IT ALSO CONVERTS THE BUS EXCHANGE PROTOCOL INTO 68000 FORMAT. THIS DESIGN ASSUMES NO OTHER BUS MASTERS IN THE SYSTEM. /RB AND /TB CONTROL THE TRANSCEIVERS WHEN CPU IS BUS MASTER. /CS MUST BE A FUNCTION OF ALL DEVICES CONNECTED TO THE CPU BUS NOT JUST THE Am9516 /CS AS SHOWN HERE.

The /CS to /DS set-up time of 30 ns is met in the following ways:

1) During a read cycle the only effect from not meeting this set-up time is that the data valid access time from the Am9516 will be delayed by a proportional amount. Since the minimum /DS Low width from the 10-MHz 68000 (during a read) is 193 ns and the minimum /DS Low width to the Am9516 is 150 ns, we have 43 ns margin not counting gate delays which will further increase this margin.

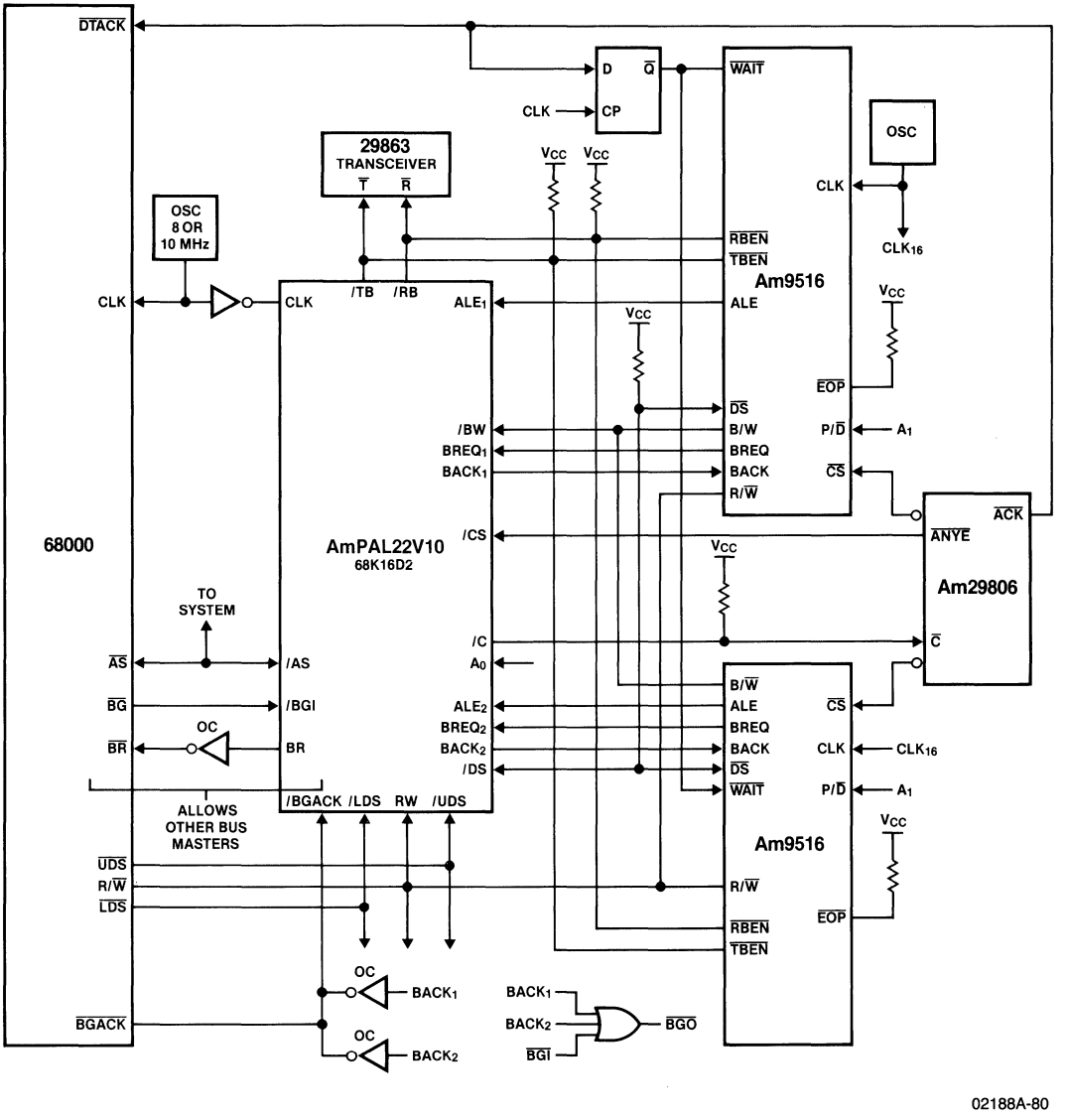
2) During a write cycle this is not an issue since the /DS comes later and is stretched longer due to the Wait state.

Figure 6-3.5b

6.3.6 68000 AND DUAL Am9516 DMA CONTROLLERS INTERFACE

There has been interest shown in connecting two Am9516 DMA Controllers to obtain four channels. The example here shows that such a system can be built by incorporating one PAL device, AMD's new 22V10 (Figure 6-3.6a). Address and data buses are not shown as they are straightforward and require no explanation. The PAL device, designated 68K16D2, converts the two DREQs into the 68000 three-wire handshake, prioritizes the request, and converts the control signals appropriately. Equations for the PAL device are shown in Figure 6-3.6b.

The key parameters are: 1) data hold with respect to the rising edge of $\overline{DS}$ during a write, and 2) $\overline{DTACK}$ set-up time. Control for a data bus transceiver is shown because it will be required in most systems. The PAL device provides these signals when the CPU is bus master; the Am9516 generates these control signals directly when it is bus master.



02188A-80

Figure 6-3.6a Dual Am9516 UDCs to 68000 CPU Interface

```
PAL 22V10                              JOE BRCICH
PAT 001                                5 APRIL 83
68000 TO DUAL 9516 INTERFACE
ADVANCED MICRO DEVICES

CLK RW  A0  BREQ1  BREQ2  /BG  NC  ALE1  /BW  ALE2  /BGACK  GND
/CS  /LDS  /UDS  /DS  /C  /AS  /BR  BACK2  BACK1  /TB  /RB  VCC

              BR = BREQ1*/BGACK + BREQ1*BG +
                   BREQ2*/BGACK + BREQ2*BG

           BACK1 = BREQ1*BG*/AS + /BG*BGACK

           BACK2 = BREQ2*/BREQ1*BG*/AS + /BG*BGACK

    IF (/BGACK) RB = CS*RW*UDS + CS*RW*LDS

    IF (/BGASK) TB = CS*/RW

IF (/BGACK*AS) DS := AS*/C*/RW + AS*RW

     IF (BGACK) AS = ALE1 + ALE2

    IF (BGACK) UDS = DS*/A0*/BW + BW*DS

    IF (BGACK) LDS = DS*A0*/BW + BW*DS

        IF (AS) C := UDS*BGACK + LDS*BGACK

DESCRIPTION
       THE GOAL IN THIS DESIGN WAS TO BE COMPATIBLE WITH
8 MHZ AND FASTER 68000'S.  BECAUSE THE EQUATIONS FOR DS AND
C WOULD EXTEND INTO THE NEXT CYCLE, THE OUTPUTS WERE TRI-
STATED TO TERMINATE THESE SIGNALS EARLIER. THIS REQUIRES THAT
PULL UP RESISTORS BE PLACED ON THESE OUTPUTS. THE INVERTED
CLOCK WAS USED TO PROVIDE MORE SETUP TIME FOR DTACK IN THESE
HIGH SPEED SYSTEMS. AT 8MHZ THE EQUATIONS CAN BE MODIFIED AND
THE PULL UP RESISTORS ELIMINATED. THIS EXAMPLE IMPLEMENTS
FIXED PRIORITY WHERE BREQ1 IS HIGHEST. NOTE THAT /AS SHOULD
NOT BE INCLUDED IN THE ADDRESS DECODER.
```

Figure 6-3.6b

The Am9516s are shown with independent clocks. The clocks may be divided from the CPU clock or may be generated independently of the CPU. Because $\overline{\text{WAIT}}$ must meet the set-up and hold times, $\overline{\text{DTACK}}$ may need synchronization by the use of a flip-flop, as shown. This flip-flop may be eliminated in synchronous systems.

The clock is inverted to the PAL device to meet $\overline{\text{DTACK}}$ set-up time in systems of 10 MHz or faster. This inverter may be deleted in slower systems, if appropriate changes to the PAL device equations are made. This PAL device implements fixed priority between the Am9516s. BREQ_1 is the highest priority. Rotating priority can be implemented by adding another PAL device.

$\overline{\text{EOP}}$s are pulled up separately, but could be tied together since they affect a channel only if it is active. The bus error function can be supported by connecting $\overline{\text{BERR}}$ and $\overline{\text{EOP}}$

If a bus error occurs, $\overline{\text{EOP}}$ will stop the current transfer and interrupt the CPU. The Interrupt Service routine can read the status to determine if $\overline{\text{EOP}}$ caused the interrupt or if termination was normal. If $\overline{\text{EOP}}$ caused the interrupt, the Address Register can be read to determine where the bus error occurred. After the problem is corrected, the CPU can program the Am9516 to complete the transfer or do an alternate transfer, as appropriate.

When operating the DMA in interleave mode, an external $\overline{\text{EOP}}$ should be gated with $\overline{\text{DACK}}$ to prevent affecting the wrong channel. This is unnecessary if interleave is not used, since the UDC releases the bus.

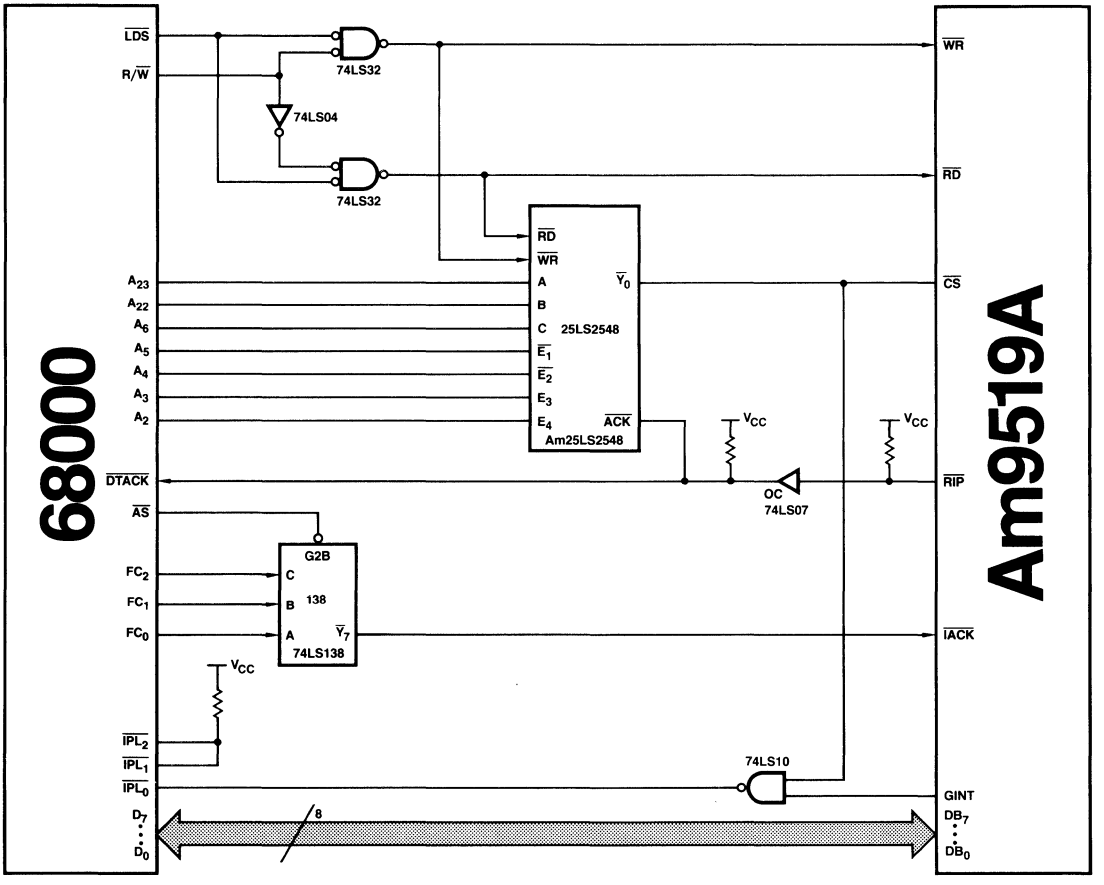
This arbiter design supports both serial and parallel expansion techniques and is therefore compatible with VME bus protocol. Bus grant out was implemented with an external gate due to a shortage of pins. The VME/BCLR function was not implemented because the Am9516 does not support preemption.

6.3.7 68000 AND Am9519A INTERRUPT CONTROLLER INTERFACE

Figure 6-3.7 shows the 68000 to Am9519A interface. This interface for the Am9519A will work for the 4 MHz and 6 MHz 68000s. It will also work with faster CPUs, if Wait States are inserted during programming. $\overline{RIP}$ will automatically insert the appropriate number of Wait States during the Interrupt Acknowledge cycle. Further decoding the upper Address bits will reduce the address space occupied by this part. Appropriate decoding of the address bits should generate an I/O space separated from the memory address space. This will prevent spurious glitches on Chip Select and meet the Am9519A's requirement that Chip Select be High for at least 100 ns prior to an Interrupt Acknowledge cycle. If there are no other interrupting devices in the system, $\overline{GINT}$ can be connected directly to one of the $\overline{IPL}$ pins, as shown.

$\overline{AS}$ is used to de-glitch the 74LS138's output, giving a clean $\overline{INTA}$. Note that the Am9519A must be connected to the lower Data Bus and $\overline{AS}$ must not be used to de-glitch $\overline{CS}$.

One last thing to consider is the recovery time. Do not use the MOVEP instruction to program the Am9519A.



02188A-81

Figure 6-3.7 68000 to Am9519A Interface

SECTION C
8-BIT PROCESSORS

7.0 INTERFACING TO THE Z80

7.1.0 OVERVIEW OF THE Z80

The Z80 is an 8-bit microprocessor which is software-compatible with the 8080. In addition, it has some additional instructions and registers which improve its processing capability. The dual-register set of the Z80 CPU allows high-speed context switching and efficient interrupt processing. In particular, the Z-80 provides a refresh signal to interface with dynamic memory devices. The four traditional functions of a microcomputer system (parallel I/O, serial I/O, counting/timing, and direct memory access) are easily implemented by the Z80 CPU.

The Z80 and the associated family of peripheral controllers are linked by a vectored interrupt system. This system may be daisy-chained to allow implementation of a priority interrupt scheme. Little, if any, additional logic is required for daisy-chaining. Two other modes of interrupt are programmable; one to be used with non-8080/Z80 systems, and the other compatible with the 8080 system. The CPU services interrupts by sampling $\overline{NMI}$ and $\overline{INT}$ signals at the rising edge of the last clock of an instruction. Further interrupt service processing depends upon the type of interrupt that was detected.

7.1.1 Z80 TO Am9511A ARITHMETIC PROCESSOR INTERFACE

Figure 7-1.1 illustrates a programmed I/O interface technique for Am9511A with a Z80 CPU.

The Chip Select ($\overline{CS}$) signal is a decode of Z80 address lines A1-A7. This assigns the Am9511A to two consecutive addresses, an even (Data) address, and the next higher odd (Command) address. Selection between the Data (even) and the Command/Status (odd) ports is by the least significant address bit A0.

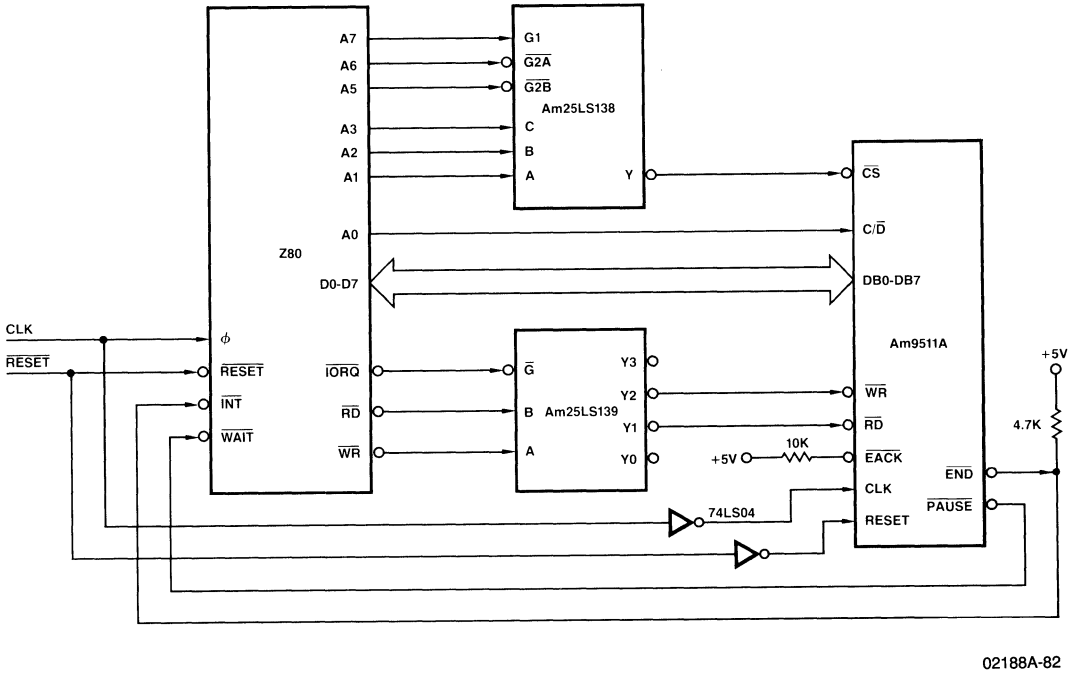


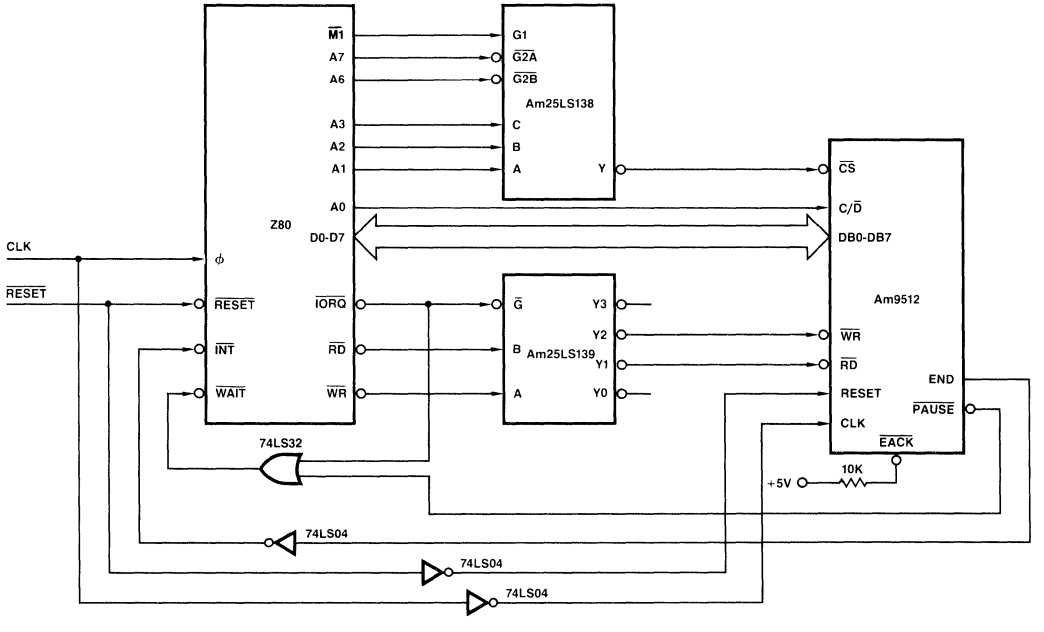
Figure 7-1.1 Z80 to Am9511A Interface

The $\overline{IORQ}$ (Input/Output Request) from the Z80 is an enable input to the 74LS139 decoder. The $\overline{WR}$ and $\overline{RD}$ from the Z80 are the two inputs to the decoder. The outputs Y1 and Y2 are tied to $\overline{WR}$ and $\overline{RD}$ of the the Am9511A. The $\overline{PAUSE}$ output of the Am9511A is connected to the $\overline{WAIT}$ line of Z80. The Am9511A outputs a LOW on $\overline{PAUSE}$ 150 ns (MAX) after $\overline{RD}$ or $\overline{WR}$ has become active. The $\overline{PAUSE}$ remains LOW for 3.5 TCY + 50 ns (MIN) for data read and is LOW for 1.5 TCY + 50 ns (MIN) for status read from Am9511A where TCY is the clock period at which Am9511A is running. Therefore, the Z80 will insert one to two extra $\overline{WAIT}$ states. The Am9511A $\overline{PAUSE}$ output responds to a data read, data write, or command write request received while the Am9511A is still busy (executing a previous command) by pulling the $\overline{PAUSE}$ output LOW. Since $\overline{PAUSE}$ and $\overline{WAIT}$ are tied together, as soon as the Z80 tries to interfere with APU execution, Z80 enters the $\overline{WAIT}$ state.

7.1.2 Z80 TO Am9512 ARITHMETIC PROCESSOR INTERFACE

Block diagram for this interface is shown in Figure 7-1.2. The Am9512 interface to Z80 requires two more gates than the Am9511A interface to Z80. An inverter is added to the interrupt request line because the sense of the $\overline{END}$/ERR signals are different. The 74LS32 is added in the $\overline{WAIT}$ line because the Am9512 $\overline{PAUSE}$ will go LOW whenever chip select on the Am9512 goes LOW. In Figure 7-1.2 the chip-select input can go LOW during second or third cycles of an instruction when the memory address matches the Am9512 I/O address. If the 74LS32 OR-gate is omitted, the $\overline{WAIT}$ input on the Z80 will to LOW and the system will be deadlocked. Strobing the chip-select decoder will not work because this would cause a negative chip select to $\overline{RD}$ or $\overline{WR}$ time on the Am9512.

The chip select decoder in this example is strobed with M1. This accomplishes a dual purpose. It not only guarantees a chip select transition on every I/O cycle, it also prevents the chip select from going LOW during an interrupt acknowledge cycle. This is vital because $\overline{IORQ}$ is also LOW during that cycle. Without the M1 strobe, $\overline{CS}$ might go LOW and cause $\overline{PAUSE}$ to go LOW which will again cause the system to deadlock.



02188A-83

Figure 7-1.2 Z80 to Zm9512 Interface

7.1.3 Z80 AND Am9513A SYSTEM TIMING CONTROLLER

Interfacing the Z80 and the Am9513A requires careful examination of timing parameters of both devices. The Am9513A data sheet specifies the timing requirements for $\overline{CS}$, C/$\overline{D}$, $\overline{RD}$, $\overline{WR}$, and DATA signals. Figure 7-1.3a shows timing relationship requirements for the Am9513A during a Read cycle. The $\overline{CS}$ to $\overline{RD}$ setup time is 20 ns. This timing specification will not be met, with the Z80, because the CPU generates $\overline{RD}$ signal at a maximum of 10 ns after $\overline{IORQ}$; $\overline{IORQ}$ is the "enable" for the Am29806, Figure 7-1.3g, which produces the $\overline{CS}$ signal for the Am9513A. During the Write cycle (Figure 7-1.3b), $\overline{WR}$ to $\overline{CS}$ Hold time is 20 ns. This again will not be compatible for both devices because the Z80 specifies $\overline{IORQ}$ will go inactive at the same time $\overline{WR}$ goes inactive. Generally, violation of these timing parameters is not a serious problem. In most data sheets, $\overline{CS}$ and $\overline{RD}$ or $\overline{WR}$ are specified as in Figure 7-1.3c. Designers often have difficulties meeting these requirements.

Since $\overline{CS}$ and $\overline{RD}$ or $\overline{WR}$ are actually ANDed inside the device, the specification must be interpreted with some common sense. T_{pw} only describes the shortest $\overline{RD}$ or $\overline{WR}$ pulse. There is nothing wrong with it starting earlier (Figure 7-1.3d). There is also nothing wrong with $\overline{RD}$ or $\overline{WR}$ lasting longer (Figure 7-1.3e). The Write Data Hold time is then related to the end of $\overline{CS}$ and its value is increased by T_s.

There are many possible combinations to accomplish the desired effect. Figure 7-1.3f shows a model that ascertains the legal operating conditions. The delay box represents the setup time of $\overline{CS}$ with respect to $\overline{RD}$ or $\overline{WR}$. It also should be clear that this time needs to be added to the Data Hold time since the internal $\overline{WR}$ will be delayed by this amount. This time must also be added to the Read Access time because the internal $\overline{RD}$ is also delayed by the same amount.

Figure 7-1.3g shows the interface between the Z80 and the Am9513A System Timing Controller, along with the comparator-decoder Am29806 and a dip-switch.

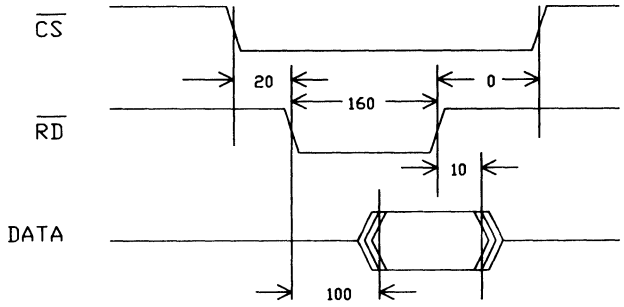
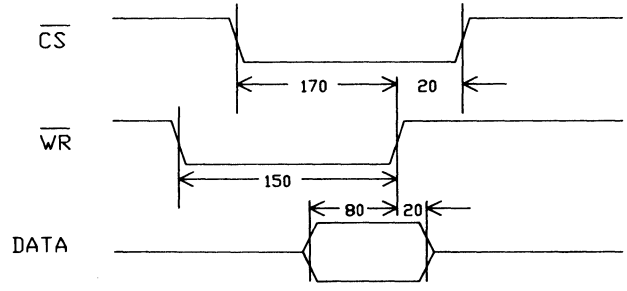


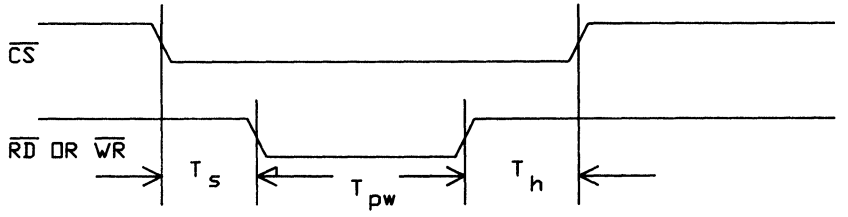
Figure 7-1.3a

02188A-84



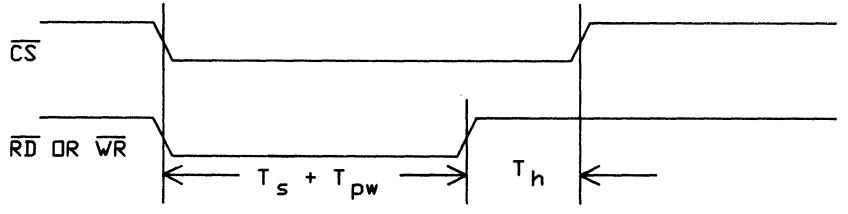
02188A-85

Figure 7-1.3b



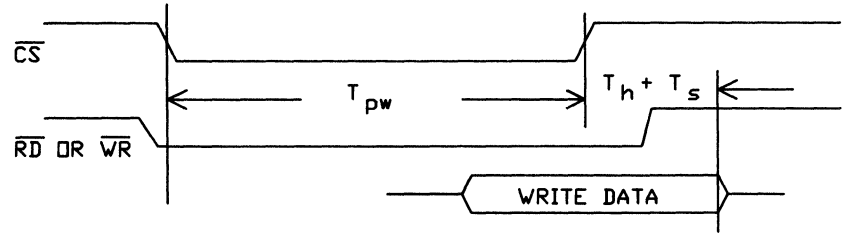
02188A-86

Figure 7-1.3c



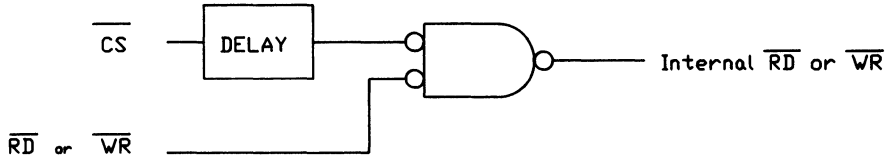
02188A-87

Figure 7-1.3d



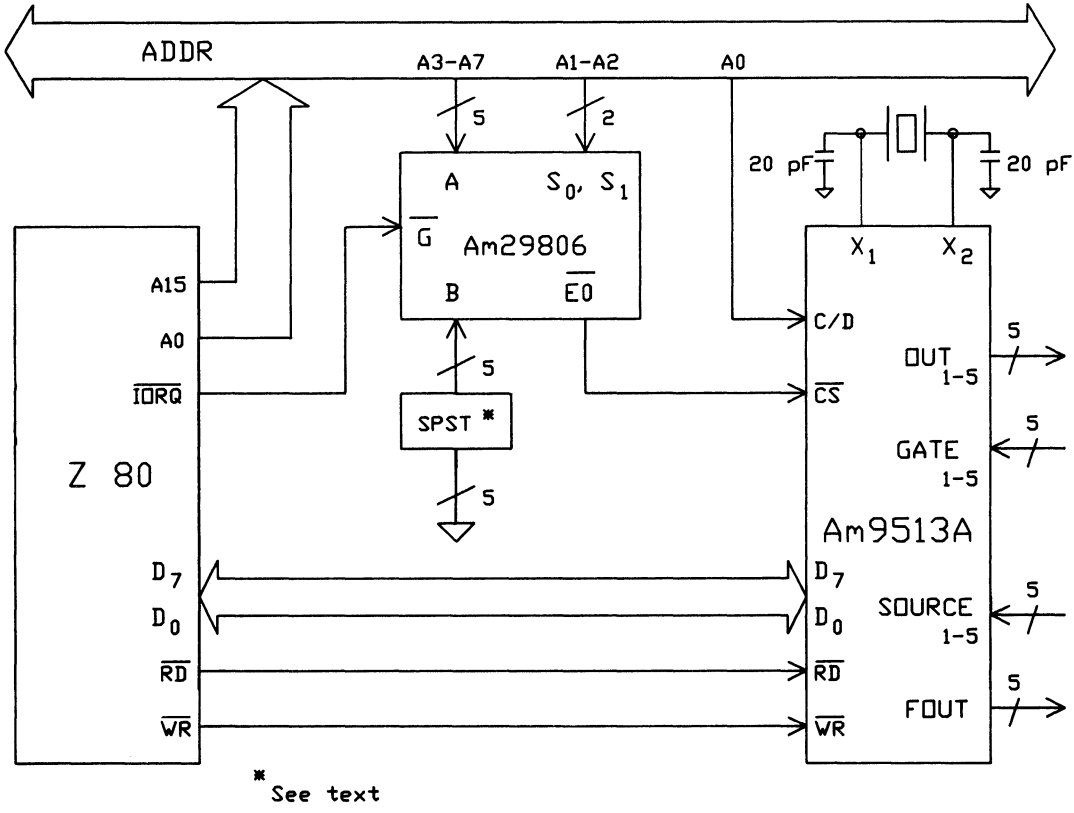
02188A-88

Figure 7-1.3e



02188A-89

Figure 7-1.3f



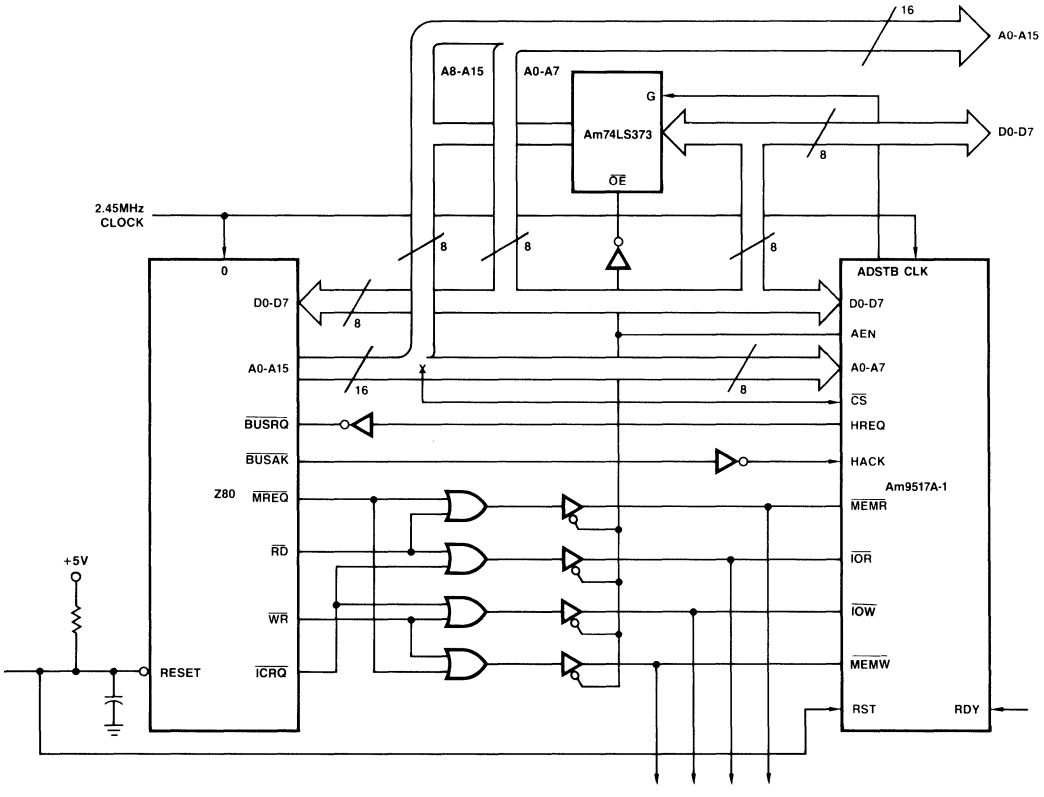
02188A-90

Figure 7-1.3g

7.1.4 Z80 AND Am9517A DMA CONTROLLER

The circuit diagram in Figure 7-1.4a shows the interface involving the Am9517A with the Z80 microprocessor. The high order A8-A15 memory address is latched by an 8-bit Am74LS373 latch, while the $\overline{IOR}$, $\overline{IOW}$, $\overline{MEMR}$ and $\overline{MEMW}$ signals are decoded from appropriate Z80 outputs.

Since the Am9517A can be cascaded to provide multiple controllers in a system, Figure 7-1.4b shows some possibilities of latch hookups for both single and double controller configurations. Figure 7-1.4c shows the general expansion scheme of the Am9517A and Figure 7-1.4d shows the interconnections for interfacing a single DMA controller to a microprocessor.



02188A-91

Figure 7-1.4a Z80 to Am9517A Interface

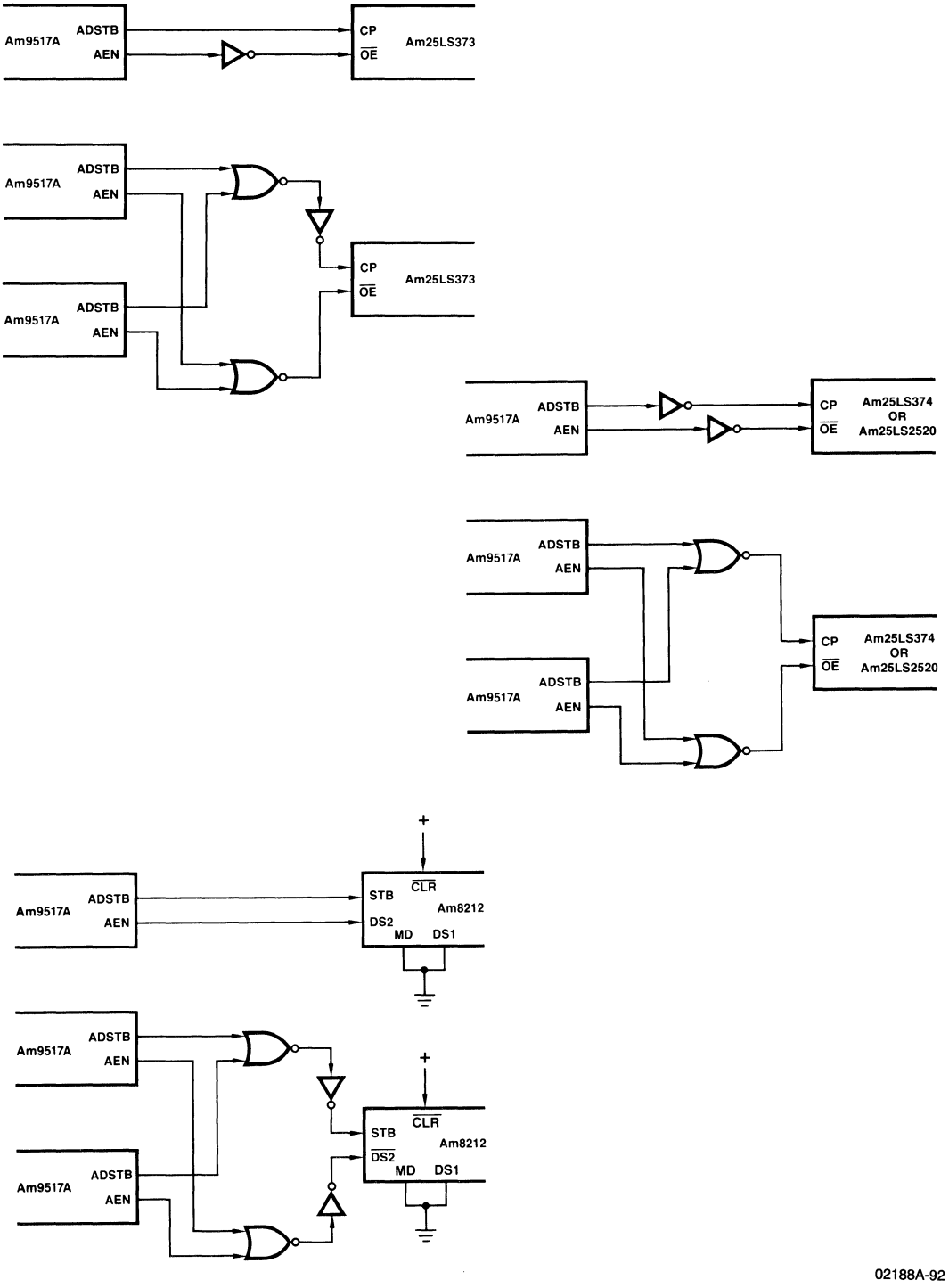
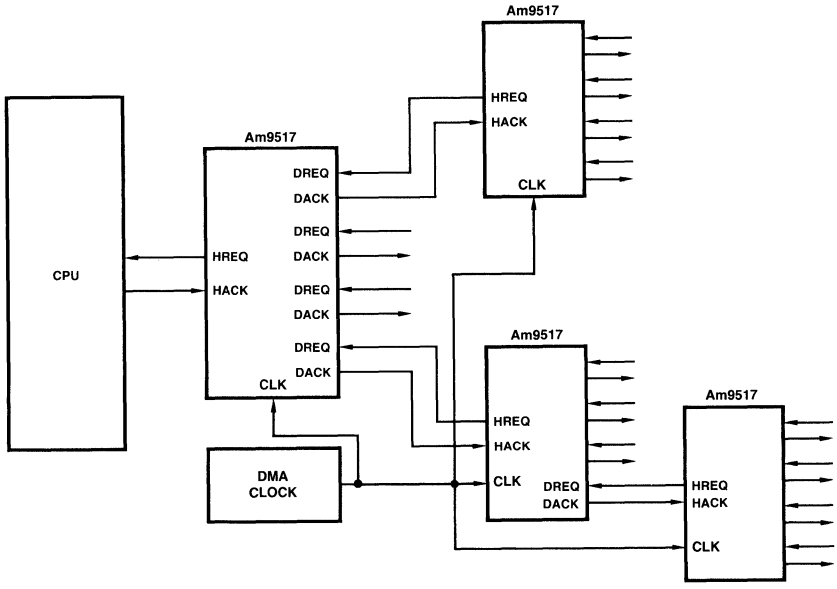

Am9517A
ADSTB
AEN
CP
OE
Am25LS373
Am25LS374
OR
Am25LS2520
+
CLR
STB
DS2
MD
DS1
Am8212
02188A-92

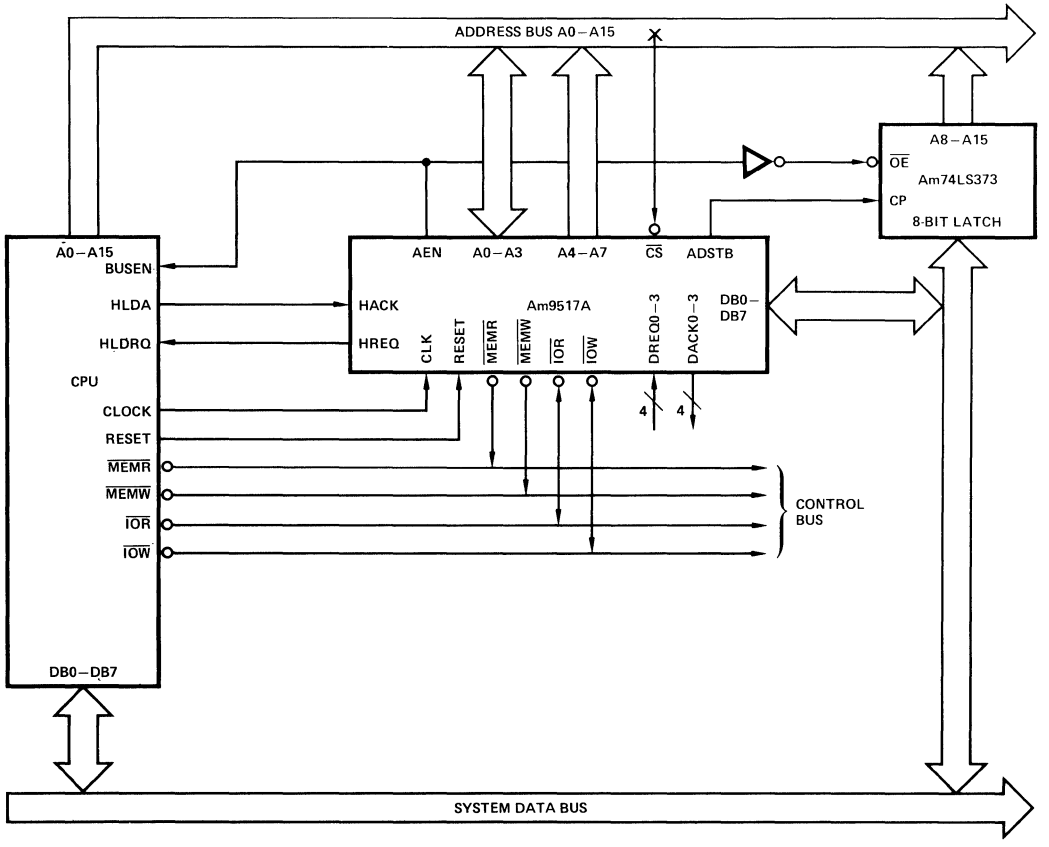
Figure 7-1.4b



Note: All Am9517A devices must use common clock in Cascade mode.

02188A-93

Figure 7-1.4c Cascade Configuration



02188A-94

Figure 7-1.4d Basic DMA Configuration

7.1.5 Z80 AND Am9568 DATA CIPHERING PROCESSOR WITH THE Am9517 DMA CONTROLLER

This application design shows how to operate the ciphering throughput up to 890 Kbyte/s by using the advanced 8-bit DMA Controller Am9517A-5 (also called the 8237-5). The host CPU is a Z80A. (Figure 7-1.5a)

The CPU sets up a data block in memory and programs the DMA controller to transfer this data block to the DCP via the Master Port. The DCP encrypts the data. A high-speed peripheral device can read out the ciphered data from the Slave Port. This dual-port configuration allows data input and output simultaneously and increases the throughput, compared to a single-port configuration, by a factor of two. In the single-port configuration, only the Master Port is used for data transfer; it handles both the clear and ciphered data.

The multiplexed address/data bus of the DCP is simulated in a two-cycle operation. For output operation to an even address, the PAL interface timing controller generates a Master Port Address Strobe ($\overline{MAS}$) to select one of the internal registers. Subsequent I/O operations to an odd address (A_0=HIGH) transfer data to or from the preselected DCP register. During I/O operations to an odd address, the PAL device generates Master Port Data Strobes ($\overline{MRD}$ or $\overline{MWR}$). Before the DMA block transfer starts, the CPU must preselect the DCP data register. The register address of the data register is 00_H.

The DMA controller operates in "flyby" mode. Data is transferred on the system data bus one byte at a time from memory to the DCP, or vice versa, without going through a DMA register. An I/O Read ($\overline{IOR}$) and Memory Write ($\overline{MEMW}$) or I/O Write ($\overline{IOW}$) and Memory Read ($\overline{MEMR}$) are active at the same time. The DCP is selected by DMA Acknowledge ($\overline{DACK}$). The PAL device treats $\overline{DACK}$ as $\overline{CS}$ active and A_0=HIGH. In this design the DMA controller can only execute data transfer cycles; it is not able to change the internal register address of the DCP.

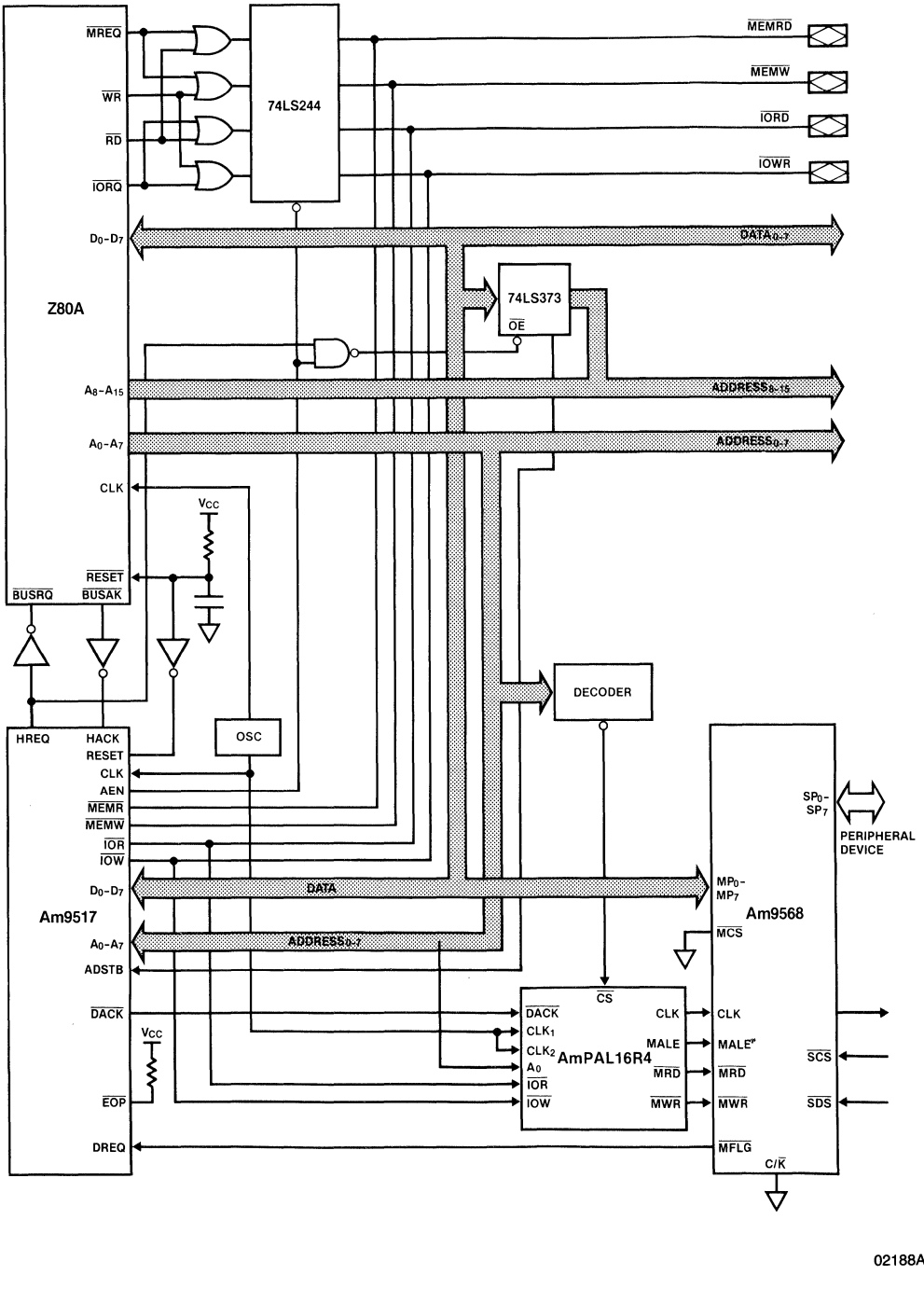


Figure 7-1.5a Z80 and Am9568 DCP Interface

The DMA controller is set up for Demand Transfer Mode. It releases the bus when the data request input goes inactive. The Master Port Flag ($\overline{MFLG}$) is wired to the data request input. The flag output goes active when the DCP is ready to accept data, or the output data is ready to be read out. After transferring one block of data (8 bytes), this flag goes inactive until a new block can be put in or read out. The inactive time depends on the response time of the peripheral logic at the Slave Port. This flag is inactive for a minimum of five clocks.

SPEED

The DMA controller needs three clock cycles to transfer one byte. After each block transfer (8 bytes), the DMA controller releases the bus and requests it back if $\overline{MFLG}$ goes active again. This time is assumed to be 12 clocks. The ciphering of one block is done concurrently with the input of the next block; the internal operation is pipelined. The maximum throughput can be calculated as:

$T = 8 / (8 * 3 + 12) * 4\ MHz = 0.89\ Mbyte/s$

The Compressed Transfer mode of the DMA controller cannot be used, because the PAL synchronization logic needs normal timing to synchronize the Data Strobes to the DCP clock.

INITIALIZATION

The Multiplexed Control Mode ($C/\overline{K}$ =LOW) of the DCP is selected to enable access to the internal registers. The CPU first programs the Mode Register to reset the DCP and to set up the port configuration and ciphering mode. After that, the keys and initial vectors can be loaded. To initialize the DCP for DMA transfer, the CPU executes one Address Latch Cycle, to pre-select the data register.

The DMA controller must be programmed such that $\overline{DREQ}$ and $\overline{DACK}$ are active LOW.

TIMING

The PAL device simulates the multiplexed address/data bus of the DCP assuming a two-cycle operation mode. In the first cycle the CPU latches the address of the internal register into the DCP; subsequent cycles transfer data to or from the selected register. Address A_0 distinguishes the two cycles (Figure 7-1.5b). An I/O instruction with A_0=LOW generates an address latch cycle; an I/O instruction with A_0=HIGH generates a data transfer cycle.

The DMA controller must be initialized for "extended" I/O write in order to have a similar I/O bus timing to the Z80A CPU. A "late" I/O write delays the Master Port Write Strobe ($\overline{MWR}$) to the DCP by one clock cycle. If a late write is used, the data bus will not be valid at the time data is latched.

To execute a DCP-to-memory transfer, the DMA does an I/O read and memory write. The DMA controller can be programmed for an "extended" or "late" write, depending on the memory design.

In "flyby" mode the DMA controller generates no I/O address, so the CPU has to preselect the data Input or Output Register. A DMA Acknowledge ($\overline{DACK}$) enables $\overline{MRD}$ or $\overline{MWR}$ to control the data transfer.

Figure 7-1.5b shows the DMA-DCP data transfer timing. When the DMA Controller has transferred one block of data, the data transfer has to be stopped until the DCP is ready for the next block transfer. The DCP makes the DMA Controller stop the transfer by deactivating $\overline{MFLG}$. If $\overline{MFLG}$ is LOW, data may be transferred; if $\overline{MFLG}$ is HIGH, the DCP does not accept data transferred. The timing of the $\overline{MFLG}$ to DREQ path is the most critical in this application. If $\overline{MFLG}$ is deactivated too late, the DMA Controller will issue another data transfer which will be disregarded by the DCP. The critical signal path will be analyzed below.

To prevent the DMA from issuing another cycle the Data Request input has to go inactive by the falling edge of the DMA clock at the end of cycle S3. The DMA controller samples the input at this time and instigates another cycle if the request is still active. The setup time of DREQ is 0 ns. The Master Port Flag which is connected to the DREQ input goes inactive in the eighth cycle with a maximum delay time of 150 ns after the Data Strobes. The Data Strobe itself has a maximum delay time of 190 ns (Am9517A-5) after the rising edge of the clock in cycle S_2. That gives a time window of 375 ns of which 340 ns are already used for the two delays (190 ns + 150 ns). The propagation delay of a fast PAL device is 25 ns. This leaves 10 ns for other delays in the signal path.

The PAL design assumes that the system memory needs no Wait States.

The peripheral logic at the Slave Port can use the Slave Port Flag ($\overline{SFLG}$) to time the transfer. If $\overline{SFLG}$ is active LOW, data can be written to or read from the data register.

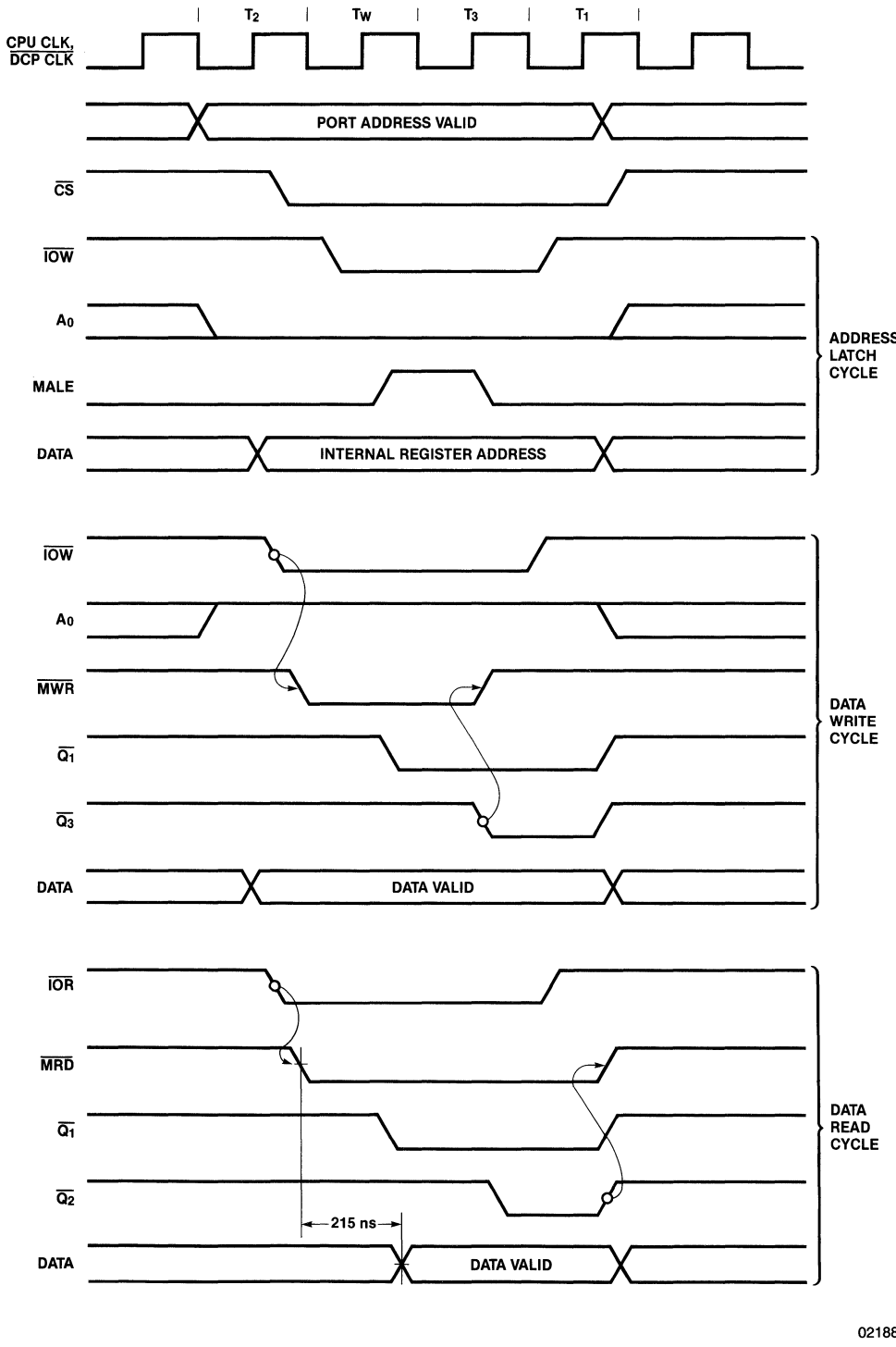


Figure 7-1.5b CPU-DCP Timing Diagram

```
PAL16R4                                          PAL DESIGN SPECIFICATION
DCP048                                           JUERGEN STELBRINK  8-9-83
Z80A- AM9517(DMA)- AM9568(DCP) INTERFACE DEVICE
ADVANCED MICRO DEVICES

CLK1   CLK2   /CS    /IOR   /IOW   A0     /MFLG  /DACK  NC    GND
/OE    /MWR   /MRD   /Q1    /Q2    /Q3    MALE   NC     CLK   VCC

/MALE :=  /IOW+IOR+/CS+A0+MALE          ; MASTER PORT ADDRESS STROBE

Q1    :=  CS*A0*IOR*/IOW*/Q2    +
          CS*A0*IOW*/IOR*/Q3    +
          DACK*IOR*/IOW*/Q2     +
          DACK*IOW*/IOR*/Q3

Q2    :=  CS*A0*IOR*/IOW*Q1     +
          CS*A0*IOR*/IOW*Q2     +
          DACK*IOR*/IOW*Q1      +
          DACK*IOR*/IOW*Q2

Q3    :=  CS*A0*IOW*/IOR*Q1     +
          CS*A0*IOW*/IOR*Q2     +
          DACK*IOW*/IOR*Q1      +
          DACK*IOW*/IOR*Q2

MRD    =  CS*A0*IOR*/IOW        +      ; MASTER PORT READ
          DACK*IOR*/IOW         +
          Q2

MWR    =  CS*A0*IOW*/IOR*/Q3    +      ; MASTER PORT WRITE
          DACK*IOW*/IOR*/Q3

/CLK  =   CLK2                          ; DCP CLOCK


FUNCTION TABLE

CLK1 CLK2 /CS /IOR /IOW /DACK A0   CLK MALE /MRD /MWR /Q1 /Q2 /Q3

;                  /
; C  C      /   /  D          M  /  /
; L  L  /   I   I  A       C  A  M  M  /  /  /
; K  K  C   O   O  C  A    L  L  R  W  Q  Q  Q
; 1  2  S   R   W  K  0    K  E  D  R  1  2  3      COMMENT
-------------------------------------------------------------
; CLOCK GENERATION
  X  L  X   X   X  X  X    H  X  X  X  X  X  X
  X  H  X   X   X  X  X    L  X  X  X  X  X  X
;  ADDRESS LATCH
  C  X  H   H   H  H  L    X  L  H  H  H  H  H   ; CYCLE T2 (CPU)
  C  X  L   H   L  H  L    X  H  H  H  H  H  H   ; CYCLE TW
  C  X  L   H   L  H  L    X  L  H  H  H  H  H   ; CYCLE T3
  C  X  H   H   H  H  L    X  L  H  H  H  H  H   ; CYCLE T1
; READ DATA
  X  X  H   H   H  H  H    X  L  H  H  H  H  H   ; CYCLE TW (CPU)
  X  X  L   L   H  H  H    X  L  L  H  H  H  H
  C  X  L   L   H  H  H    X  L  L  H  L  H  H
  C  X  L   L   H  H  H    X  L  L  H  L  L  H   ; CYCLE TW (EXTRA WAIT STATE)
  C  X  L   L   H  H  H    X  L  L  H  H  L  H   ; CYCLE T3
  C  X  H   H   H  H  H    X  L  H  H  H  H  H   ; CYCLE T1
  X  X  H   L   H  L  X    X  L  L  H  H  H  H   ; CYCLE S3 (DMA)
  C  X  H   L   H  L  X    X  L  L  H  L  H  H
  C  X  H   L   H  L  X    X  L  L  H  L  L  H   ; CYCLE S4
  C  X  H   H   H  H  X    X  L  H  H  H  H  H   ; CYCLE S2
```

```
; WRITE DATA
  X  X  L  H  L  H  H    X  L  H  L  H  H  H   ; CYCLE TW (CPU)
  C  X  L  H  L  H  H    X  L  H  L  L  H  H
  C  X  L  H  L  H  H    X  L  H  H  L  H  L   ; CYCLE T3
  C  X  H  H  H  H  H    X  L  H  H  H  H  H   ; CYCLE T1
  X  X  H  H  L  L  H    X  L  H  L  H  H  H   ; CYCLE S3 (DMA)
  C  X  H  H  L  L  H    X  L  H  L  L  H  H
  C  X  H  H  L  L  H    X  L  H  H  L  H  L   ; CYCLE S4
  C  X  H  H  H  H  H    X  L  H  H  H  H  H   ; CYCLE S2
;
```

DESCRIPTION:

THIS PAL GENERATES ALL NECESSARY BUS CONTROL SIGNALS, TO INTERFACE A Z80A CPU AND A AM9517 DMA CONTROLLER TO THE AM9568 DATA CIPHERING PROCESSOR. THE MAXIMUM SYSTEM CLOCK FOR ALL PARTS IS 4 MHZ.

1 INPUT AND 3 INPUT/ OUTPUT PINS ARE NOT USED.

INPUT SIGNALS:

CLK1, CLK2 — Z80 SYSTEM CLOCK

/CS — CHIP SELECT FOR THE DCP, GENERATED BY A DECODER LOGIC

/IOR — INPUT/OUTPUT READ

/IOW — INPUT/OUTPUT WRITE

A0 — LEAST SIGNIFICANT BIT OF THE Z80 ADDRESS BUS TO SELECT THE TYPE OF OPERATION:
- A0 = LOW — SELECT DCP REGISTER FOR NEXT DATA CYCLES (ADDRESS LATCH)
- A0 = HIGH — READ OR WRITE INTERNAL REGISTER (DATA TRANSFER TO CONTROL, MODE, INPUT OR OUTPUT REGISTER)

/DACK — DMA ACKNOWLEDGE FROM DMA CONTROLLER, TREATED AS /CS=LOW AND A0=HIGH

OUTPUT SIGNALS:

CLK — INVERTED SYSTEM CLOCK FOR THE DCP

MALE — MASTER PORT ADDRESS LATCH ENABLE, ACTIVE DURING ADDRESS LATCH CYCLES TO LATCH THE REGISTER ADDRESS ON MP1 AND MP2 (2 LINES OF THE MASTER PORT BUS) AND THE STATE OF /MCS IN. THE DCP STORES INTERNALLY THE ADDRESS AND CHIP SELECT TO THE NEXT ADDRESS LATCH CYCLE

/MRD — MASTER PORT READ, TO ENABLE REGISTER READ OPERATIONS

/MWR — MASTER PORT WRITE, TO ENABLE REGISTER WRITE OPERATIONS

/Q1, /Q2, /Q3 — INTERNAL USED STATE SIGNALS (DO NOT CONNECT). Q1 IS ACTIVE 2 CLOCK CYCLES IN EACH DATA TRANSFER OR DMA ACKNOWLEDGE CYCLE. IT IS USED TO GENERATE THE DELAYED Q2 AND Q3. Q2 IS USED TO HOLD /MRD ACTIVE UNTIL /IOR IS GONE INACTIVE. Q3 MASKS /MWR OFF.

7.1.6 Z80 TO Am9518/Am9568 DCP

This chapter shows, in two examples, how the Data Ciphering Processor (DCP) can be interfaced to a Z80 (Z80A, Z80B) CPU (Figure 7-1.6a). All interface control signals are generated by one PAL device.

In CPU transfer mode,ciphering speed can reach up to 280 Kbyte/s. A Z80A DMA controller can double this value, and a Z80-DMA-DCP hookup can increase the speed to 1.1 Mbyte/s.

The multiplexed address/data bus of the DCP is simulated using a two-cycle operation mode. An output instruction to an even address (A_0=LOW) selects one of the internal registers of the DCP. In all subsequent I/O operations with A_0=HIGH, the CPU can transfer data to or from DCP registers. The register address stays latched in the chip until the next Address Strobe latches in a new address. The Address Latch Cycle does not represent significant overhead in an encryption or decryption session because, once the DCP is initialized and the data register is selected, no further Address Latch Cycle is needed.

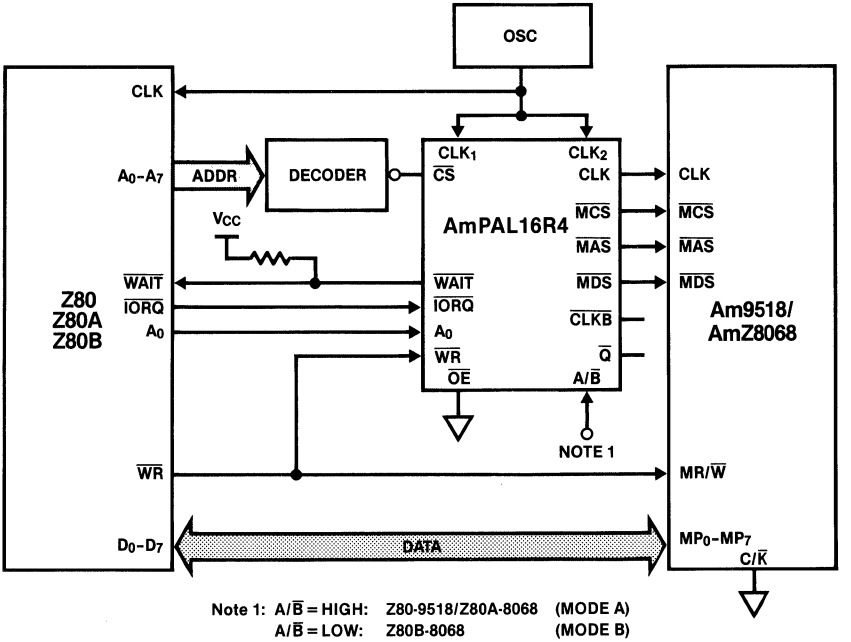
I/O addresses: XXXX XXX0 - Address Latch Cycle
XXXX XXX1 - Data Transfer Cycle
X - User definable

The AmPAL16R4 device controls the interface timing. It generates the synchronized strobe signals for the DCP and the Wait for the CPU to extend the cycles.

The PAL device is programmed to allow two operation modes. In Mode A the DCP works with the same clock rate as the CPU. Mode B increases the ciphering speed by allowing higher than 4-MHz system clock rates for the CPU. In this mode, the PAL device provides half the system clock rate for the DCP.

A system with a Z80B at 6 MHz and an AmZ8068 at 3 MHz increases the ciphering speed compared to a system where both the CPU and the DCP clock are 4 MHz; the limiting factor is the data transfer capability of the CPU.

The key requirement in interfacing the DCP to a Z80 CPU is to meet the timing relationship between the Master Port Data Strobe ($\overline{MDS}$) and the DCP clock. The rising edge of $\overline{MDS}$ must be synchronous to the falling edge of the clock.



Note 1: A/B = HIGH: Z80-9518/Z80A-8068 (MODE A)
A/B = LOW: Z80B-8068 (MODE B)

02188A-97

Figure 7-1.6a Z80-DCP Interface

THE OPERATION MODES

Mode A: Both the Z80 CPU and the DCP are operating synchronously at the same frequency. The DCP clock is inverted. This mode can be used with system clocks up to 4 MHz. No extra Wait States are inserted.

Mode B: To get higher ciphering throughput, the data transfer speed of the Z80 bus should be increased by using a higher system clock rate. In Mode B the PAL device divides the system clock by two to generate the DCP clock. The DCP clock is synchronized to the $\overline{MDS}$ by delaying the clock one half cycle if they are not in phase (Figures 7-1.6d and 7-1.6e). During a Data Write Cycle, one extra Wait State is inserted. An AmZ8068 must be used in this mode, even at a DCP clock rate of 3 MHz, because of its faster register access time.

Figure 7-1.6a shows the interface. The A/$\overline{B}$ input of the PAL device is wired HIGH to select Mode A or LOW to select Mode B.

THE INTERFACE TIMING

ADDRESS LATCH CYCLE: (FIGURES 7-1.6b AND 7-1.6c)

Master Port Chip Select ($\overline{MCS}$) is active when $\overline{IORQ}$ and $\overline{CS}$ are active LOW and A_0=LOW (even address). Master Port Address Strobe ($\overline{MAS}$) is strobed LOW for one system clock cycle during the Z80's automatically inserted Wait cycle T_W to meet the hold time requirement of $\overline{MAS}$ HIGH to $\overline{MCS}$ HIGH (parameter 35).

DATA READ CYCLE: (FIGURES 7-1.6b AND 7-1.6c)

A Data Read Cycle reads the register whose address was latched in the previous Address Latch Cycle. $\overline{MCS}$ and $\overline{MAS}$ are inactive the whole cycle. $\overline{MDS}$ is active during the last two clock cycles, T_W and T_3. In both A and B Modes, no Wait State is inserted. $\overline{WR}$ and A_0 must be HIGH. In Mode B the DCP clock is set HIGH in the beginning of T_3 using an internal signal $\overline{Q}$ to synchronize the falling edge of the DCP clock to the rising edge of $\overline{MDS}$. $\overline{Q}$ is only active in Mode B during Wait State T_W. This interface meets the data hold time of the Z80, because the data is stable to the beginning of T_1 of the next machine cycle.

DATA WRITE CYCLE

In this cycle, the CPU can write one byte into the addressed register. $\overline{MCS}$ and $\overline{MAS}$ are inactive. $\overline{WR}$ is active and A_0 is HIGH.

Mode A (Figure 7-1.6a)

$\overline{MDS}$ is strobed LOW for T_W. The DCP reads the data in at the beginning of T_3. No Wait State is inserted.

Mode B (Figure 7-1.6d)

$\overline{MDS}$ is strobed LOW for the Wait cycle T_W and the additional Wait cycle T_W to meet the minimum data strobe active time (parameter 44) of the DCP. The DCP reads the data in at the begin of T_3.

DATA CIPHERING SPEED

The byte transfer capability of the Z80 system bus limits the data ciphering throughput of the DCP. A Z80 DMA controller doubles the maximum throughput compared to a CPU-controlled transfer as indicated in the following table:

System Clk	DCP Clk	CPU	DCP	Mode	N	T
6 MHz	3 MHz	Z80B	AmZ8068	B	168/176	0.28/0.27
4 MHz	4 MHz	Z80A	AmZ8068	A	168	0.19
2.5 MHz	2.5 MHz	Z80	Am9518	A	168	0.14

N — Number of DCP clock cycles to transfer and cipher 8 bytes of data. In CPU-controlled modes the use of the Z80 block transfer commands like INIR, INDR, OTIR or OTDR is assumed.

T — Throughput in Mbyte/s

The formula for calculating the throughput is:

T = (8 * f) / (N + m) Mbyte/s

f — DCP clock in MHz

8 — 8 bytes per block

m — Number of extra DCP clock cycles to get a minimum delay time of five clocks between transferring the last byte of one block and the first byte of the next block. In CPU controlled transfers m=0 can be assumed, because the CPU has to evaluate instruction fetches and memory data transfers between two I/O accesses. $\overline{MFLG}$ indicates if the DCP accepts data transfer.

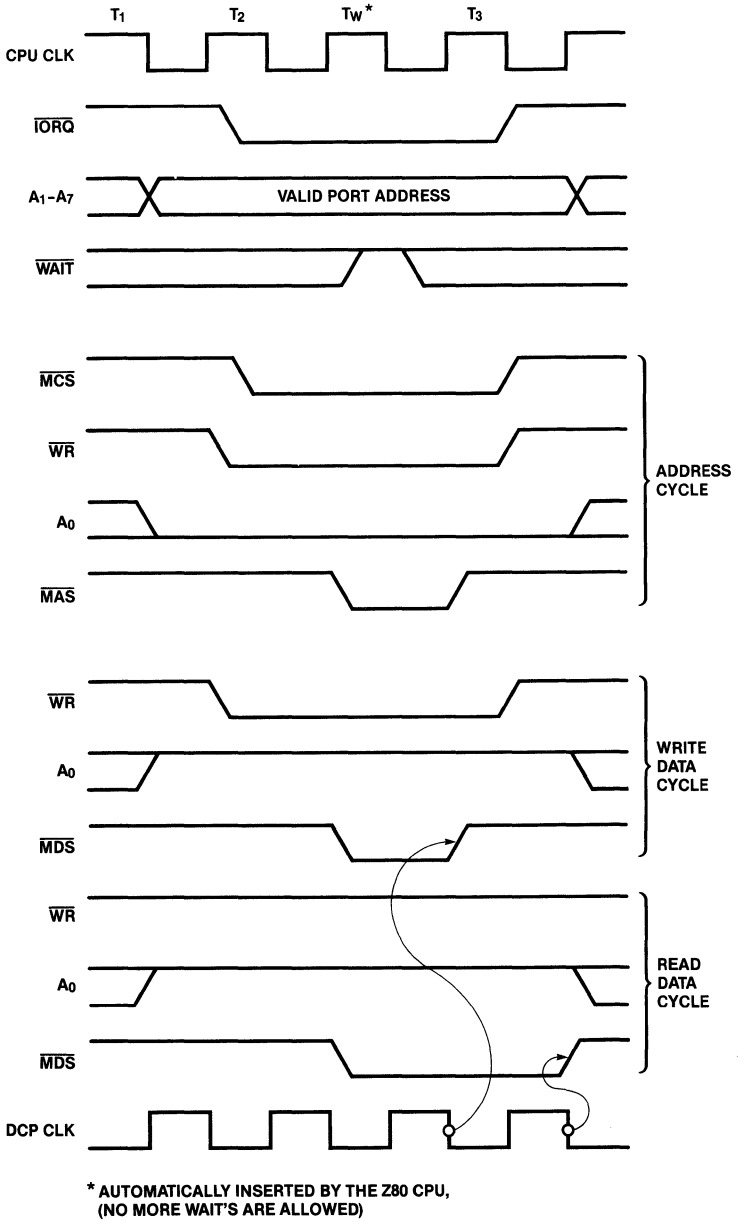


Figure 7-1.6b Z80-Am9518/Z80A-AmZ8068 Timing Diagram (Mode A)

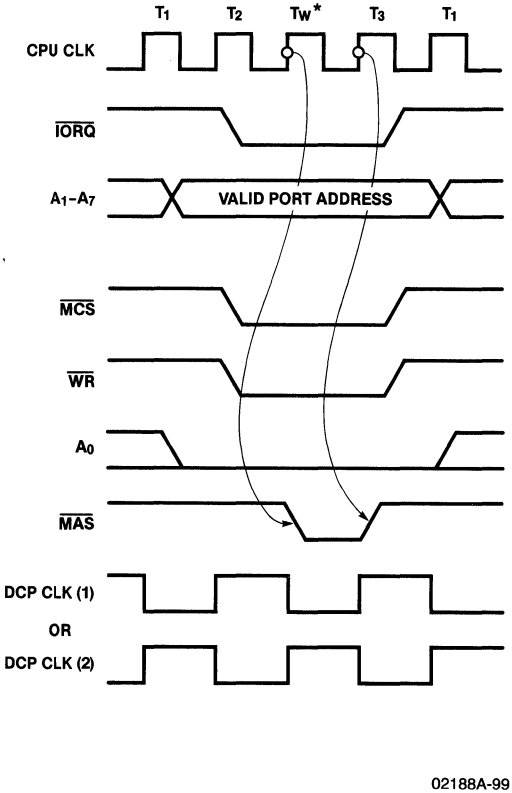


Figure 7-1.6c Address Latch Cycle (Mode B)
(No Clock Synchronization)

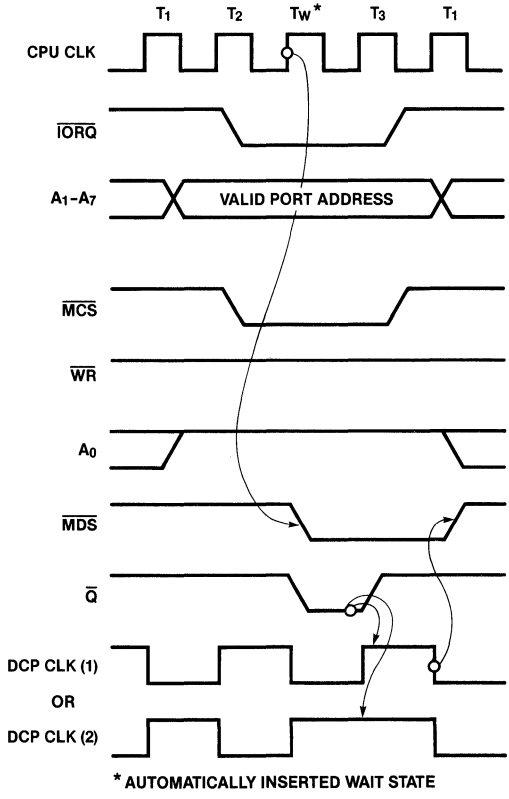
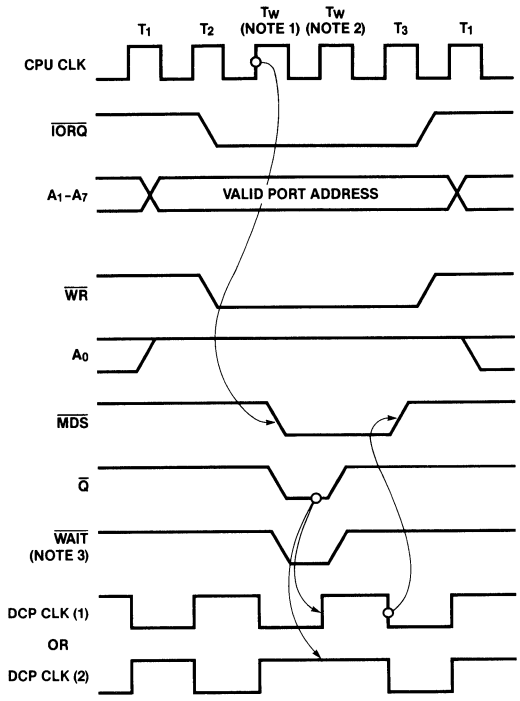


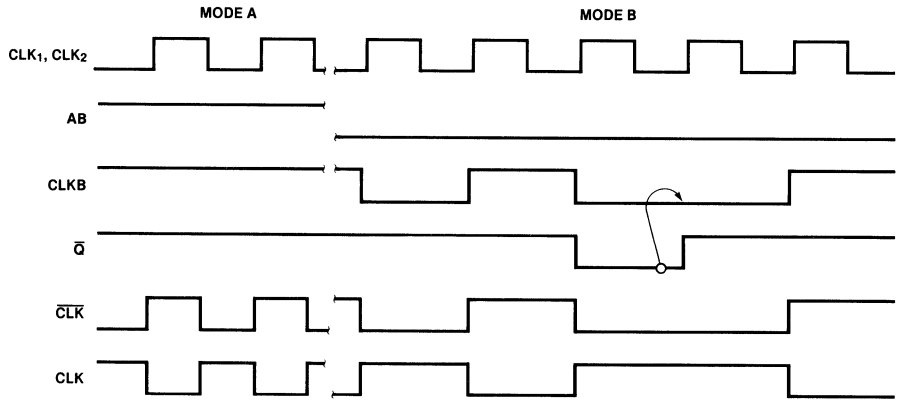
Figure 7-1.6d Data Read Cycle (Mode B)



NOTE: 1. AUTOMATICALLY INSERTED WAIT STATE
2. EXTRA WAIT STATE
3. OPEN COLLECTOR OUTPUT

02188A-101

Figure 7-1.6e Data Write Cycle (Mode B)



02188A-102

Figure 7-1.6f Clock Timing Diagram (Mode A and B)

```
PAL16R4                                        PAL DESIGN SPECIFICATION
DCP046                                         JUERGEN STELBRINK    5/2/83
Z80- AM9518/AMZ8068  INTERFACE CONTROLLER
ADVANCED MICRO DEVICES

CLK1  CLK2  /CS    /IORQ  A0  /WR   AB    NC    NC   GND
/OE   NC    /WAIT  /CLKB  /Q  /MDS  /MAS  /MCS  CLK  VCC

MCS  =   IORQ*CS*/A0                      ; MASTER PORT CHIP SELECT

MAS  :=  IORQ*CS*/A0*WR*/MAS              ; MASTER PORT ADDRESS STROBE

MDS  :=  IORQ*CS*WR*/MDS*A0*AB          + ; WRITE DATA STROBE (MODE A)
         IORQ*CS*WR*A0*/MDS*/Q*/AB      + ; WRITE DATA STROBE (MODE B)
         IORQ*CS*WR*A0*MDS*Q*/AB        + ; WRITE DATA STROBE (MODE B)
         IORQ*CS*/WR*A0                  ; READ  DATA STROBE (MODE A+B)

CLKB :=  /CLKB*/Q*/AB                     ; CLOCK FOR MODE B

/CLK =   CLK2*AB  +                       ; (MODE A)
         CLKB                             ; (MODE B)

Q    :=  IORQ*CS*/MDS*/Q*A0*/AB           ; USED TO GENERATE MDS AND WAIT

IF (Q*WR) WAIT =   Q*WR                   ; WAIT TO Z80

FUNCTION TABLE

 CLK1 CLK2 AB  /CS /IORQ  A0  /WR       CLK /MCS /MAS /MDS /WAIT  /Q /CLKB

;              /                        /
; C  C        I              /  /  /   W     C
; L  L     /  O     /    C  M  M  M   A     L
; K  K  A  C  R  A  W    L  C  A  D   I  /  K
; 1  2  B  S  Q  0  R    K  S  S  S   T  Q  B     COMMENT
-------------------------------------------------------------------------
;
; MODE A:  Z80- AM9518 OR Z80A- AMZ8068 INTERFACE
;          (DCP CLOCK = CPU CLOCK)
;
; CLOCK GENERATION
;
  X  L  H  X  X  X  X    H  X  X  X  Z  H  H
  X  H  H  X  X  X  X    L  X  X  X  Z  H  H
;
; ADDRESS LATCH
;
  H  X  H  H  H  X  H    X  H  H  H  Z  H  H   ; MACHINE CYCLE T1
  L  X  H  L  H  L  H    X  H  H  H  Z  H  H
  C  X  H  L  H  L  H    X  H  H  H  Z  H  H
  H  X  H  L  L  L  L    X  L  H  H  Z  H  H   ; CYCLE T2
  C  X  H  L  L  L  L    X  L  L  H  Z  H  H
  C  X  H  L  L  L  L    X  L  H  H  Z  H  H   ; CYCLE TW
  H  X  H  L  H  L  H    X  H  H  H  Z  H  H   ; CYCLE T3
```

```
  C  X  H  L  H  L  H    X  H  H  H  Z  H  H
;
; WRITE DATA OPERATION
;
  C  X  H  L  H  H  H    X  H  H  H  Z  H  H   ; CYCLE T1
  C  X  H  L  L  H  L    X  H  H  L  Z  H  H   ; CYCLE T2
  C  X  H  L  L  H  L    X  H  H  H  Z  H  H   ; CYCLE TW
  C  X  H  L  H  H  H    X  H  H  H  Z  H  H   ; CYCLE T3
;
; READ DATA OPERATION
;
  C  X  H  L  H  H  H    X  H  H  H  Z  H  H   ; CYCLE T1
  C  X  H  L  L  H  H    X  H  H  L  Z  H  H   ; CYCLE T2
  C  X  H  L  L  H  H    X  H  H  L  Z  H  H   ; CYCLE TW
  C  X  H  L  H  H  H    X  H  H  H  Z  H  H   ; CYCLE T3
;
; INVALID OPERATION (READ IN ADDRESS LATCH)
;
  C  X  H  L  L  L  H    X  L  H  H  Z  H  H   ; NO /MAS !
;
;
;                 /                       /
; C  C            I          /  /  /      W      C
; L  L      /     O     /    C  M  M  M    A      L
; K  K  A   C  R  A  W    L  C  A  D  I  /  K
; 1  2  B   S  Q  0  R    K  S  S  S  T  Q  B    COMMENT
;-------------------------------------------------------------------------
;
; MODE B: Z80B- AMZ8068 INTERFACE
;         (DCP CLOCK = CPU CLOCK/2)
;
; WRITE DATA OPERATION
;
  C  X  L  H  H  H  H    L  H  H  H  Z  H  L   ; CYCLE T1
  C  X  L  H  H  H  H    H  H  H  H  Z  H  H   ; CYCLE T2
  C  X  L  L  L  H  L    L  H  H  L  L  L  L   ; FIRST WAIT CYCLE (CLK=L)
  C  X  L  L  L  H  L    H  H  H  L  Z  H  H   ; SECOND WAIT CYCLE
  C  X  L  L  L  H  L    L  H  H  H  Z  H  L   ; CYCLE T3
;
  C  X  L  L  L  H  L    H  H  H  L  L  L  H   ; FIRST WAIT CYCLE (CLK=H)
  C  X  L  L  L  H  L    H  H  H  L  Z  H  H   ; SECOND WAIT CYCLE (SYNC !)
  C  X  L  L  L  H  L    L  H  H  H  Z  H  L   ; CYCLE T3
;
; READ DATA OPERATION
;
  C  X  L  H  H  H  H    H  H  H  H  Z  H  H   ; CYCLE T1
  C  X  L  H  H  H  H    L  H  H  H  Z  H  L   ; CYCLE T2
  C  X  L  L  L  H  H    H  H  H  L  Z  L  H   ; WAIT CYCLE
  C  X  L  L  L  H  H    H  H  H  L  Z  H  H   ; CYCLE T3 (SYNC!)
  C  X  L  L  H  H  H    L  H  H  H  Z  H  L   ; NEXT CYCLE
;
-------------------------------------------------------------------------
```

DESCRIPTION:

THIS PAL GENERATES ALL NECESSARY BUS CONTROL SIGNALS, TO INTERFACE THE AM9518 OR AMZ8068 TO THE Z80 CPU WITH A SYSTEM CLOCK UP TO 6 MHZ.

2 INPUT AND 1 INPUT/ OUTPUT PINS ARE NOT USED, SO THAT FOR EXAMPLE A DATA BUS TRANSCEIVER CONTROL LOGIC CAN BE ADDED.

IN SYSTEMS WITH A CLOCK UP TO 4 MHZ, THE DCP RUNS DIRECTLY AT THIS FREQUENCY (MODE A, INPUT AB = HIGH).
IF THE FREQUENCY IS HIGHER, THE DCP IS DIVIDED BY TWO FROM THE SYSTEM CLOCK (MODE B, AB = LOW).

INPUT PINS:

CLK1, CLK2 — CLK1 IS THE CLOCK INPUT FOR THE FOUR INTERNAL D-FLIP-FLOPS. THEY ARE CLOCKED BY THE RISING EDGE OF CLK1. THE DCP DATA STROBE MUST BE SYNCHRONOUS TO THE FALLING EDGE OF THE CLOCK; THE INVERTED CLK2 IS THEREFORE SENT TO THE OUTPUT CLK. IN MODE B CLK2 IS SYNCHRONIZED BEFORE IT APPEARS ON THE CLK OUTPUT. BOTH INPUTS ARE CONNECTED TO THE Z80 SYSTEM CLOCK.

/CS — CHIP SELECT GENERATED BY AN ADDRESS DECODER LOGIC (ACTIVE LOW). IF /CS IS ONLY ACTIVE IN I/O CYCLES, THE /IORQ INPUT CAN BE WIRED LOW.

/IORQ — INPUT/ OUTPUT REQUEST OF THE Z80 (LOW ACTIVE)

A0 — LEAST SIGNIFICANT BIT OF THE Z80 ADDRESS BUS TO SELECT TYPE OF OPERATION:
- A0= LOW — SELECT REGISTER FOR NEXT DATA CYCLES (ADDRESS LATCH)
- A0= HIGH — READ OR WRITE INTERNAL REGISTER (DATA TRANSFER TO CONTROL, MODE, INPUT OR OUTPUT REGISTER)

/WR — WRITE SIGNAL OF THE Z80, DEFINES DATA TRANSFER DIRECTION

AB
- AB= HIGH — MODE A
- AB= LOW — MODE B

OUTPUT SIGNALS:

/WAIT — ACTIVE LOW DURING FIRST WAIT CYCLE IN WRITE DATA OPERATION IN MODE B, TO GENERATE AN EXTRA WAIT STATE. THE OTHER TIME /WAIT IS IN THREE STATE.

/MCS — MASTER PORT CHIP SELECT, ONLY ACTIVE IN ADDRESS LATCH CYCLES

/MAS — MASTER PORT ADDRESS STROBE, ACTIVE IN ADDRESS CYCLES TO LATCH THE REGISTER ADDRESS AND /MCS IN. THE DCP STORES INTERNALLY THE ADDRESS AND THE CHIP SELECT

TO THE NEXT ADDRESS LATCH CYCLE

/MDS — MASTER PORT DATA STROBE TO ENABLE DATA TRANSFER TO THE INTERNAL REGISTERS OF THE DCP

CLK — DCP CLOCK, IN MODE B SYNCHRONIZED TO THE MASTER PORT DATA STROBE (/MDS)

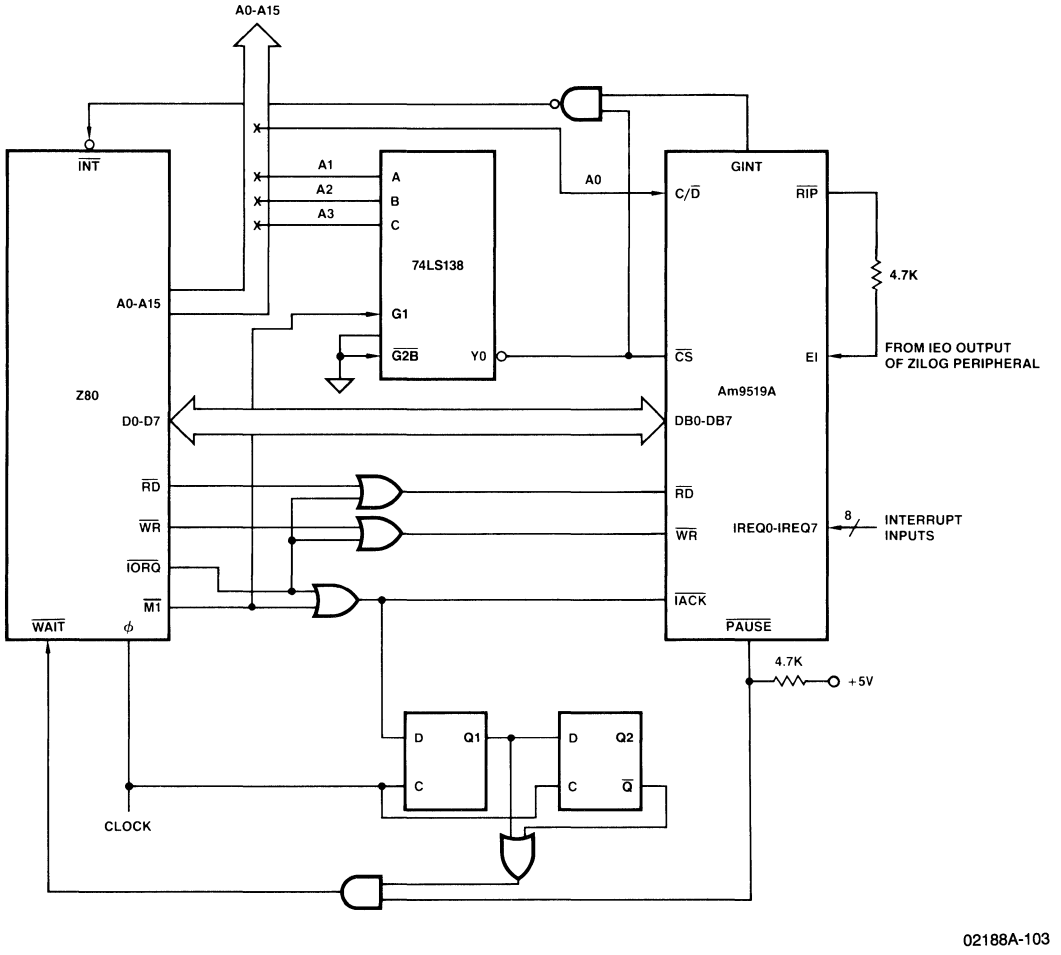
/CLKB — DCP CLOCK OUTPUT INTERNALLY USED FOR MODE B (NOT CONNECT)

/Q — INTERNAL STATUS SIGNAL (NOT CONNECT)

7.1.7 THE Z80 AND Am9519A INTERRUPT CONTROLLER

The Am9519A is an Universal Interrupt Controller, a processor support device designed to enhance the interrupt handling capability of a wide variety of processors. A single Am9519A can manage the masking, priority resolution and vectoring of up to eight interrupts, and it may be easily expanded by the addition of other Am9519A chips to handle a nearly unlimited set of interrupt inputs. It offers many programmable operating options to improve both the efficiency and versatility of the host system operations. The diagram in figure 7-1.7a shows the interface between the Z80 and the Am9519A.

The Am9519A can be configured as the sole interrupting device to the Z80 or in conjunction with other devices (such as the DIO,CIO,CTC, etc.) in a daisy chain interrupt scheme. When the Am9519A is part of a daisy-chain, it must be the lowest priority device because the Am9519A activates its Enable Out (EO) output after receipt of the interrupt acknowledge signal ($\overline{IORQ} \cdot \overline{M1}$) from the microprocessor, while the Zilog parts require their IEI (Interrupt Enable In) signal to setup at least 200 nsec (2.5 MHz Z80) prior to the Interrupt Acknowledge going active. Earlier design examples decoded $\overline{IORQ}$ and $\overline{M1}$ to determine an interrupt acknowledge cycle. Such an approach has the disadvantage of making the $\overline{IACK}$ to occur late in the cycles. The new approach decodes $\overline{MREQ}$ and $\overline{M1}$ instead. The flip-flop ensures glitches that result from a false decoding, during other cycles, do not affect the 9519A. This allows the 9519A sufficient time to assert $\overline{PAUSE}$ and, in most cases, additional Wait States may not be necessary.



02188A-103

Figure 7-1.7a

8.0 INTERFACING TO THE 8085

8.1 OVERVIEW OF THE 8085

The 8085A is a complete 8-bit parallel central processing unit (CPU). Its instruction set is 100% software compatible with the 8080A microprocessor. The 8085A reduces the parts count of 8080A system while increasing the speed. Essentially, it integrates the 8080A, the 8224, and the 8228 into a single chip.

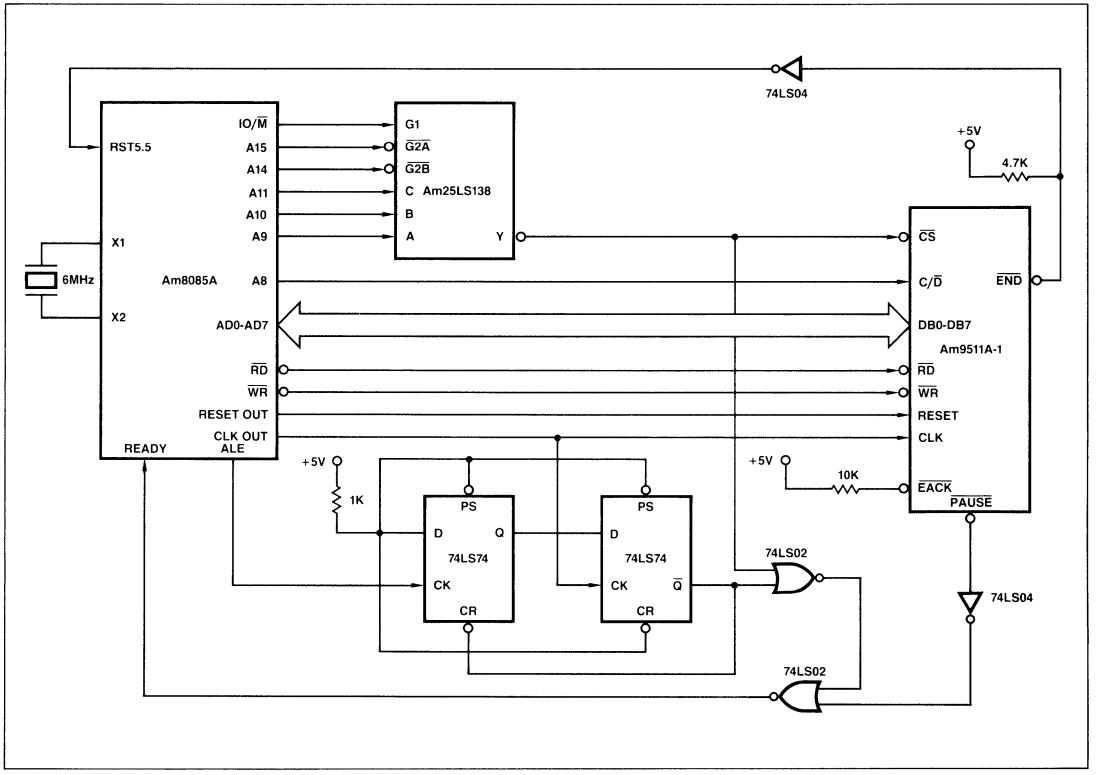
The 8085A uses a multiplexed Data Bus. The address is split between the 8-bit Address Bus and the 8-bit Data Bus. Only 8 bits of address is used for I/O operations. The on-chip address latches of 8155/8355 memory products allow a direct interface with 8085A. 8085A components, including various timing compatible support chips, allow system speed optimization.

The 8080/8085 processors duplicate the I/O address on the upper 8 bits as well as the lower 8 bits of the Address Bus. The upper address bus is used in the following examples because they are latched.

The 8085A-2 is a faster version of the 8085A. The 8085AH is a 3 MHz CPU with 10% supply tolerances and lower power consumption.

8.1.1 THE 8085 AND Am9511A INTERFACE

The system clock rate is 3 MHz in a typical 8085A system. The floating point processor to be used in this kind of interface should be the Am9511-1 because of its 3 MHz maximum clock rate. The Am8085A has an earlier Ready setup window compared with the Am9080A. If the PAUSE signal is connected directly o the READY input of the Am8085A, the ready line will be pulled down too late for the Am8085A to go into the WAIT state. The 74LS74 is used for forcing one WAIT state when the Am9511-1 is accessed. After the first WAIT state, the 74LS74 Q output is reset to HIGH and the PAUSE of the Am9511-1 controls any additional wait states if necessary. The chip-select decoder is gated with IO/$\overline{M}$ signal to prevent the Am9511-1 from responding to memory accesses when bits 9 to 15 of the memory address coincides with Am9511-1 I/O address. Figure 8-1.1 shows the Am8085A to 9511-1 interface.



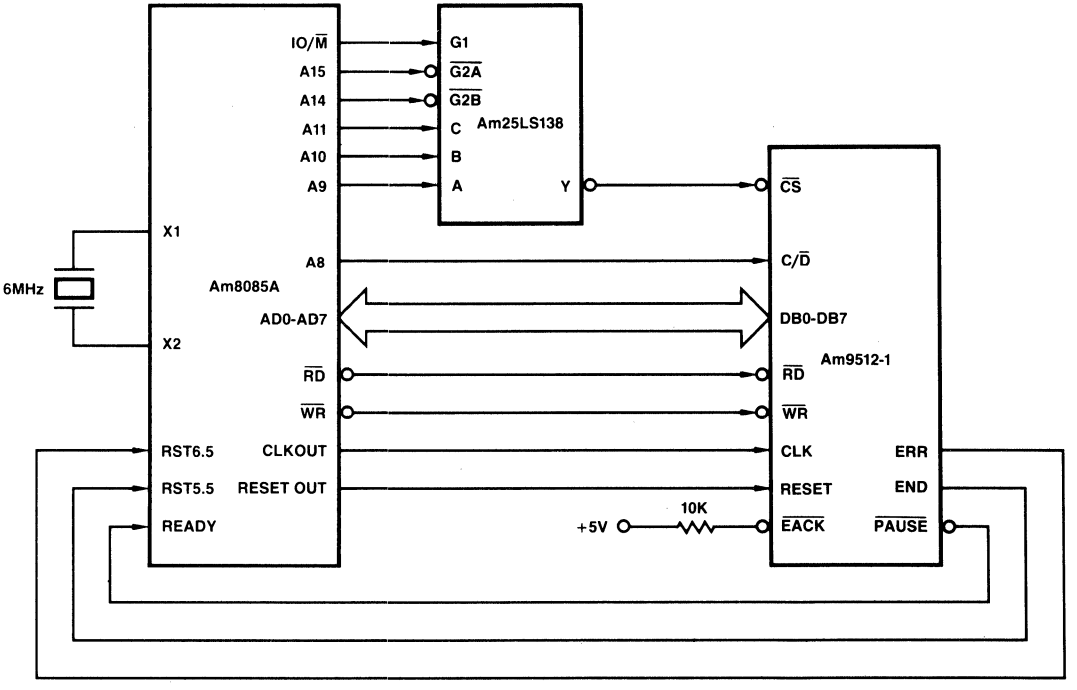
02188A-104

Figure 8-1.1 Am8085A to Am9511A-1 Interface

8.1.2 Am8085A TO Am9512-1 INTERFACE

The Am9512 is designed specifically to interface to the Am8085A. The interface is straightforward and no additional logic is required. The Am9512-1 is used here, instead of the Am9512, because the typical Am8085A system runs at 3 MHz.

The ERR output and END output are connected to separate interrupt inputs so that the CPU can identify the source of interrupt without reading the status register of the Am9512-1. Since the Chip Select decoder is gated by the IO/$\overline{M}$ signal, a transition is guaranteed with each I/O operation without the concern of insufficient address decode. The diagram in Figure 8-1.2 shows the Am8085A to Am9512-1 interface.

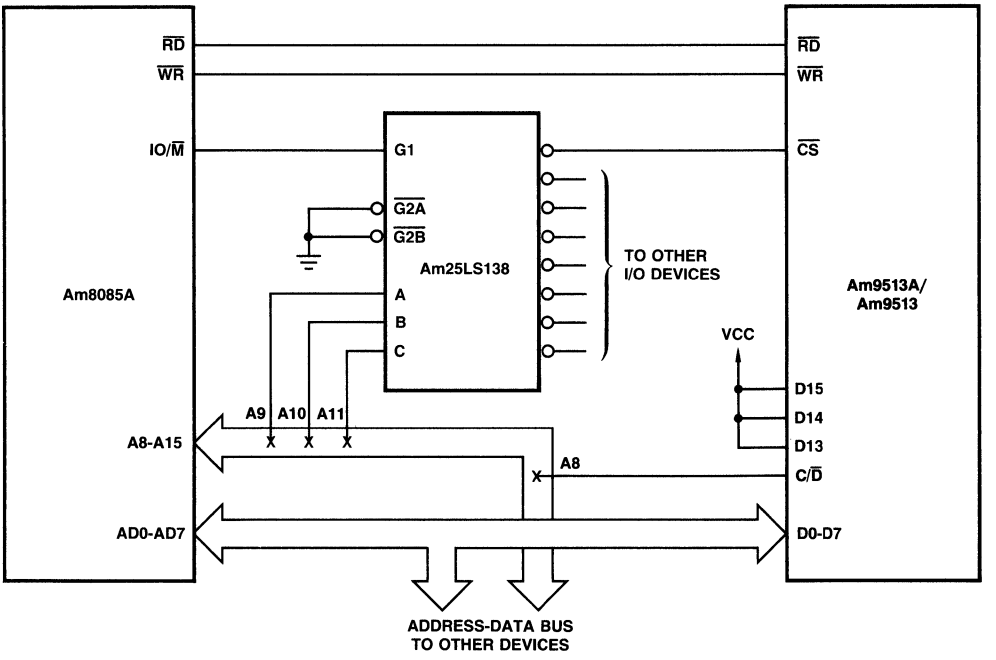


02188A-105

Figure 8-1.2 Am8085A to Am9512-1 Interface

8.1.3 THE 8085 AND Am9513A SYSTEM TIMING CONTROLLER

The Am9513A is designed to interface easily to both the 8-bit and the 16-bit family of CPUs. Master Mode register bit MM13 allows the user to program the Am9513A data bus for either an 8-bit or 16-bit width, allowing the Am9513A's data bus to be tailored to match that of the host CPU. Figure 8-1.3 shows an interface between the Am9513A and an Am8085A CPU. The Am9513A is configured to appear in the CPU's I/O space; connecting the IO/$\overline{M}$ output of the CPU to the $\overline{G2A}$ input of the decoder and tying G1 high will memory-map the Am9513A. The Am9513A operates with an 8-bit data bus. Master Mode register bit MM13 should be 0 and data bus pins DB13-DB15 should be tied high.

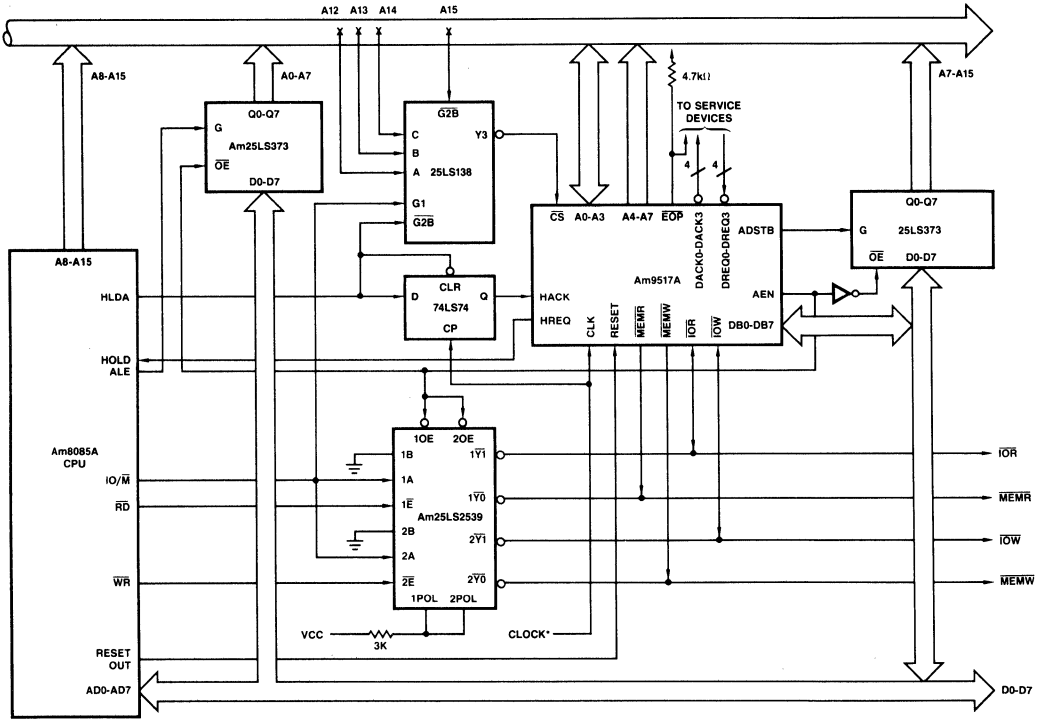


02188A-106

Figure 8-1.3

8.1.4 THE 8085A AND Am9517A INTERFACE

This interface incorporates the Am9517A DMA into the 8085 microprocessor system. The multimode DMA controller issues a Hold Request whenever there is at least one valid DMA Request from a peripheral device. When the processor replies with a Hold Acknowledge signal, the Am9517A takes control of the address, data, and control buses. The 25LS2539 dual one-of-four decoder with three-state outputs easily decodes the I/O and MEM space control lines from the processor, and also isolates the 8085A processor when DMA operations are in process. The address for the first transfer operation comes out of the DMA controller in two bytes, the least-significant eight bits on the eight address outputs and the most significant eight bits on the data bus. The contents of the data bus are conveniently latched into the Am25LS373 register to complete the full 16 bits of the address bus. After the initial transfer takes place, the register is updated only after a carry or borrow is generated in the least-significant address byte. Figure 8-1.4 shows the block diagram for this particular interface.



02188A-107

Figure 8-1.4 Am9517A to Am8085A Interface

8.1.5 8085 TO Am9518 DATA CIPHERING PROCESSOR INTERFACE

Figure 8-1.5 shows the interface diagram between the 8085 microprocessor and the Am9518 Data Encryption device. The DCP and the CPU operate synchronously at a maximum clock rate of 2.2 MHz, considerably simplifying the interface requirements. The 8-bit address/data bus of the CPU is directly connected to the Master Port of the DCP. The Master Port Data Strobe is driven by $\overline{RD}$ or $\overline{WR}$. The MR/$\overline{W}$ input of the DCP is connected to the status line S1 of the 8085. This line is High whenever the CPU executes a read instruction. The Master Port Address Strobe ($\overline{MAS}$) is the inverted Address Latch Enable (ALE). A decoded address and M/$\overline{IO}$=Low produces an active Low Master Port Chip Select. It is latched by $\overline{MAS}$. The DCP can operate with the inverted CPU clock, if the clock is slowed down to satisfy the minimum High time requirement of the DCP. The 8085A data sheet gives a formula to determine the minimum clock High and Low times for slower clocks.

Minimum High time: 0.5 * T - 8 ns (T=clock cycle width)

The minimum High time must be at least 150 ns for an Am9518 and 115 ns for an AmZ8068, resulting in a maximum clock rate of 2.2 MHz and 2.5 MHz respectively.

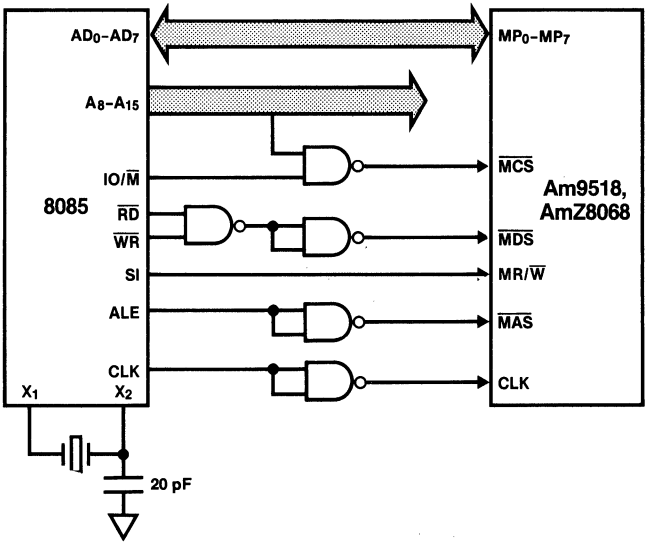
Minimum Low time: 0.5 * T - 4 ns

The minimum Low time is 190 ns at 2.2 MHz.

The DCP requires that the $\overline{MDS}$ is synchronous to the clock. The range is 0 - TWL - 100 ns for the Am9518. TWL is the real Low time of the clock.

Since the 8085 timing specification does not specify a timing relationship between the clock and $\overline{RD}$ or $\overline{WR}$, it is up to the designer to do the verification.

A more sophisticated interface can avoid missing the timing specification and allows interfacing to a faster CPU. Ideas can be found in the iSBX Bus Interface or 68000 Interface. The first shows a totally asynchronous operation of the DCP and the CPU; the second shows how to delay the rising edge of the clock following $\overline{MDS}$.



02188A-108

Figure 8-1.5 8085-DCP Interface

8.1.6 THE 8085 AND Am9519A INTERRUPT CONTROLLER INTERFACE

The Am9519A is a Universal Interrupt Controller, a processor support device designed to enhance the interrupt handling capability of a wide variety of processors. A single Am9519A can manage the masking, priority resolution and vectoring of up to eight interrupts, and it may be easily expanded by the addition of other Am9519A chips to handle a nearly unlimited set of interrupt inputs. It offers many programmable operating options to improve both the efficiency and versatility of the host system operations. The diagram in Figure 8-1.6 shows the interface between the Am8085 and the Am9519A. The two D-type flip-flops in conjunction with S0, S1 and IO/$\overline{M}$ insert a Wait State to the Am8085A whenever the microprocessor acknowledges an interrupt. The circuitry is required since the Am8085A samples a READY (WAIT) signal on the rising edge of CLK OUT during T2, and READY requires a 100 ns minimum set up time prior to the rising edge of T2. The 3-input NAND gate decodes an early $\overline{INTA}$ active signal using the two status outputs (S0,S1) and the IO/$\overline{M}$ pin. This is required since the $\overline{INTA}$ output from the Am8085A becomes valid too late to meet the 100 ns setup time of READY. The Am9519A's $\overline{PAUSE}$ output takes over control of the READY line following T2. Insertion of a Wait State during the read and write operation to the Am9519A with $\overline{CS}$ is required only when the 5MHz Am8085A-2 part is used. The Wait State is needed because the minimum $\overline{RD}$ and $\overline{WR}$ pulse width of the Am8085A-2 is 230 ns, while the Am9519A-1, as example, requires 250 ns minimum and the Am9519A, 300 ns minimum.

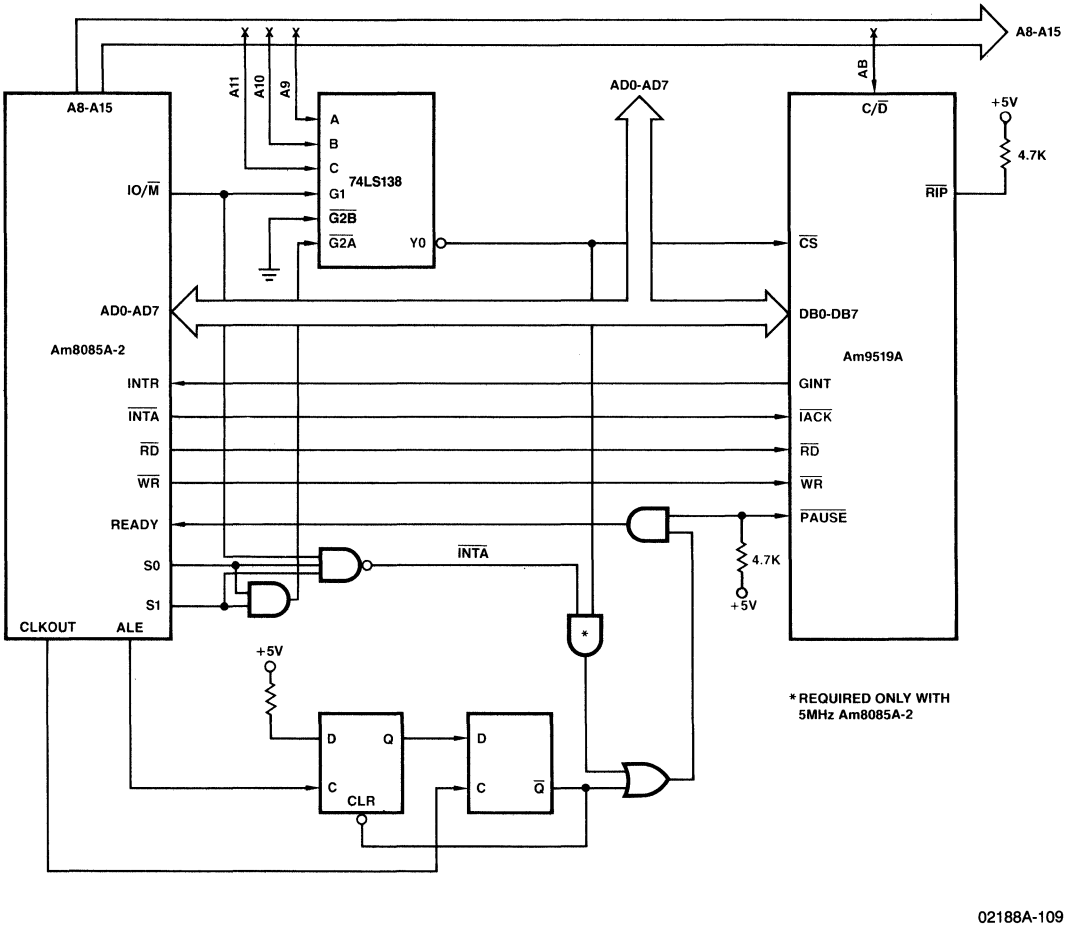


Figure 8-1.6 8085 to Am9519A Interface

9.0 INTERFACING TO THE MC6809

9-1 OVERVIEW OF THE MC6809

This 8-bit microprocessor, with five internal 16-bit registers, is compatible with all 6800 bus-oriented supplementary circuits and peripherals. It is upward compatible with existing 6800 software and can work with a number of peripherals to enhance system designs. The MC6809E uses external clocking to provide the flexibility required in a multi-processor system. Three of the AMD peripherals are introduced here to interface with this microprocessor.

9-1.1 THE MC6809 AND Am9511A FLOATING POINT PROCESSOR

Figure 9-1.1a shows the interface between the MC6809 and the Am9511A. The MC6809 uses memory mapped I/O, therefore, the Chip Select input of the Am9511A needs to be derived from decoding address lines A_1 to A_{15}. The C/$\overline{D}$ input of the Am9511A is generated from the address line A_0 so that an even address selects the data port and an odd address selects the command or status port. The Am9511A requires a 60 ns Hold Time for $\overline{CS}$ or C/$\overline{D}$ after $\overline{WR}$ goes High (inactive); this is accomplished by using the Am25LS2519. The R/$\overline{W}$ output of the MC6809 is decoded by the complementary latch outputs and the $\overline{RD}$ and $\overline{WR}$ pulses are generated by gating these complementary outputs with the MC6809 timing signal E. The Am2956 octal latch is necessary to meet the Data Hold Time requirement during writes to the Am9511A. An external timing source is connected to the Am9511A CLK input since this clock input can be asynchronous to the $\overline{RD}$ and $\overline{WR}$ control signals from the MC6809.

9-1.2 MC6809 TO Am9512 FLOATING POINT PROCESSOR

The Am9512 Floating Point Processor can be added to a MC6809 based system to give throughput increase of an order of magnitude. Due to the peculiar timing characteristics of the MC6809 some additional interface circuitry is necessary. Figure 9-1.2a shows a circuit implementing this interface. Figure 9-1.2b is a timing diagram of the signals involved.

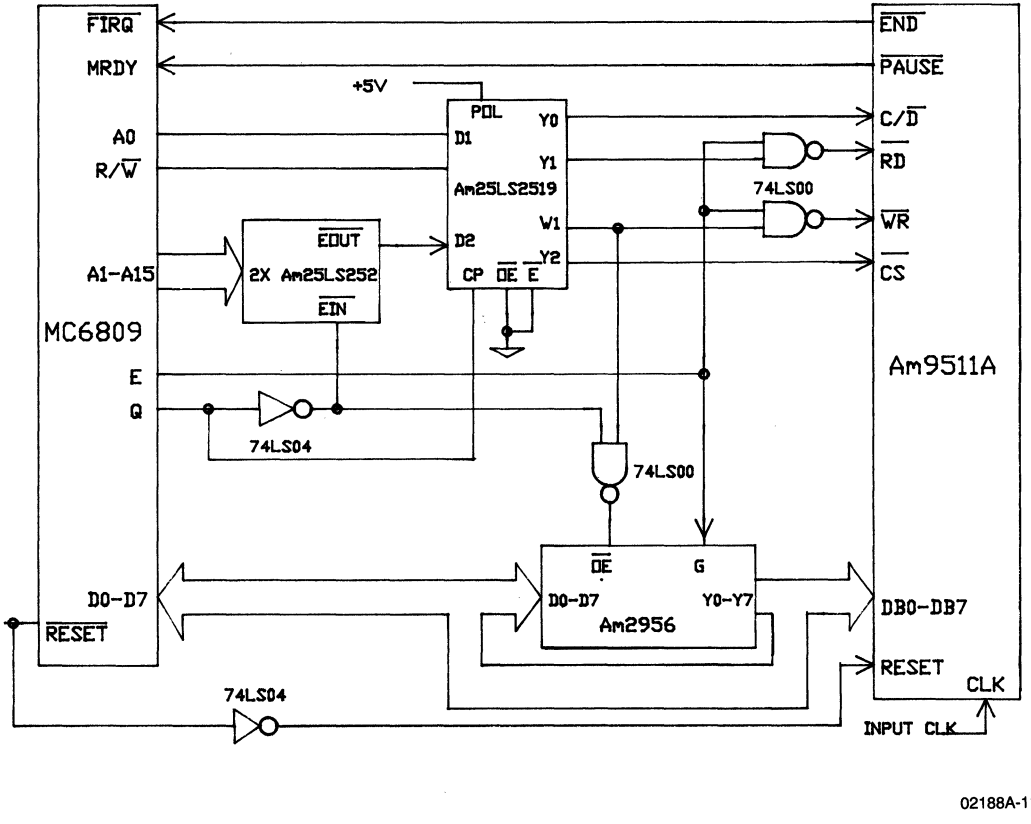
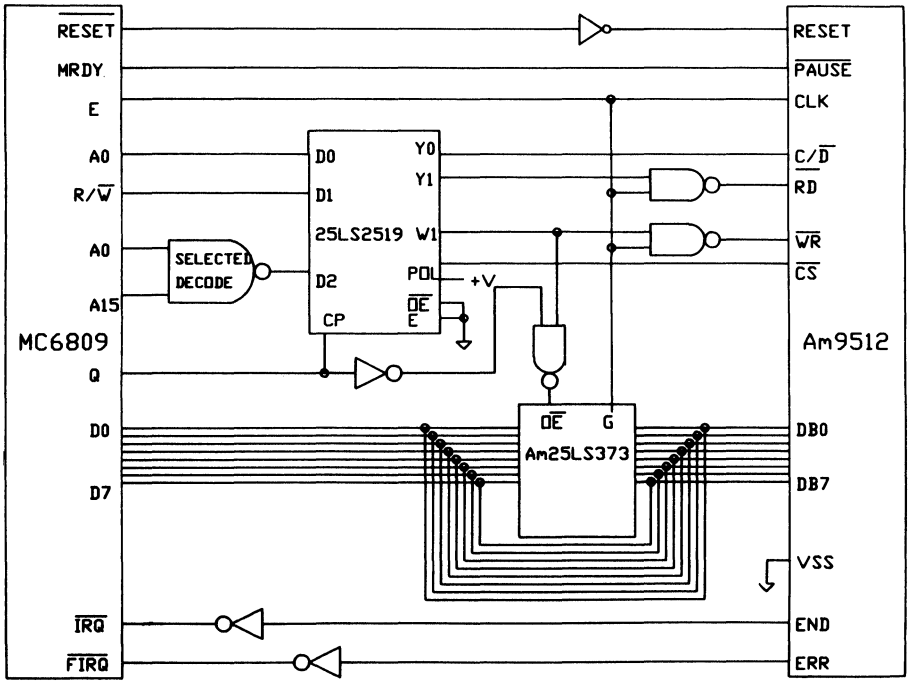
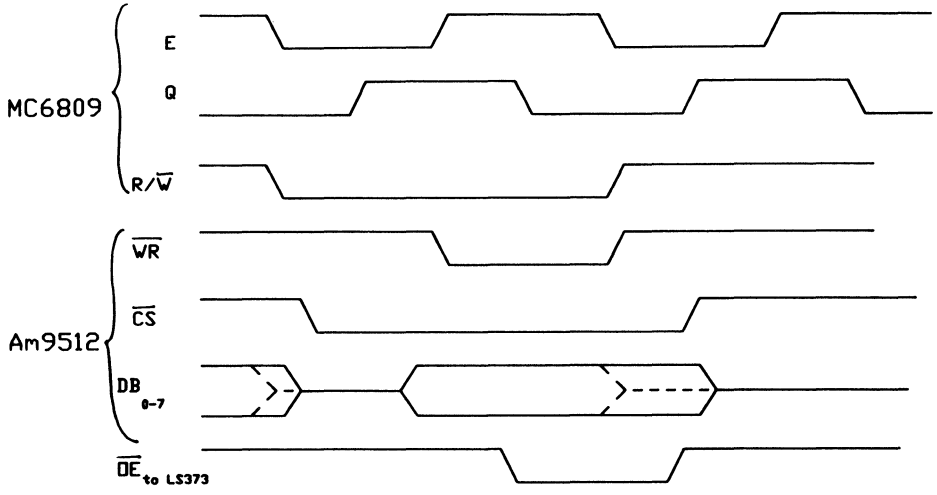


Figure 9-1.1a MC6809 and Am9511A Interface



02188A-111

Figure 9-1.2a MC6809 to Am9512 Interface



Note: Dotted line is data from MC6809 without help from LS373

02188A-112

Figure 9-1.2b MC6809 to Am9512 Write Cycle Timing Diagram

— $\overline{WR}$ and $\overline{RD}$ pulses are generated from R/$\overline{W}$ status line gated with timing signal E. R/$\overline{W}$ is decoded by the complementary latch outputs.

— The 25LS2519 is necessary to hold $\overline{CS}$, C/$\overline{D}$ (and $\overline{WR}$ to LS373) after $\overline{WR}$ (at least 60 ns required).

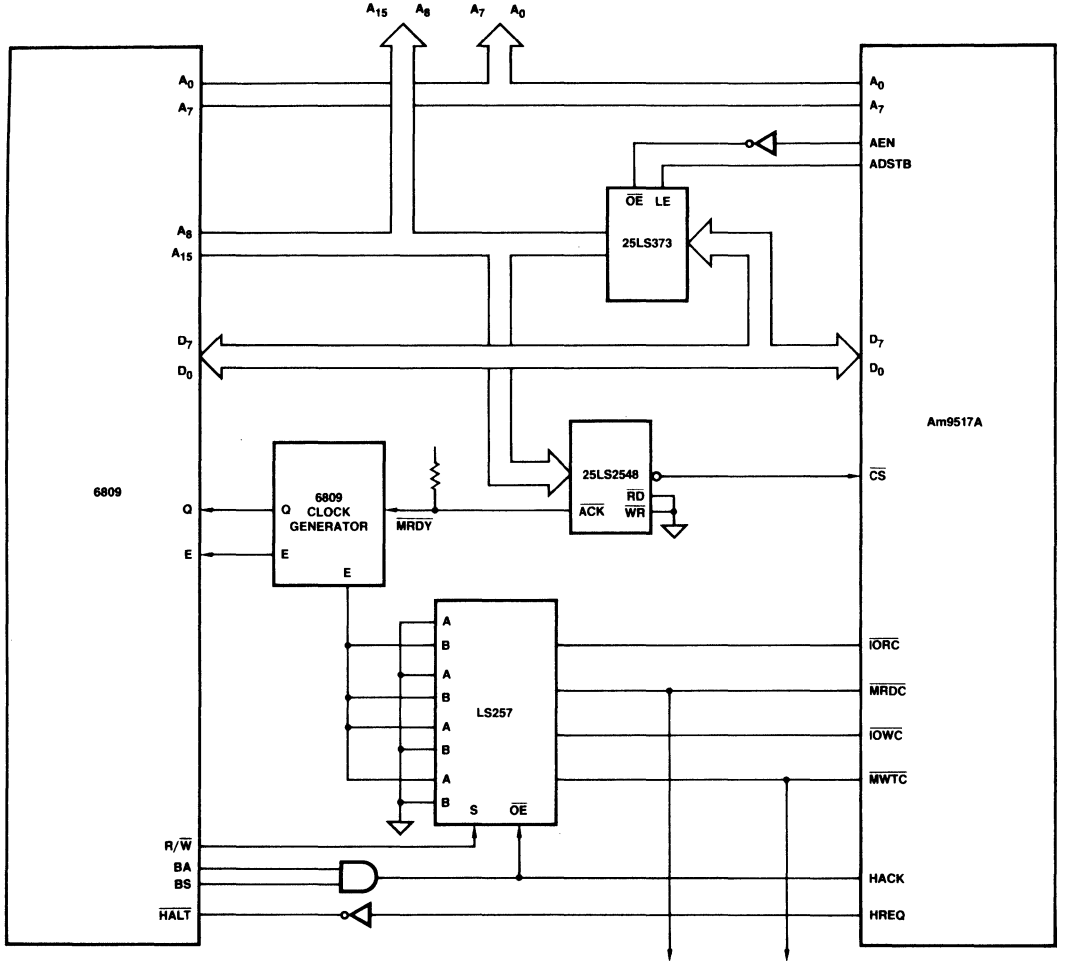
(Care should be taken to ensure $\overline{CS}$ decode meets setup time requirements)

— The 25LS373 is necessary to meet Data Hold Time requirements during writes to the 9512.

At the start of a Write cycle, data is available immediately to the Am9512 because the latch is bypassed and the latch outputs disabled. After the data is valid the latched outputs are enabled, thereby holding the written data after the CPU releases the data bus. On Read cycles, the 25LS373 is bypassed and disabled.

9-1.3 THE MC6809 AND Am9517A DMA CONTROLLER

This interface illustrates the connection of the 9517A DMA controller to the MC6809 processor, Figure 9-1.3a. Many designs do not use multiplexers because they generate glitches when select changes. This particular design is an exception; no glitches can occur because E is always low when R/W switches. If a 68A09 or 68B09 is used, a 9517A-4 should also be used.



02188A-113

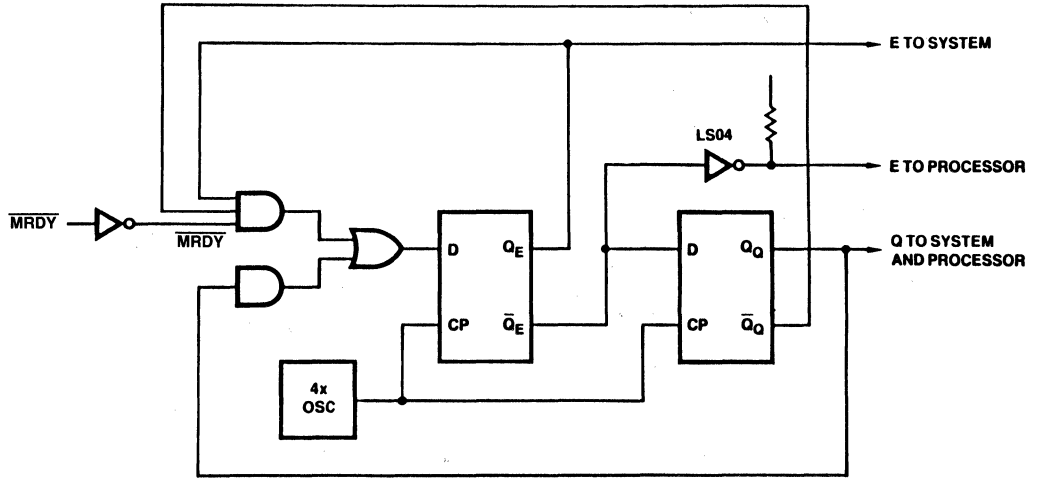
Figure 9-1.3a

When the 9517A is a slave, it ignores $\overline{MRDC}$ and $\overline{MWTC}$, and it does not matter that this circuit generates $\overline{IORC}$ AND $\overline{MRDC}$ simultaneously. Because the MC6809 I/O is memory-mapped, some interesting results can be achieved. For example, one block of memory may be connected to $\overline{IORC}$ and $\overline{IOWC}$ while another block is connected to $\overline{MRDC}$ and $\overline{MWTC}$. With some external hardware, this allows flyby transfers from memory-to-memory, while peripheral-to-memory and memory-to-peripheral are still possible.

The input control signal, $\overline{MRDY}$, shown in the MC6809 data sheet, allows stretching of E and Q to extend data access time. E and Q operate normally while $\overline{MRDY}$ is high. And $\overline{MRDY}$ must always be high, except when E is high. E and Q may be stretched in integral multiples of quarter (1/4) bus cycles, in order to allow interfacing to slower memories.

The clock design illustrated in Figure 9-1.3b makes $\overline{MRDY}$ an active low signal, and is sampled at one 4x clock before the end of E. This allows various devices to be wire-OR'ed to control $\overline{MRDY}$ and, by making $\overline{MRDY}$ a "don't care" during other parts of the cycle, makes generating $\overline{MRDY}$ easier. Figure 9-1.3c illustrates the timing relationships between E, Q and $\overline{MRDY}$.

In many systems, $\overline{MRDY}$ will be generated from the system clock; otherwise, a synchronizing flip-flop is added to prevent metastable states.



02188A-114

Figure 9-1.3b

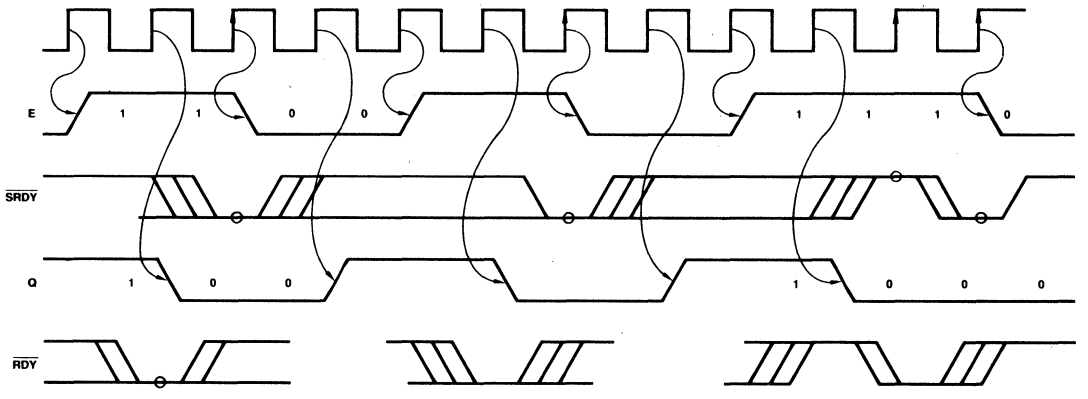
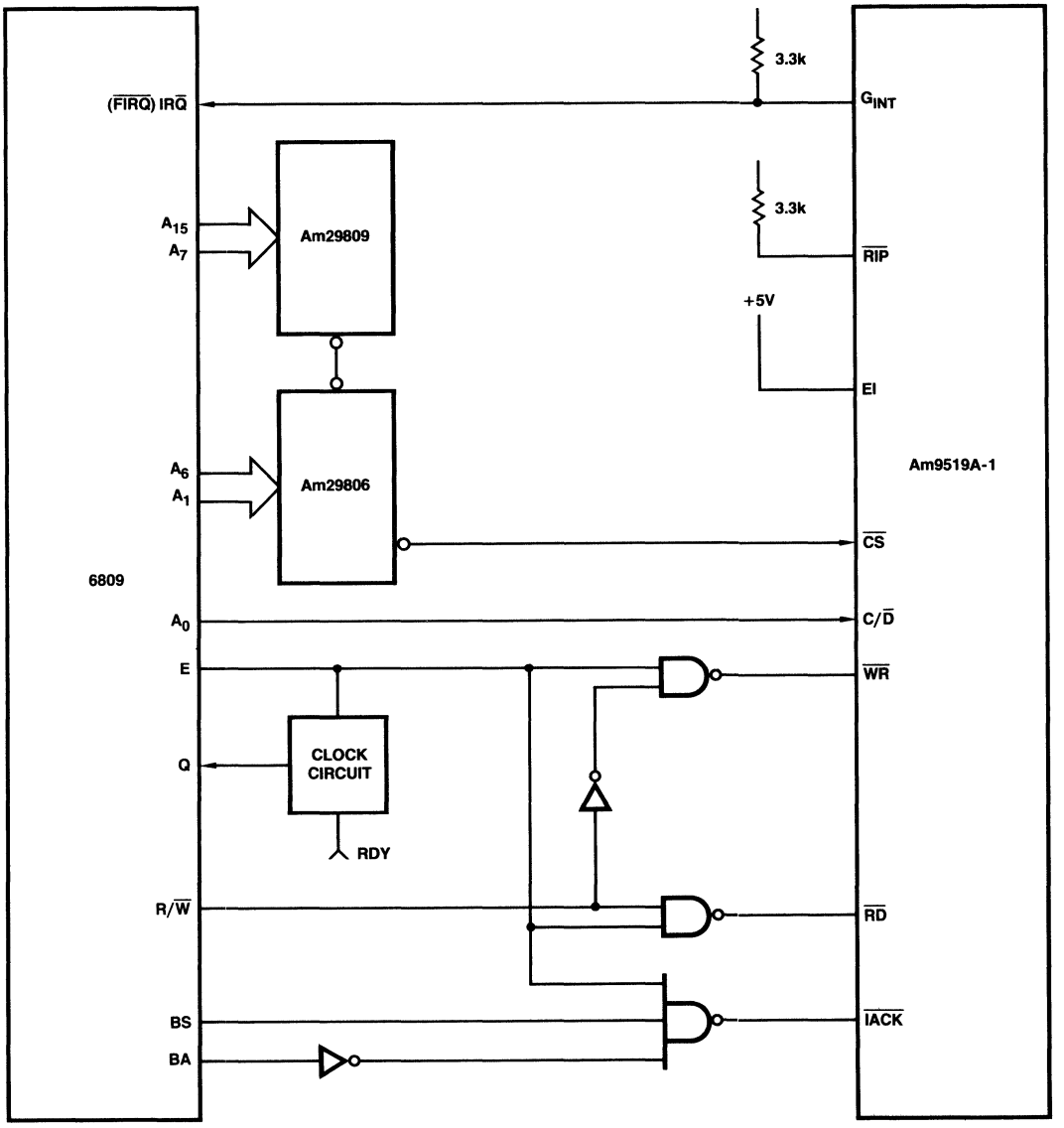


Figure 9-1.3c

02188A-115

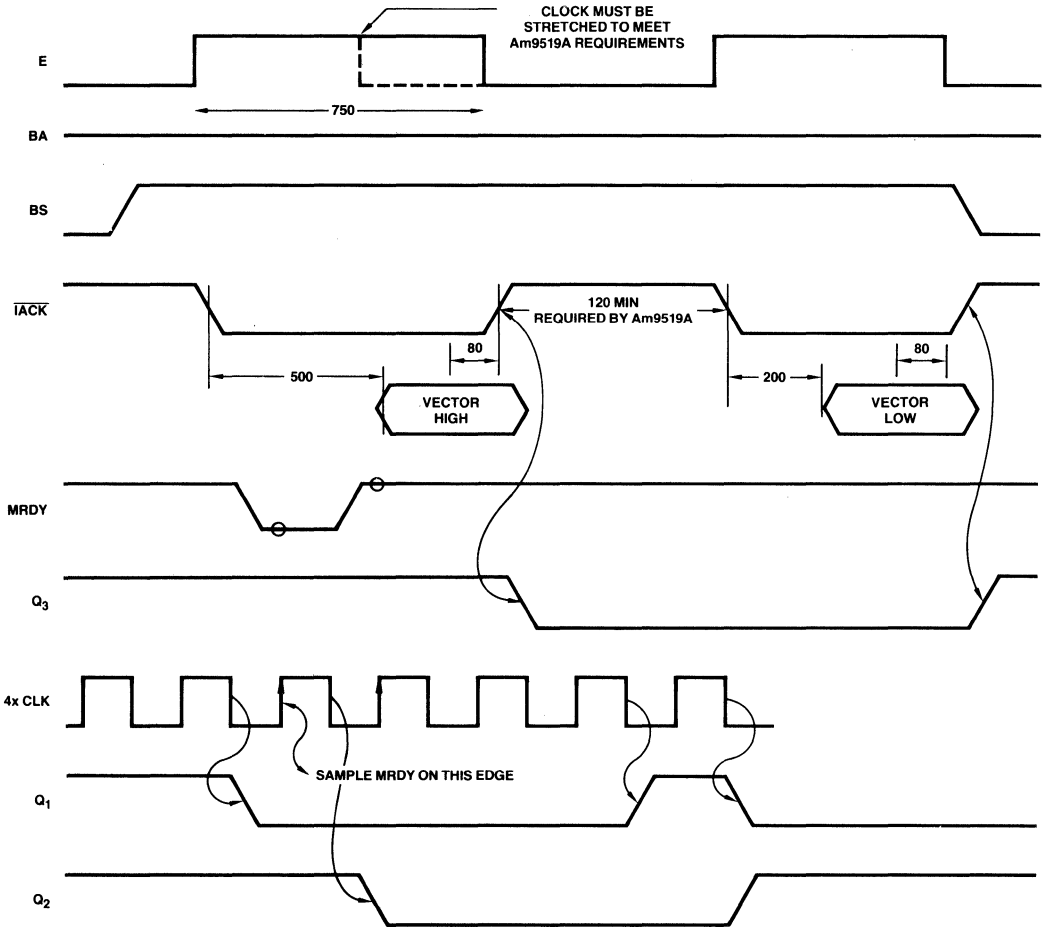
9-1.4 THE MC6809 AND Am9519A INTERRUPT CONTROLLER

The Am9519A is an Universal Interrupt Controller, a processor support device designed to enhance the interrupt handling capability of a wide variety of processors. A single Am9519A can manage the masking, priority resolution and vectoring of up to eight interrupts, and it may be easily expanded by the addition of other Am9519A chips to handle a nearly unlimited set of interrupt inputs. It offers many programmable operating options to improve both the efficiency and versatility of the host system operations. Figure 9-1.4a shows the interface between the MC6809 and the Am9519A; Figure 9-1.4b shows the timing relationships. Figure 9-1.4c shows how to generate MRDY. Q_3 insures MRDY remains High during the second $\overline{IACK}$ pulse.



02188A-116

Figure 9-1.4a



Note: Time in nanosecods.

02188A-117

Figure 9-1.4b

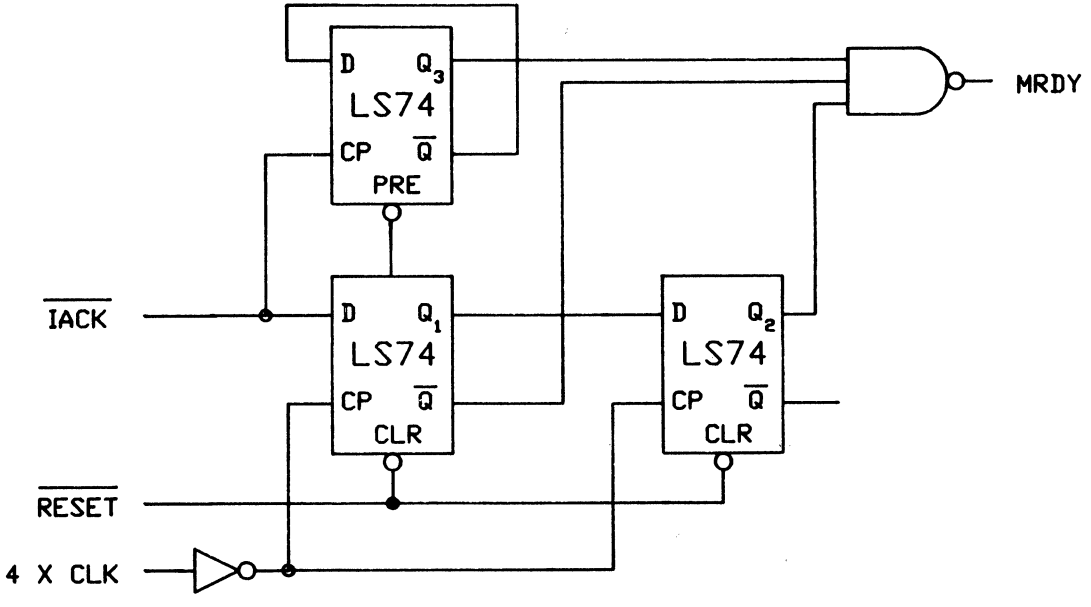

D
Q3
LS74
CP
Q
PRE
MRDY
IACK
D
Q1
LS74
CP
Q
CLR
D
Q2
LS74
CP
Q
CLR
RESET
4 X CLK
02188A-118

Figure 9-1.4c

10.0 INTERFACING TO THE 8051

10.1 OVERVIEW OF THE 8051

The 8051 is an 8-bit microcomputer with an internal 4K x 8 ROM; 128 x 8 RAM; 32 I/O lines; two 16-bit timer/counters; a five-source, two-priority-level, nested interrupt structure; a serial I/O port for either multiprocessor communications, I/O expansion, or full duplex UART; and on-chip oscillator and clock circuits. The 8051 can be expanded using standard TTL compatible memories and the byte oriented 8080 and 8085 peripherals. The 8051 is efficient both as a controller and as a Boolean processor; it has extensive facilities for binary and BCD arithmetic and excels in bit-handling capabilities. The generic term "8051" is used to refer collectively to the 8051, 8751, 9761, i8052, and 8031. The 8031 differs in that it lacks program memory. The 8751 has an EPROM while the 9761 and the i8052 have double the memory.

10.1.1 8051 TO Am9518 DCP INTERFACE

8051 - Am9518/AmZ8068

The 8031/8051/8751 Single-Component 8-Bit Microcomputer family can easily be interfaced to the DCP. Both devices, together with TTL logic, can form a stand-alone data ciphering system for low to medium-speed data communication networks. Clear and ciphered data is handled serially with a programmable handshake protocol.

Figure 10-1.1a shows the 8051-DCP interface. The 8051 must be programmed so that Port 0 provides a multiplexed address/data bus. Port 0 is connected to the Master Port of the DCP.

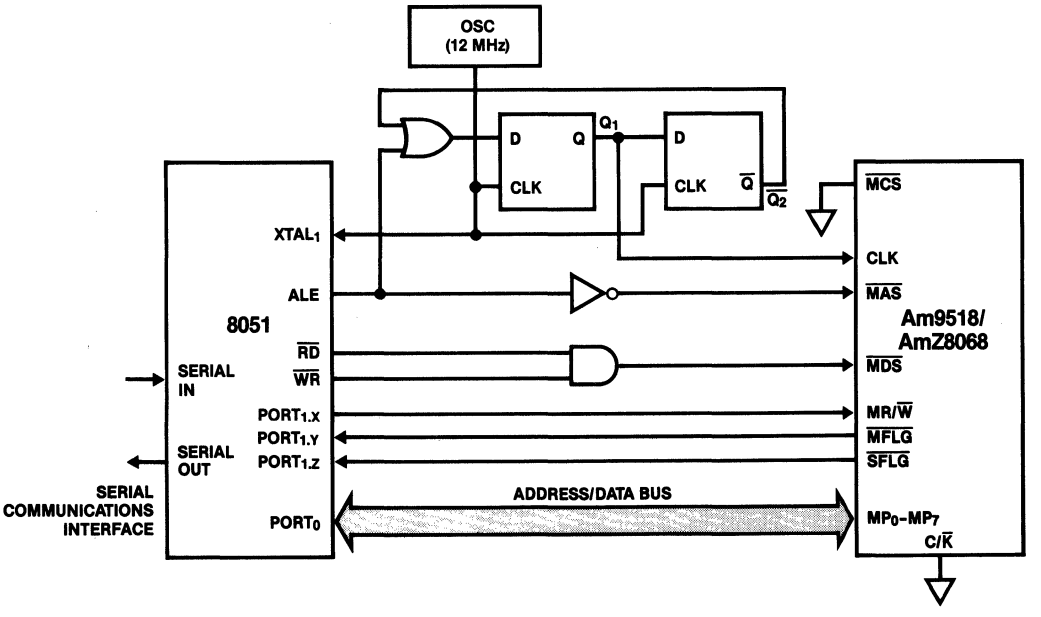
$\overline{RD}$ and $\overline{WR}$ are logically ORed to generate the Master Port Data Strobe. Port 1.X controls the Master Port Read/Write input (MR/$\overline{W}$). This satisfies the set-up time requirement of MR/$\overline{W}$ to $\overline{MDS}$.

Master Port Chip Select can be tied Low if it is guaranteed that $\overline{RD}$ or $\overline{WR}$ only become active in a DCP access cycle. Otherwise, it must be generated by an address decoder.

The 9568 can be used instead of the 9518/8068. The Am9568 eliminates the need of Port 1.X to control Master Port Read/Write. $\overline{RD}$ and $\overline{WR}$ can directly be connected to the corresponding inputs of the DCP ($\overline{MRD}$ and $\overline{MWR}$). ALE does not have to be inverted when connected to MALE.

CLOCK DIVIDER

The DCP clock divider logic as shown in Figure 10-1.1a divides the CPU clock by four or six, depending on the type of instruction the CPU executes (See the timing diagram in Figure 10-1.1b). If the CPU generates an ALE every sixth clock, the CPU clock is divided by six. This is the normal case, and the speed calculation of the DCP should be done for this clock rate. If the



02188A-119

Figure 10-1.1a 8051-DCP Interface

CPU is to execute "MOVX" instructions, every second ALE is left out and the divide factor is four. For both cases the minimum DCP clock High or Low width is two CPU clock periods, which guarantees that even a CPU clock of 12 MHz satisfies the minimum clock requirement for the Am9518 as well as the AmZ8068.

The AmZ8068 gives a wider range for the Data Strobe to $\overline{RD}$ or $\overline{WR}$ delay. The typical value for the 8051 at room temperature with a full load at these outputs is 50 ns. At a CPU clock rate of 10 MHz, this timing requirement is 0-100 ns (two clocks minus 100 ns) for the Am9518 and 0-135 ns (two clocks minus 65 ns) for the AmZ8068.

PROGRAMMING

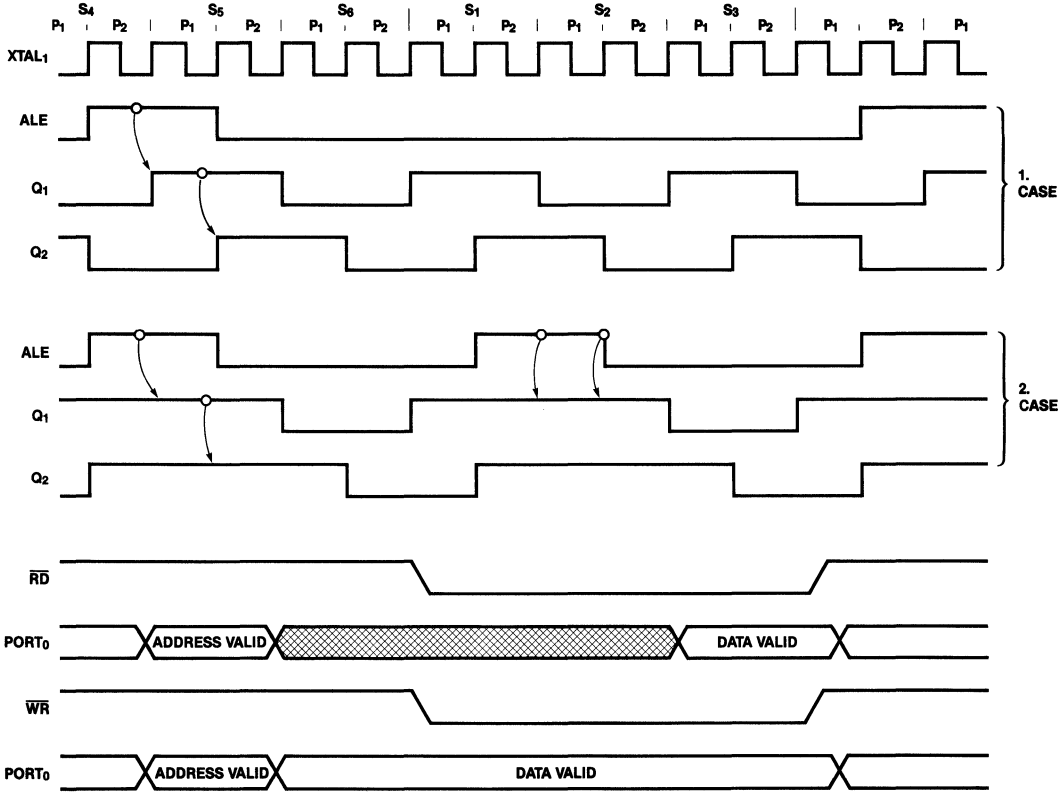
Port 1.X must be High for a Read access and Low for a Write access. Data is transferred using a "MOVX Ri,A" or "MOVX A,Ri" instruction. Ri is register R_0 or R_1. Only this instruction generates the interface timing needed for the DCP. The internal register address is loaded into Rn before executing this instruction.

00 – Data Input or Output Regster

02 – Command or Status Register

06 – Mode Register

The Flags can be monitored by two input pins of the CPU, Port 1.Y and 1.Z. One Flag corresponds to the status of the Input Register, the other one to the status of the Output Register. They become active Low if the CPU can perform a data transfer. In high-speed data ciphering applications, it might be too time consuming to toggle Port 1.X (MR/$\overline{W}$). The toggling can be avoided by choosing the dual port configuration of the DCP. Both the Master and Slave Port are connected to Port 0 of the CPU. During data ciphering session, one port operates as the data input port, the other port operates as the data output port. This means that during the whole session, the data flow direction does not have to be turned around; MR/$\overline{W}$ can stay Low or High for the whole session. $\overline{MCS}$ and $\overline{SCS}$ select the appropriate port.



02188A-120

Figure 10-1.1b 8051-DCP Timing Diagram

11.0 INTERFACING TO THE Z8000

11.1.0 Z8001/8002 OVERVIEW

This CPU has a 16-bit data multiplexed address/data bus (the Z8001 has 7 additional non-multiplexed address outputs, called segment bits, extending its memory address range to 8 Mbyte).

Z8001/8002 addresses are specified as bytes. In a 16-bit word the least significant byte has the higher address, the most significant byte has the lower address. This differs from the 8080, 8085, 8086, and Z80. Moreover, 16-bit and 32-bit words must be aligned with an even address.

The 8-bit peripherals should be connected to the lower half of the Z8001/2 data bus and must be addressed with odd addresses (non-contiguous addressing).

The data bus is "asynchronous", i.e. the CPU machine cycle can be stretched without clock manipulation by inserting Wait States between T2 and T3 of a read or write cycle to accommodate slower memory or peripherals.

Separate address space for I/O (64 kbytes) defined by Status Code 0010 on the four Status outputs and by the R/$\overline{W}$ output from the Z8001/2. Addresses are guaranteed valid before the rising edge of $\overline{AS}$.

Data from I/O must be valid before the rising edge of $\overline{DS}$. Data to I/O is valid during $\overline{DS}$.

DMA: The bus is requested by activating the $\overline{BUSRQ}$ input to the Z8001/2.
Bus Grant is confirmed by the $\overline{BUSAK}$ output from the Z8001/2.

Interrupt is requested by activating either the $\overline{NMI}$ (non-maskable), $\overline{NVI}$ (non-vectored interrupt), $\overline{VI}$ (vectored interrupt), or $\overline{SEGT}$ (segment trap from the memory management unit).

Interrupt is acknowledged by the STATUS code output from the Z8001/2. During the Interrupt Acknowledge Cycle the Z8001/2 reads a 16-bit word containing an 8-bit jump vector for vectored interrupt.

11.1.1 Z8001 TO Am7990 LANCE INTERFACE

Z8001 interface to the LANCE is easily accomplished since the Z8000 also has a multiplexed bus and most of the control signals can directly be connected (Figure 11-1.1). This design also uses the PAL (AmPAL16L8) to reduce parts count. $\overline{INTR}$ pin of the LANCE is connected to $\overline{NVI}$ pin of the Z8001, since LANCE does not return a vector during the interrupt acknowledge cycle. The PAL uses the status lines ST_3-ST_0 (when Z8000 is the bus master), or $\overline{HLDA}$ from LANCE (when LANCE is in bus master mode), to generate $\overline{M}$, the memory request signal. The user should program the ACON, BCON, and BSWP to 1, prior to initializing the LANCE. When the LANCE is the bus master, $\overline{DALI}$ and $\overline{DAL0}$ control the transceiver. When the CPU is the bus master, $\overline{T}$ and $\overline{R}$ are generated from R/$\overline{W}$ and $\overline{DS}$ to control the transceiver.

11.1.2 THE Z8000 AND AmZ8068 DATA CIPHERING PROCESSOR INTERFACE

The DCP is one of the state-of-the-art peripheral devices that works hand in hand with most of the popular microprocessors on the market. At the heart of the DCP is the Data Encryption Standard (DES) algorithm unit that encrypts blocks of clear text into corresponding blocks of cipher text using a 56-bit key. The DCP can hold three keys simultaneously: a Master Key to generate session keys, an Encryption Key, and a Decryption Key. The DCP's Input and Output registers can each transfer data to and from the Master or Slave Port, on the 8-bit input/output buses. The dual ports, with separate internal buses and separate input and output registers, compose a highly pipelined data path that maximizes the throughput by allowing simultaneous input, ciphering, and output operations.

Figure 11-1.2 shows an interface between a 4-MHz Z8001/2 microprocessor and the AmZ8068. The CPU and the DCP can operate synchronously at a clock rate up to 3.5 MHz. All control and strobe signals can be connected directly to the DCP.

The clock rate is reduced to 3.5 MHz to satisfy timing parameter (Clock LOW to $\overline{MDS}$ HIGH). The delay time from clock falling to Data Strobe ($\overline{DS}$) rising is specified at 0 to 70 ns for the Z8000; the DCP requires 0 to 50 ns at 4 MHz. By reducing the clock rate, this parameter becomes 0 to 70 ns at 3.5 MHz.

The system can operate at 4 MHz, if a 10-MHz Z8001/2 is used. This faster version is specified for 0 to 45 ns.

A sample program for testing the DCP is shown in Figure 11-1.2b. The program is written in Z8000 (nonsegmented) assembly language. The DCP must be initialized for multiplexed Control Mode and "Master Port only" configuration. The ciphering mode can be ECB or CBC. The mode is defined by the varible "MODE". A one-cycle operation of the interface is assumed. For a two-cycle operation interface, instructions to latch the register address must be added.

11.1.3 AmZ8002 TO Am9511A FLOATING POINT PROCESSOR INTERFACE

The Am9511A can be interfaced to a 16-bit microprocessor such as the AmZ8002. Since the data bus of the Am9511A is only 8 bits wide, the operations performed must be byte-oriented.

The $\overline{RD}$ and $\overline{WR}$ inputs to the Am9511A can obtained by demultiplexing the data strobe ($\overline{DS}$) output of the AmZ8002. The data bus of the Am9511A can be connected to either the upper 8 bits or the lower 8 bits of the AmZ8002 data bus. If the Am9511A data bus is connected to the upper 8 bits, (Figure 11-1.3a) the I/O address of the Am9511A is always even. If the Am9511A data bus is connected to the low 8 bits, the I/O address is always odd. The chip select is derived from a decode of A_2 to A_{15}. A_1 is used to select between data/status during READ and data/command during WRITE.

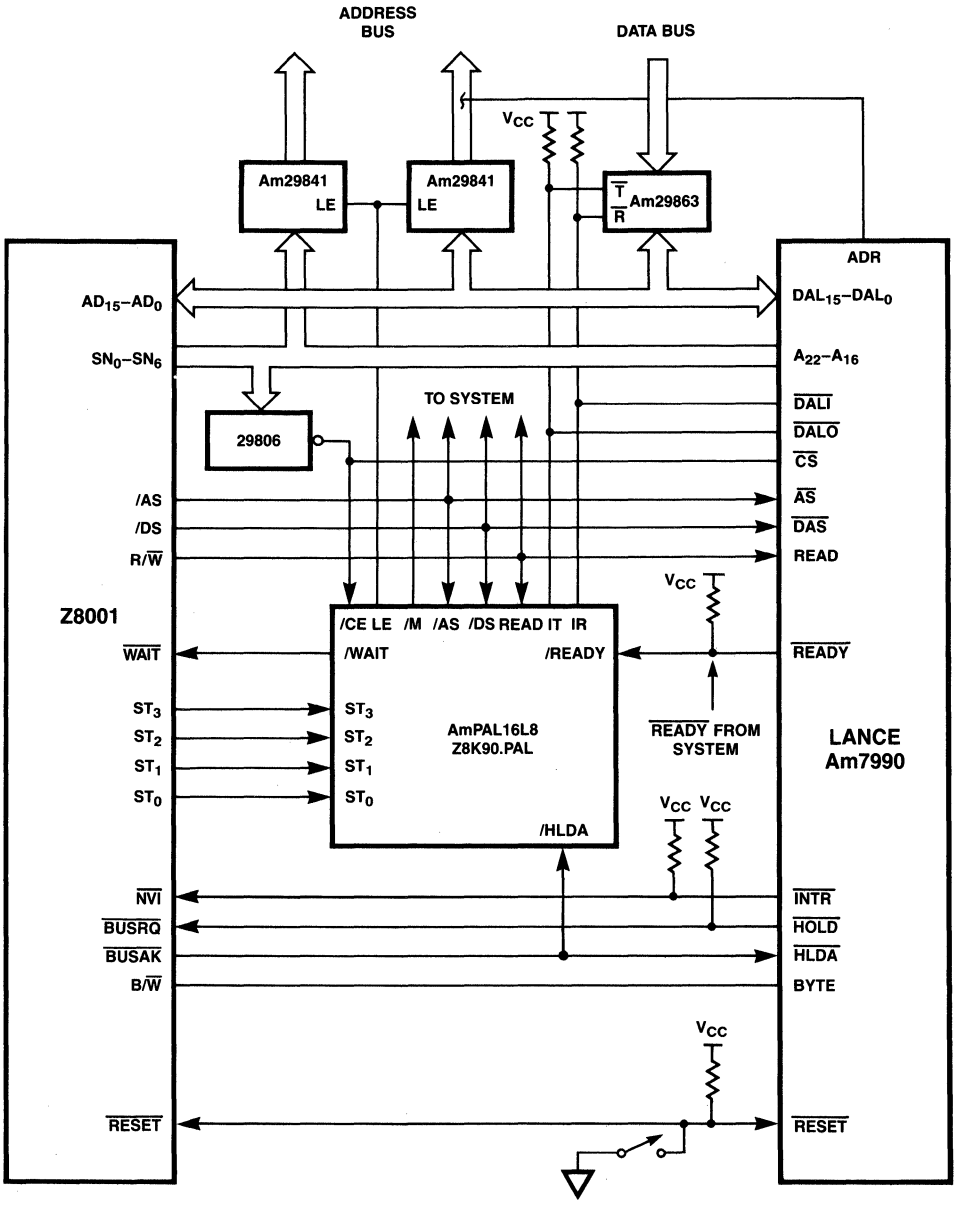
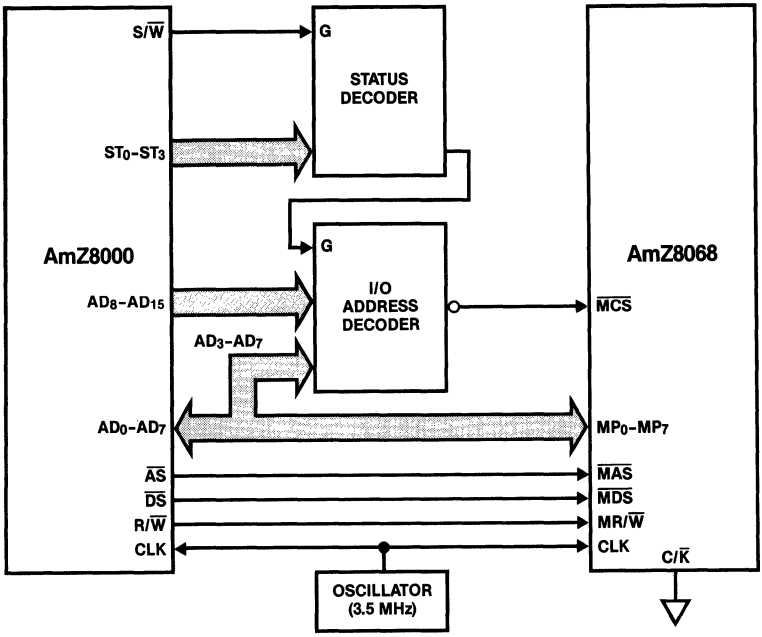

ADDRESS BUS
DATA BUS
Am29841 LE
Am29841 LE
T/R Am29863
VCC
ADR
AD15–AD0
SN0–SN6
DAL15–DAL0
A22–A16
29806
TO SYSTEM
DALI
DALO
CS
/AS
/DS
R/W
AS
DAS
READ
Z8001
/CE LE /M /AS /DS READ IT IR
/WAIT
/READY
WAIT
READY
ST3
ST2
ST1
ST0
AmPAL16L8
Z8K90.PAL
READY FROM SYSTEM
LANCE
Am7990
/HLDA
NVI
BUSRQ
BUSAK
B/W
INTR
HOLD
HLDA
BYTE
RESET
RESET
02188A-121

Figure 11-1.1

```
AMPAL16L8                    RASOUL M. OSKOUY
PAT001                       MARCH 13, 1984
File: Z8K90.PAL

Z8001 TO LANCE INTERFACE
ADVANCED MICRO DEVICES

/AS  /HLDA READ /CS   /DS /READY ST3 ST2 ST1 GND

STO /T   /M    /WAIT LE /R     NC  NC  NC  VCC

IF /(HLDA)   T = /READ*/CS

IF (/HLDA)   R = READ*DS*/CS

             M = /HLDA*ST3*/ST2*/ST1*/STO +

                 /HLDA*/ST2*/ST1*STO +

                 /HLDA*ST3*ST2*/ST1 +

                 HLDA

          WAIT = /READY

           /LE = AS
```

Figure 11-1.1b

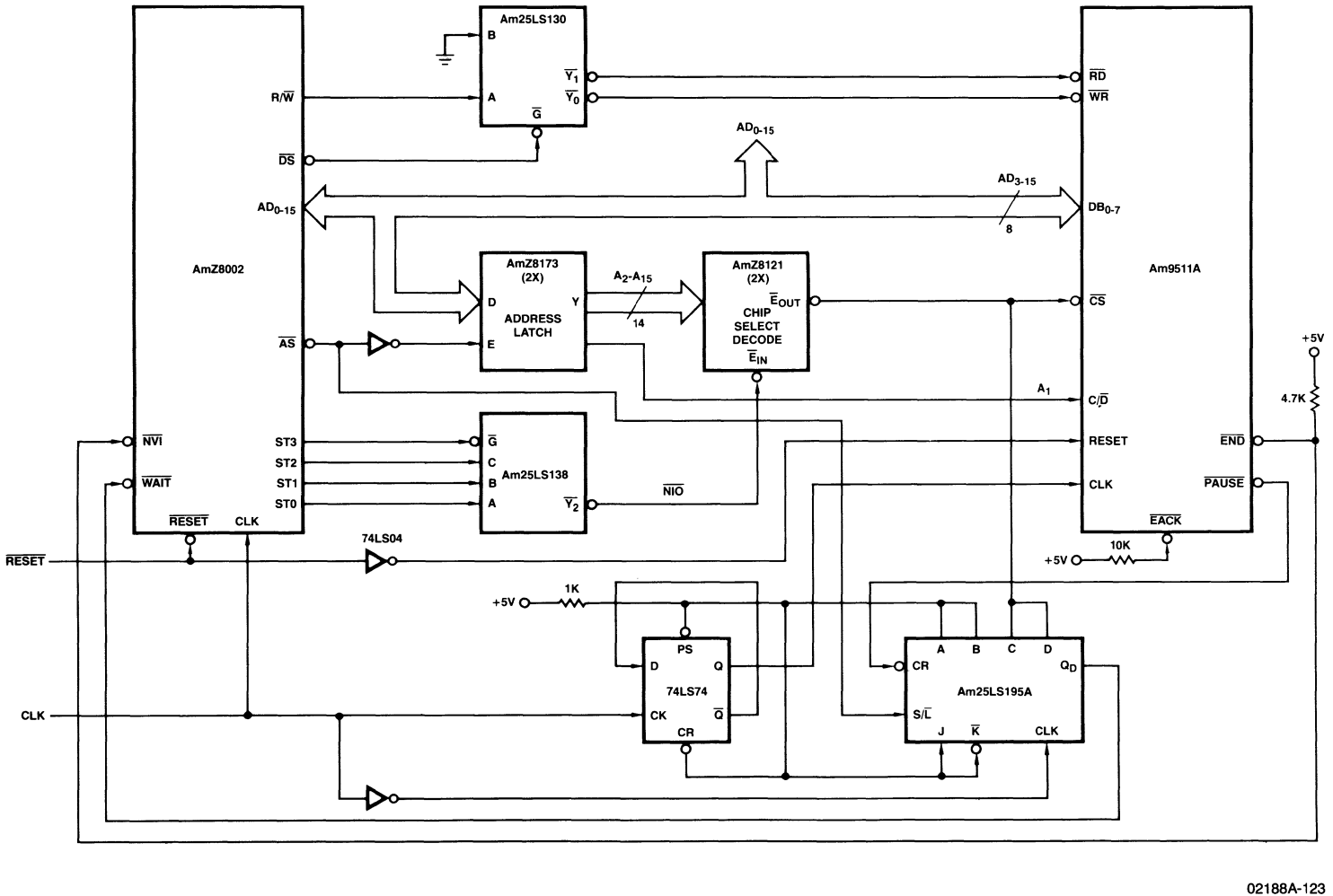


02188A-122

Figure 11-1.2a Z8000-AmZ8068 Interface

```
MACRO8000:        Version 2.0   9/19/80                    Page 1

0000                    %*************************************************************************
0000                    %*                                                              JS   3/12/84 *
0000                    %*        ENCRYPTION EXAMPLE FOR Z8000                                      *
0000                    %*                                                                          *
0000                    %*************************************************************************
0000
0000                         PROGRAM DCP_SHOW;
0000                         ORIGIN  #1000;
1000
1000                    DCP_OUT:        BYTE (32);      % DCP OUTPUT STORAGE AREA
1020                    DCP_IN:         BYTE (32);      % DCP INPUT STORAGE AREA
1040                    CIVE:           BYTE (8);       % CLEAR IV STORAGE FOR CBC/CFB ENCRYPTION
1048                    CE_KEY:         BYTE (8);       % CLEAR ENCRYPTION KEY
1050                    MODE:           BYTE (1);       % MODE VALUE
1051    00
1052                    DATAREG:        WORD (1);       % DATA REGISTER ADDRESS (MASTER PORT)
1054                    CSREG:          WORD (1);       % COMMAND/STATUS REGISTER ADDRESS
1056                    MODEREG:        WORD (1);       % MODE REGISTER ADDRESS
1058
1058                    DCP_SHOW:
1058    6103 1052               LD      R3,DATAREG;     % LOAD DATA REGISTER ADDRESS
105C    6101 1054               LD      R1,CSREG;       % LOAD COMMAND/STATUS REGISTER ADDRESS
1060    6102 1056               LD      R2,MODEREG;     % LOAD MODE REGISTER ADDRESS
1064    600F 1050               LDB     RL7,MODE;       % LOAD MODE VALUE
1068    3E2F                    OUTB    R2,RL7;         % SET MODE (INCLUDES SOFTWARE RESET)
106A
106A                    % LOAD IVE REGISTER
106A    CFA5                    LDB     RL7,#A5;        %IVE LOAD COMMAND
106C    3E1F                    OUTB    R1,RL7;
106E    2108 0008               LD      R8,#8;          % BYTE COUNTER
1072    2109 1040               LD      R9, CIVE;       % ADDRESS OF CLEAR IVE FIELD
1076    3A92 0830               OTIRB   R3,R9 ,R8;      % STROBE 8 BYTE IV IN
107A
107A                    % LOAD E KEY REGISTER
107A    CF11                    LDB     RL7,#11;        % LOAD E KEY COMMAND
107C    3E1F                    OUTB    R1,RL7;
107E    2108 0008               LD      R8,#8;          % BYTE COUNTER
1082    2109 1048               LD      R9, CE_KEY;     % ADDRESS OF CLEAR E KEY FIELD
1086    3A92 0830               OTIRB   R3,R9 ,R8;      % STROBE 8 BYTES KEY IN
108A
108A                    % ENCRYPTION SESSION
108A    CF41                    LDB     RL7,#41;        % START ENCRYPTION COMMAND
108C    3E1F                    OUTB    R1,RL7;
108E    2108 0008               LD      R8,#8;          % BYTE COUNTER
1092    2109 1020               LD      R9, DCP_IN;     % DATA INPUT FIELD
1096    3A92 0830               OTIRB   R3,R9 ,R8;      % TRANSFER FIRST BLOCK
109A    2108 0008               LD      R8,#8;          % BYTE COUNTER
109E    3A92 0830               OTIRB   R3,R9 ,R8;      % TRANSFER SECOND BLOCK
10A2    2108 0008               LD      R8,#8;          % BYTE COUNTER
10A6    210A 1000               LD      R10, DCP_OUT;   % DATA OUTPUT FIELD
10AA    3A30 08A0               INIRB   R10 ,R3,R8;     % READ FIRST CIPHERED BLOCK BACK
10AE    2108 0008               LD      R8,#8;          % BYTE COUNTER
10B2    3A92 0830               OTIRB   R3,R9 ,R8;      % TRANSFER THIRD BLOCK
10B6    2108 0008               LD      R8,#8;          % BYTE COUNTER
10BA    3A30 08A0               INIRB   R10 ,R3,R8;     % READ SECOND CIPHERED BLOCK BACK
10BE    2108 0008               LD      R8,#8;          % BYTE COUNTER
10C2    3A92 0830               OTIRB   R3,R9 ,R8;      % TRANSFER FOURTH BLOCK
10C6    2108 0008               LD      R8,#8;          % BYTE COUNTER
10CA    3A30 08A0               INIRB   R10 ,R3,R8;     % READ THIRD CIPHERED BLOCK BACK
10CE    2108 0008               LD      R8,#8;          % BYTE COUNTER
10D2    3A30 08A0               INIRB   R10 ,R3,R8;     % READ FOURTH CIPHERED BLOCK BACK
10D6
10D6                    % TERMINATE CIPHERING SESSION
10D6    CFE0                    LDB     RL7,#E0;        % LOAD STOP COMMAND
10D8    3E1F                    OUTB    R1,RL7;         % ISSUE STOP COMMAND
10DA
10DA                            END.
```

Figure 11-1.2b



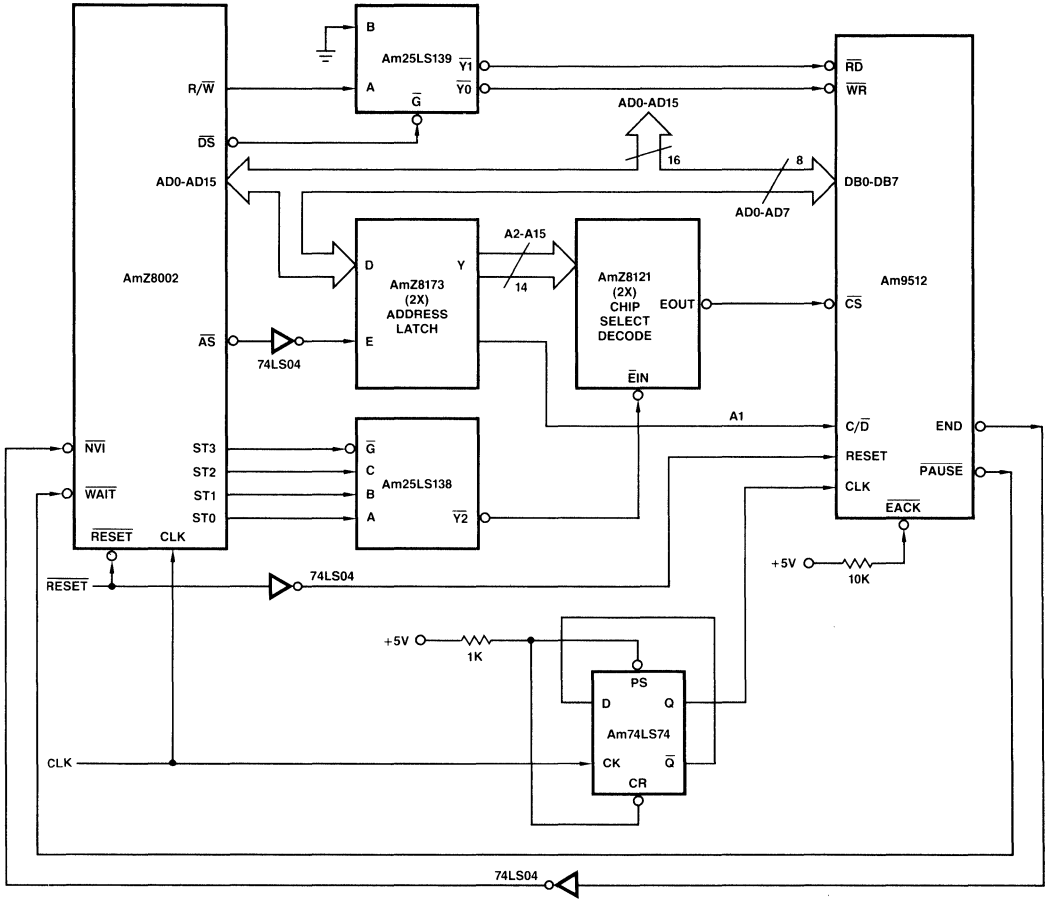
02188A-123

Figure 11-1.3a AmZ8002 to Am9511A Interface

Due to the long READ access time of the Am9511A, the AmZ8002 must be put in a WAIT state for each READ access to the Am9511A. If the PAUSE output of the Am9511A is connected directly to the WAIT input of the AmZ8002, the PAUSE output will arrive too late to put the AmZ8002 into the WAIT state. The Am25LS195A 4-bit shift register is used to solve this problem. During each address strobe, the Q_D output will be forced LOW if chip select to the Am9511A is present. The Q_D will remain LOW for two clock periods. If PAUSE is LOW during this period, the WAIT line will remain LOW because the Am25LS195A is held at the reset state. After the PAUSE returns to HIGH, the Q_D output will go HIGH after two clocks and the AmZ8002 can proceed with the current operation.

11.1.4 AmZ8002 TO Am9512 FLOATING POINT PROCESSOR INTERFACE

The AmZ8002 to Am9512 interface is similar to the AmZ8002 to Am9511A interface, except the PAUSE output of the Am9512 can be connected directly to the WAIT input of the AmZ8002. This is because the PAUSE output of the Am9512 follows the chip select instead of RD and WR and the AmZ8002 has sufficient time to go into the WAIT state. Figure 11-1.4 shows the interface block diagram.



02188A-124

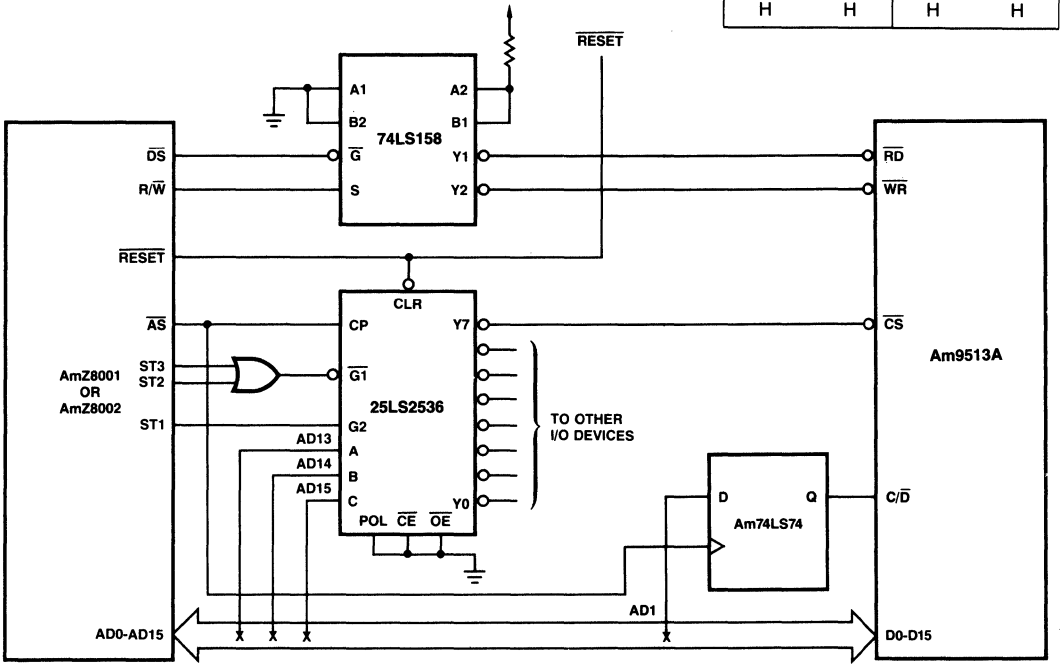
Figure 11-1.4 AmZ8002 to Am9512 Interface

11.1.5 THE Z8000 AND Am9513A SYSTEM TIMING CONTROLLER INTERFACE

In this interface the Am9513A is connected with an AmZ8001 or AmZ8002 CPU (Figure 11-1.5a). The Am9513A appears in both Regular and Special I/O space, by virtue of the decoding of status lines ST1-ST3. Status line ST0 should be decoded also if it is necessary to separate the Regular and Special I/O spaces. The Am25LS2536 is a latched decoder which stores the address information on the rising edge of $\overline{AS}$, providing the Am9513A with a stable $\overline{CS}$ for the duration of the transfer. The Am74LS158 multiplexer generates $\overline{RD}$ and $\overline{WR}$ from the CPU's $\overline{DS}$ and R/$\overline{W}$ lines. For maximum data bandwidth between the CPU and the Am9513A, Master Mode register bit MM13 should be set to 1 to configure the Am9513A for a 16-bit data bus width. This can be accomplished by writing command opcode FFEF(hex) to the Am9513A following each reset and power up.

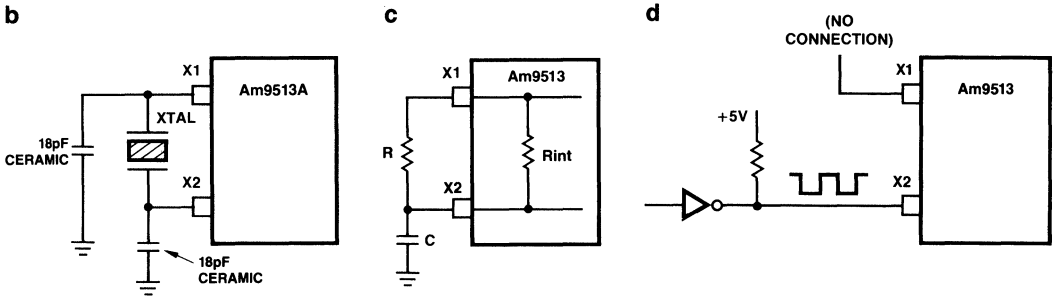
Figures 11-1.5b, c, and d show various ways to drive the X1 and X2 inputs. Figure 11-1.5d shows the Am9513A internal oscillator being driven by an external signal. The Am9513A Electrical Specification should be consulted for the voltage levels required on the X2 input to guarantee proper oscillator operation in this configuration.

AmZ8001/2		Am9513	
$\overline{DS}$	R/$\overline{W}$	$\overline{WR}$	$\overline{RD}$
L	L	L	H
L	H	H	L
H	L	H	H
H	H	H	H



02188A-125

Figure 11-1.5a AmZ8001/8002–Am9513 Interface



02188A-126

*Note: The Am9513A oscillator was changed from the Am9513. The capacitor values in previous designs should be changed to the values shown.

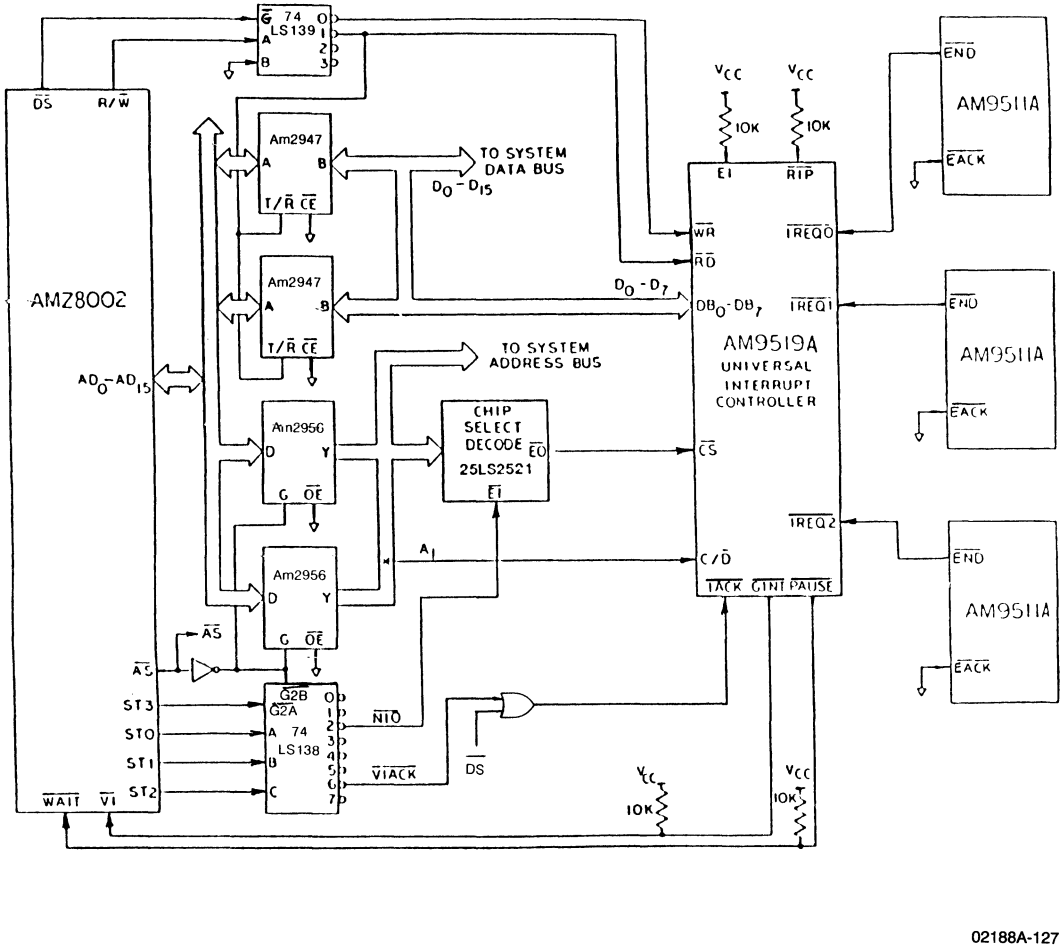
Figure 11-1.5b, c, d Driving the X1 and X2 Inputs

11.1.6 THE Z8002 AND Am9519A INTERFACE

The Am9519A Interrupt Controller is used in a system to relieve the CPU from the task of determining the sources of interrupts. It also serves to prioritize multiple interrupts. The interface block diagram is shown in Figure 11-1.6a. The Am2956 latches and Am8304 bidirectional buffers are used to generate separate address and data buses from the CPU address/data bus. The latched address is checked by two 25LS2521 8-bit comparators which generate a Chip Select ($\overline{CS}$) for the Am9519A. The addresses used here are 0021H or 0023H in normal I/O space. Latched address bit LADDR1 is not included in the comparison, but drives the C/$\overline{D}$ input to the Am9519A. This defines address 0021H as being the address for data transactions and address 0023H as being the address for command/status transaction. Note the I/O addresses are odd, since the 9519A is on the lower data bus.

The lower half of the buffered data bus drives the bidirectional data bus of the Am9519A. The read and write commands required by the Am9519A are generated with half of a 74LS139 decoder. $\overline{DS}$ is applied 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. The Am9519A makes a vectored interrupt request to the CPU using the Group Interrupt line ($\overline{GINT}$) which should be a LOW active output from the AM9519A. $\overline{GINT}$ 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.

The Am9519A 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 Am9519A 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 interrupt inputs to the Am9519A from the devices may be levels or pulses. If levels are employed, some form of request reset will have to be included. In the block diagram, pulsed interrupt requests are assumed and should be connected to the $\overline{IREQ}$ inputs of the Am9519A, which should be programmed for Low Active interrupt requests.



02188A-127

Figure 11-1.6a AmZ8000 to Am9519A Interface

SECTION D
BUS INTERFACING

12.0 MISCELLANEOUS

12.1 MULTIBUS * TO Am9516 INTERFACE

This interface shows the Am9516 connected to the MULTIBUS * (Figure 12-1a). This is accomplished by two PAL device designs. The equations for the PAL devices are shown in Figures 12-1b and 12-1c. The first designated Am9516MBC does the MULTIBUS* arbitration as defined by the MULTIBUS* specification. Common bus request ($\overline{CBRQ}$) was not implemented in this design. Additionally this design holds the bus as long as BREQ is active. If the user wishes to release the bus after each transaction the Am9516 should be programmed for CPU interleave.

The second PAL device in this interface designated 9516MBC, converts the Am9516 signals into MULTIBUS * control signals. It also generates $\overline{RD}$ and $\overline{WR}$ for the 8530 so that flyby transfers can be done. When not doing flyby, this part of the PAL device should be changed. There are several considerations when using the SCC with DMA not addressed here. These are covered in a separate application. This example will work for moderate serial data serial data rates. To operate the SCC at maximum speed, a local RAM should be used as bus arbitration overhead could cause problems. The main purpose here was to illustrate the versatility of PAL devices and how easy it is to interface apparently incompatible devices to the MULTIBUS *.

The two PAL device shown are similar in function to the 8289 and 8288 shown with the 8086 CPU. A similar design could be done for processors such as the 68000 or other bus masters such as the 8052 CRT controller which has its own DMA.

* MULTIBUS is a registered trademark of Intel Corporation

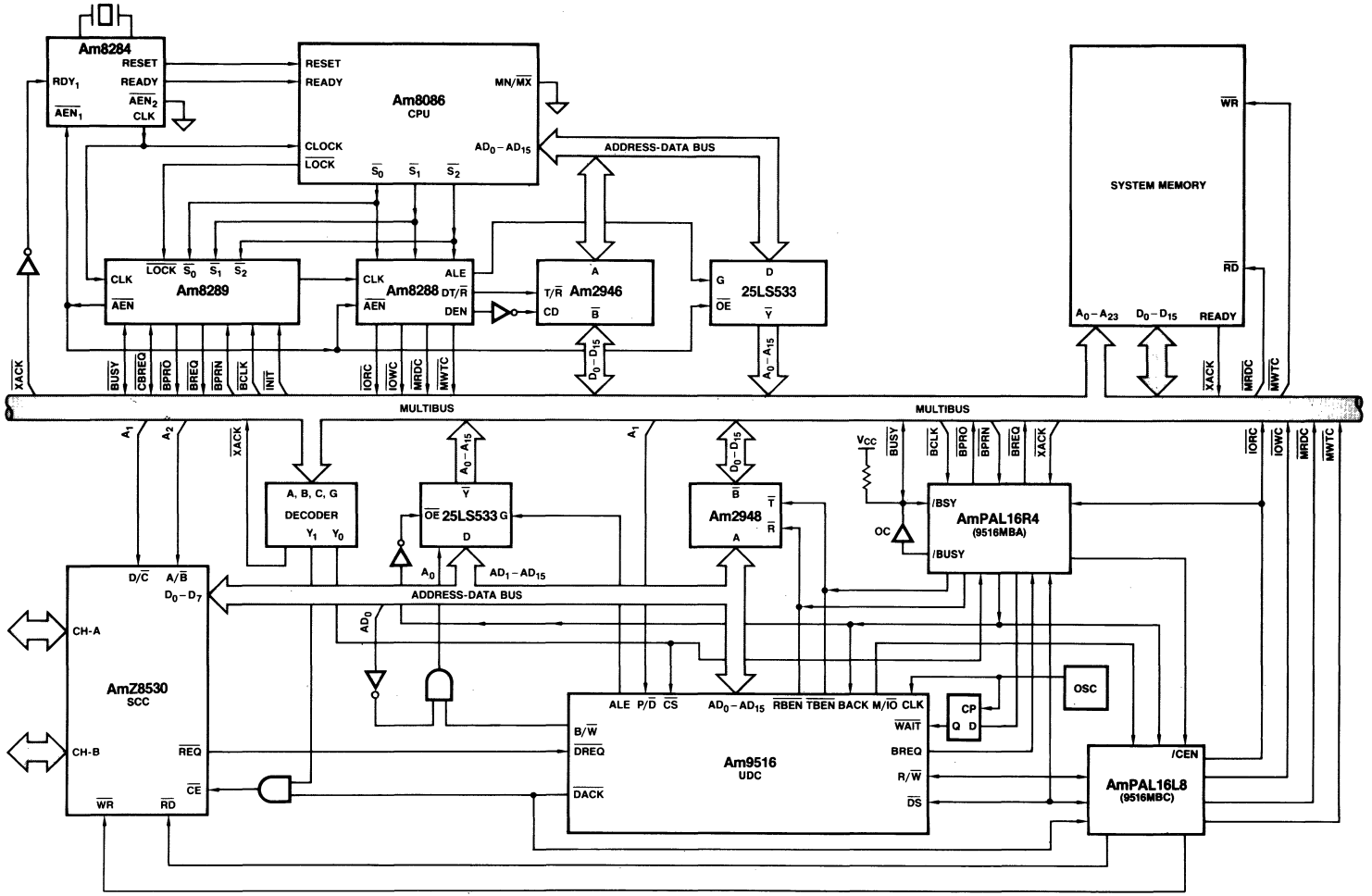


Figure 12-1a Am9516 to 8086 in a MULTIBUS® Environment

02188A-128

```
PAL16L8                                          PAL DESIGN SPECIFICATION
PAT 002                                          JOE BRCICH 9 SEPT 82
MULTIBUS CONTROL FOR 9516
ADVANCE MICRO DEVICES

BACK  MIO   NC   NC   /DACK  NC   NC   NC   /CEN  GND
NC   /RD  /IORC  /DS  /MWTC  /MRDC  /IOWC  /RW  /WR  VCC

IF (BACK) IORC = /MIO*DS*/RW*CEN

IF (BACK) IOWC = /MIO*DS*RW*CEN

IF (BACK) MRDC = MIO*DS*/RW*CEN

IF (BACK) MWTC = MIO*DS*RW*CEN

RD = DACK*RW*BACK + IORC*/BACK

WR = DACK*, RW*BACK + IOWC */BACK

IF (/BACK) DS = IORC + IOWC

IF (/BACK) RW = IOWC


DESCRIPTION

THIS PAL CONVERTS MULTIBUS SIGNALS INTO 9516 COMPATIBLE SIGNALS AND
VICE VERSA.  IT ALSO SUPPORTS THE 8530 IN FLYBY MODE.
```

Figure 12-1b

```
PAL16R4                                          PAL DESIGN SPECIFICATION
PAT 004                                          JOE BRCICH 30 JULY 84
MULTIBUS ARBITER FOR 9516
ADVANCE MICRO DEVICES

/BCLK  /XACK   BRQ  /BSY  /BPRN   /DS   NC   /IORC  /CS  GND
/OE  /RBEN  /TBEN  BACK  /CEN  /BREQ  /BUSY  /BPRO  /WAIT   VCC

IF (/BACK) TBEN = IORC * CS

IF (/BACK) RBEN = /IORC * CS

WAIT = /XACK * BACK

BREQ := BRQ

BPRO = /BRQ * BPRN

/BACK := /BUSY

BUSY :=  BREQ * BPRN * /BSY * /BUSY +
         BREQ * BUSY * BPRN + DS * BUSY

CEN := BACK

DESCRIPTION

/CEN DELAYS THE COMMANDS TO MEET THE MULTIBUS REQUIREMENT THAT ADDRESS
AND DATA BE VALID AT LEAST 50 NS PRIOR TO CONTROL ACTIVE.  IF WE DO
NOT ALLOW PREEMPTION;  REMOVE BPRN FROM THE SECOND EXPRESSION IN THE
BUSY EQUEATION AND ELIMINATE THE THIRD EXPRESSION.
```

Figure 12-1c

ADVANCED MICRO DEVICES DOMESTIC SALES OFFICES

ALABAMA	(205) 882-9122
ARIZONA, Tempe	(602) 242-4400
Tucson	(602) 792-1200
CALIFORNIA, El Segundo	(213) 640-3210
Newport Beach	(714) 752-6262
San Diego	(619) 560-7030
Sunnyvale	(408) 720-8811
Woodland Hills	(818) 992-4155
COLORADO	(303) 691-5100
CONNECTICUT, Southbury	(203) 264-7800
FLORIDA, Altamonte Springs	(305) 834-3333
Clearwater	(813) 530-0971
Ft. Lauderdale	(305) 484-8600
Melbourne	(305) 254-2915
GEORGIA	(404) 449-7920
ILLINOIS	(312) 773-4422
INDIANA	(317) 244-7207
KANSAS	(913) 451-3115
MARYLAND	(301) 796-9310
MASSACHUSETTS	(617) 273-3970
MINNESOTA	(612) 938-0001
NEW JERSEY	(201) 399-0002
NEW YORK, Liverpool	(315) 457-5400
Poughkeepsie	(914) 471-8180
Woodbury	(516) 364-8020
NORTH CAROLINA, Charlotte	(704) 525-1875
Raleigh	(919) 847-8471
OREGON	(503) 245-0080
OHIO, Columbus	(614) 891-6455
PENNSYLVANIA, Allentown	(215) 398-8006
Willow Grove	(215) 657-3101
TEXAS, Austin	(512) 346-7830
Dallas	(214) 934-9099
Houston	(713) 785-9001
WASHINGTON	(206) 455-3600
WISCONSIN	(414) 782-7748

ADVANCED MICRO DEVICES—CANADA

ONTARIO, Ottawa	(613) 592-0060
Willowdale	(416) 224-5193

ADVANCED MICRO DEVICES INTERNATIONAL SALES OFFICES

BELGIUM, Bruxelles	(02) 771-99 93
FRANCE, Rungis Cedex	(01) 687 36 66
GERMANY, Munich	(089) 4114-0
Stuttgart	(0711) 62 33 77
Winsen/Aller	(05143) 5055
HONG KONG, Kowloon	(852) 3 695377
ITALY, Milano	(02) 3390541
JAPAN, Tokyo	(03) 345-8241
SWEDEN, Sundbyberg	(08) 733 03 50
UNITED KINGDOM, Warrington	(0925) 828008
Woking	(04862) 22121

ADVANCED MICRO DEVICES, INC.

901 Thompson Place
P.O. Box 3453
Sunnyvale,
California 94088
(408) 732-2400
TWX: 910-339-9280
TELEX: 34 6306
TOLL FREE
(800) 538-8450

Order #02188A

CBM-MU-5M-8/85-0