CHAPTER 7
PERIPHERAL SUBSYSTEM

The peripheral (I/O) interface is an essential part of any microprocessor based system. It supports communications between the microprocessor and the peripherals. Given the variety of existing peripheral devices, a peripheral system must allow a variety of interfaces. Another important part of a microprocessor system are the buses which connect all major parts of the system. This chapter describes the connection of peripheral devices to the i486™ microprocessor bus. Design techniques are discussed for interfacing different devices including the i486 processor kit which includes LAN Controllers and EISA and MCA chip sets.

The peripheral subsystem must provide sufficiently high data bandwidth to suppport the i486 microprocessor. High speed devices like disks must be able to transfer data to memory with minimal CPU overhead or interaction. The on-chip cache of the i486 microprocessor requires further considerations to avoid stale data problems. These subjects are also covered in this chapter.

The i486 microprocessor supports 8-bit, 16-bit and 32-bit I/O devices which can be I/O mapped, memory mapped, or both. It has a 106 Mbyte/sec memory bandwidth at 33 MHz. Cache coherency is supported by cache line invalidation and cache flush cycles. I/O devices can be accessed by dedicated I/O instructions for I/O mapped devices, or by memory operand instructions for memory mapped devices. In addition, the i486 microprocessor always synchronizes I/O instruction execution with external bus activity. All previous instructions are completed before an I/O operation begins. In particular, all writes pending in the write buffers will be completed before an I/O read or write is performed. These functions are described in this chapter.

7.1 PERIPHERAL/PROCESSOR BUS INTERFACE

Because the i486 microprocessor supports both memory mapped and I/O mapped devices, this section includes brief discussion of the types of mapping, support of dynamic bus sizing, byte swap logic and critical timings. An example of a basic I/O controller implementation is also included. Additionally, some system-oriented interface considerations are discussed because they can have a significant influence on overall system performance.

7.1.1 Mapping Techniques

The system designer should have a thorough understanding of the system application and its use of peripherals in order to design the optional mapping scheme. Two techniques can be used to control the transmission of data between the computer and its peripherals. The most straight-forward approach is I/O mapping.

The i486 microprocessor allows 8-bit, 16-bit or 32-bit I/O devices which can be I/O mapped, memory mapped, or both. All I/O devices can be mapped into physical memory addresses ranging from 00000000H to FFFFFFFFH (four-gigabytes) or I/O addresses ranging from 00000000H to OOOOFFFFH (64 KBytes) for programmed I/O, as shown in Figure 7-1.

I/O mapping and memory-mapping differ in the following respects:

- The address decoding required to generate chip selects for the I/O mapped devices is much simpler than that required for memory mapped devices. I/O mapped devices reside within the I/O space of i486 microprocessor (64 Kbytes); memory-mapped devices reside in a much larger i486 microprocessor memory space (4-gigabytes) which requires more address lines to decode.
- The I/O space is 64 Kbytes in size and can be divided into 64K of 8-bit ports, 32K of 16-bit ports or 16K of 32-bit ports or any combinations of ports which add up to less than 64 Kbytes. The 64 Kbytes of I/O address space refers to physical memory since I/O instructions do not utilize the segmentation or paging hardware and are directly addressable using DX registers.
- Memory-mapped devices can be accessed using i486 microprocessor's instructions, so that I/O to memory, memory to I/O and I/O to I/O transfers as well as compare and test operations can be coded efficiently.

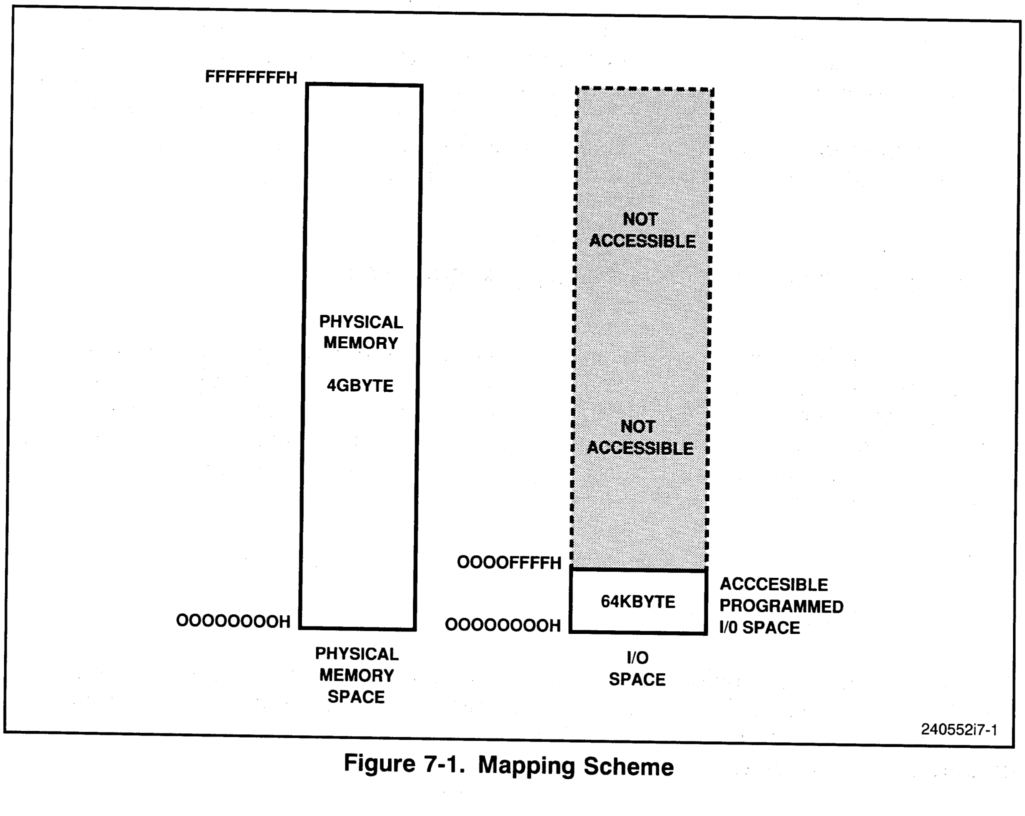

Figure 7-1. Mapping Scheme

- The I/O mapped device can be accessed only via IN, OUT, INS and OUTS instructions. I/O instruction execution is synchronized with external bus activity. All the I/O transfers are performed using the AL (8-bit), AX (16-bit), or EAX (32-bit) registers.
- Memory mapping offers more flexibility in protection than does I/O mapping. Memory mapped devices are protected by the memory management and protection features. A device can be inaccessible to a task, visible but protected, or fully accessible, depending on where it is mapped. Paging and segmentation provide the same protection levels for 4-KByte pages or variable length segments which can be swapped to the disk or shared between programs. The i486 microprocessor supports pages and segments to provide the designer with maximum flexibility.
- The I/O privilege level of the i486 processor protects I/O-mapped devices by either preventing a task from accessing any I/O devices or by allowing a task to access all I/O devices. A virtual-8086 mode I/O permission bitmap can be used to select the privilege level for a combination of I/O bytes.

7.1.2 Dynamic Bus Sizing

Dynamic data bus sizing allows a direct processor connection to 32-, 16- or 8-bit buses for memory or I/O devices. The i486 microprocessor supports dynamic data bus sizing. Data transfers to or from 32-bit, 16-bit or 8-bit devices are supported by determining the bus width during each bus cycle. The decoding circuitry may assert BS16# for 16-bit devices, or BS8# for 8-bit devices for each bus cycle. For addressing 32-bit devices both BS16# and BS8# are negated. If both BS16# and BS8# are asserted, an 8-bit bus width will be assumed.

Appropriate selection of BS16# and BS8# drives the i486 microprocessor to run additional bus cycles to complete requests larger than 16 or 8-bits. When BS16# is asserted, a 32-bit transfer is converted into two 16-bit transfers (or three transfers if the data is misaligned) Similarly asserting BS8# will convert 32-bit transfers into four 8-bit transfers. The extra cycles forced by the BS16# or BS8# should be viewed as independent cycles. BS16# or BS8# are normally driven active during the independent cycles. The only exception would be if the addressed device can vary the number of bytes that it can return between the cycles.

The i486 microprocessor drives the appropriate byte enables during the independent cycles initiated by BS8# and BS16#. Addresses A2-A31 do not change if accesses are to a 32-bit aligned area. Table 7-1 shows the set of byte enables that will be generated on the next cycle for each of the valid possibilities of the byte enables on the current cycle. BEx# must be ignored for 16-byte cycles to memory mapped devices.

The dynamic bus sizing feature of i486 microprocessor is significantly different than that of 386™ DX microprocessor. The i486 microprocessor requires that the data bytes be driven on the addressed lines only, unlike 386 DX microprocessor which expects both high and low order bytes on D0-D15. The simplest example of this function is a 32-bit aligned BS16# read. When the i486 microprocessor reads the two higher order bytes,

Table 7-1. Next Byte-Enable Values for the BSn# Cycles

Current				Next with BS8#				Next with BS16#			
BE3#	BE2#	BE1#	BE0#	BE3#	BE2#	BE1#	BE0#	BE3#	BE2#	BE1#	BE0#
1	1	1	0	n	n	n	n	n	n	n	n
1	1	0	0	1	1	0	1	n	n	n	n
1	0	0	0	1	0	0	1	1	0	1	1
0	0	0	0	0	0	0	1	0	0	1	1
1	1	0	1	n	n	n	n	n	n	n	n
1	0	0	1	1	0	1	1	1	0	1	1
0	0	0	1	0	0	1	1	0	0	1	1
1	0	1	1	n	n	n	n	n	n	n	n
0	0	1	1	0	1	1	1	n	n	n	n
0	1	1	1	n	n	n	n	n	n	n	n

NOTE: "n" means that another bus cycle will not be required to satisfy the request.

they must be driven on D16-D31 data bus, and it expects the two low order bytes on D0-D15. The 386 DX microprocessor always reads or writes data on the lower 16-bits of the data bus when BS16# is asserted.

The external system design must provide buffers to allow i486 microprocessor to read or write data on the appropriate data bus pins. Table 7-2 shows the data bus lines where i486 microprocessor expects valid data to be returned for each valid combination of byte enables and bus sizing options. Valid data is driven only on data bus pins which correspond to byte enables signals that are active during write cycles. Other data pins will also be driven but they will not contain valid data. Unlike 386 DX microprocessor, the i486 microprocessor does not duplicate write data on the data bus when corresponding byte enables are negated.

The BS16# and BS8# inputs allow external 16- and 8-bit buses to be supported using fewer external components. The i486 microprocessor samples these pins every clock cycle. This value is sampled on the clock before ready and determines the bus size. When BS8# or BS16# is asserted then only 16- or 8-bits of data are valid. If both BS8# and BS16# are asserted, an 8-bit bus width is valid.

Table 7-2. Valid Data Lines for Valid Byte Enable Combinations

BE3#	BE2#	BE1#	BE0#	w/o BS8#/BS16#	w BS8#	w BS16#
1	1	1	0	D7–D0	D7–D0	D7–D0
1	1	0	0	D15–D0	D7–D0	D15–D0
1	0	0	0	D23–D0	D7–D0	D15–D0
0	0	0	0	D31–D0	D7–D0	D15–D0
1	1	0	1	D15–D8	D15–D8	D15–D8
1	0	0	1	D23–D8	D15–D8	D15–D8
0	0	0	1	D31–D8	D15–D8	D15–D8
1	0	1	1	D23–D16	D23–D16	D23–D16
0	0	1	1	D31–D16	D23–D16	D31–D16
0	1	1	1	D31–D24	D31–D24	D31–D24

Dynamic bus sizing allows the power-up and boot up program to be stored in eight-bit EPROM while high speed program execution uses 32-bit memory.

7.1.2.1 ADDRESS DECODING FOR I/O DEVICES

Address decoding for I/O devices resembles address decoding for memories. The primary difference is that the block size (range of addresses) for each address signal is much smaller. The minimum block size is dependent on the number of addresses used by the I/O device. In most processors, where I/O instructions are separate, I/O addresses are shorter than memory addresses. Typically, microprocessors with 16-bit address bus use 8-bit address for I/O.

One technique for decoding memory-mapped I/O address is to map the entire I/O space of the i486 microprocessor into a 64-Kbyte region of the memory space. The address decoding logic can be reconfigured so that each I/O device responds to a memory address and an I/O address. This configuration is compatible with software that uses either I/O instructions memory-mapped or I/O techniques.

Addresses can be assigned arbitrarily within the I/O or memory space. Addresses for either I/O mapped or memory mapped devices should be selected to minimize the number of address lines needed.

7.1.2.1.1 Address Bus Interface

Figure 7-2 shows the i486 microprocessor address interface to 32-, 16- and 8-bit devices. To address 16-bit devices, the byte enables must be decoded to produce A1, BHE# and BLE# (A0) signal.

To access to 8-bit devices, the byte enable signals must be decoded to generate A0 and A1. Because A0 and BLE# are the same, the same generation logic can be used. For 32-bit memory/mapped devices A2–A31 can be used in conjunction with BE3#–BE0#. This logic is shown in Figure 7-3.

7.1.2.1.2 8-Bit I/O Interface

Due to the presence of dynamic data bus sizing and the variety of byte-enable pin combinations (Table 7-2), byte swapping logic for 32-to-8-bit conversions can be implemented in various ways.

This section discusses an example in which BE0#–BE3# = Low and D0–D7 are used when BS8# is enabled.

Figure 7-4 shows the interfacing of i486 microprocessor to an 8-bit device. This implementation requires seven 8-bit bidirectional data buffers.

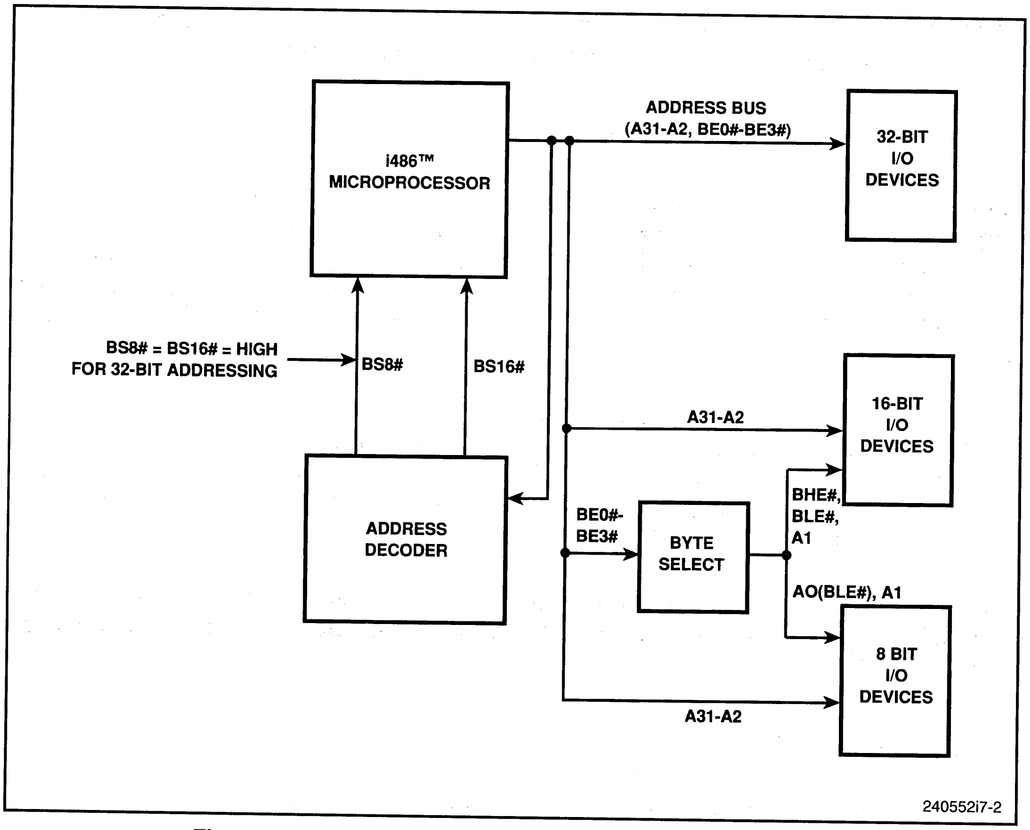

Figure 7-2. Address Interface to 32/16/8-Bit I/O Devices

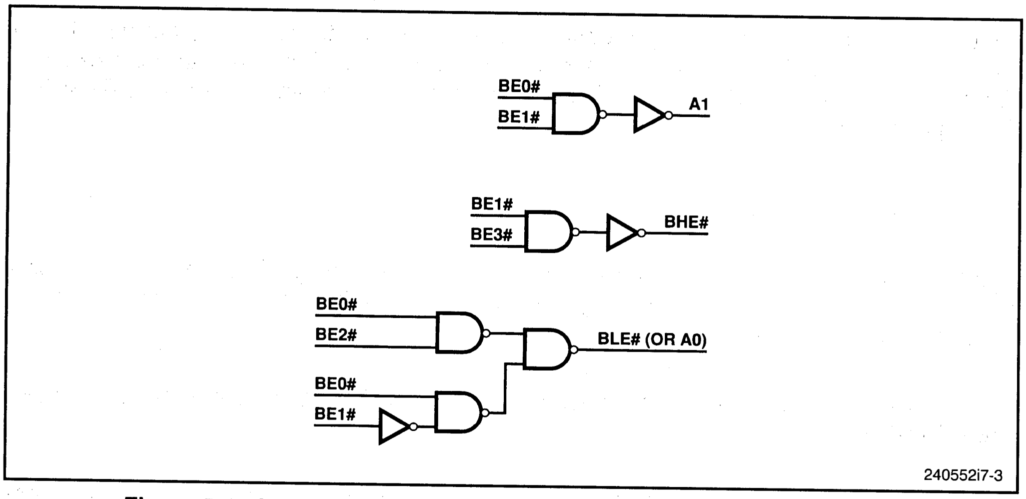

Figure 7-3. Logic to Generate A1, BHE# and BLE# for 16-Bit Buses

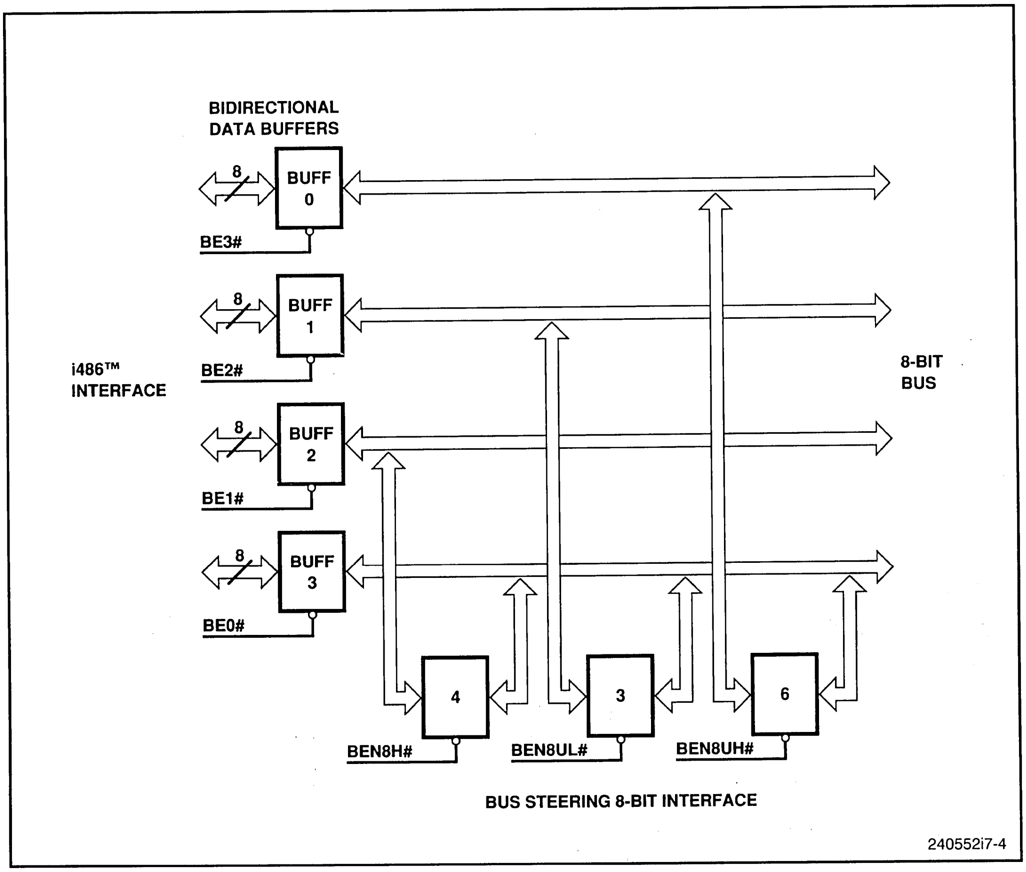

Figure 7-4. i486™ Microprocessor Interface to 8-Bit Device

In this example, in case of 32-bit writes, the BE0#–BE3# are enabled, hence 32 bits of data reside on the data buffer outputs. This is then swapped based on the control signals. Buffers are enabled in the following manner:

For Byte # 0	Buffer 3 is enabled (BE0# is true)
For Byte # 1	Buffer 2 and 4 are enabled (BE1# and BEN8H#)
For Byte # 2	Buffer 1 and 5 are enabled (BE2# and BEN8UL#)
For Byte # 3	Buffer 0 and 6 are enabled (BE3# and BEN8UH#)

Table 7-3 shows the truth table for 8-bit I/O interface to the i486 microprocessor. The table also contains the values of the control signals used to enable the second set of buffers. The PLD equations used to implement these signals are shown below:

PLD Input Signals:

BS8#: The signal is from an 8-bit device or from the system logic that interfaces to an 8-bit device.

BE0#–BE3#: When processor drives all of these signals Low, external logic should look only for BE0# while in 8-bit mode.

ADS#: An address strobe from the i486 microprocessor indicates a valid processor cycle.

OUTPUTS BEN8H#, BEN8UH, BEN8UL: Byte enables for 8 bit interface.

EQUATIONS

```
BEN8H  = ADS * BE1 * /BE0 * BS8                     ;Swapping second byte for 8 bit
            + /ADS * BEN8H                          ;interface

BEN8UL = ADS * BE2 * /BE1 * /BE0 * BS8              ;Swapping third byte for
            + /ADS * BEN8UL                         ;8-bit interface

BEN8UH = ADS * BE3 * /BE2 * /BE1 * /BE0 * BS8
            + /ADS * BEN8UH                         ;Swapping fourth byte for 8-bit
                                                    ;interface
```

Table 7-3. 32-Bit to 8-Bit Steering

i486™ CPU				8-Bit Interface						
BE3#	**BE2#**	**BE1#**	**BE0#**	**BEN16#**	**BEN8UH#**	**BEN8UL#**	**BEN8H#**	**BHE#**	**A1**	**A0**
0	0	0	0	H	H	H	H	X	L	L
1	0	0	0	H	H	H	H	X	L	L
0	1	0	0*	H	H	H	H	X	X	X
1	1	0	0	H	H	H	H	X	L	L
0	0	1	0*	H	H	H	H	X	X	X
1	0	1	0*	H	H	H	H	X	X	X
0	1	1	0*	H	H	H	H	X	X	X
1	1	1	0	H	H	H	H	X	L	L
0	0	0	1	H	H	H	L	X	L	H
1	0	0	1	H	H	H	L	X	L	H
0	1	0	1*	H	H	H	L	X	X	X
1	1	0	1	H	H	H	L	X	L	H
0	0	1	1	H	H	L	H	X	H	L
1	0	1	1	H	H	L	H	X	H	L
0	1	1	1	H	L	H	H	X	H	H
1	1	1	1	H	H	H	H	X	X	X
← Inputs →				← Outputs →						

NOTES:
X: Do not care (either L or H).
BHE# (byte high enable) is not needed in 8-bit interface.
*A non-occurring pattern of byte enables; either none are asserted or the pattern has byte enables asserted for non-contiguous bytes.

7.1.2.1.3 16-Bit I/O Interface

16-bit I/O interface byte swap logic requires six eight-bit bidirectional I/O data buffers as shown in Figure 7-5. Buffers 0 through 3 are controlled by BE0–BE3# respectively. Buffers 4 and 5 are monitored by BEN16#.

To transfer data on the lower 16-bits, buffers 2 and 3 are enabled. While the higher 16-bits are transferred through Buffer 0, 1, 4 and 5.

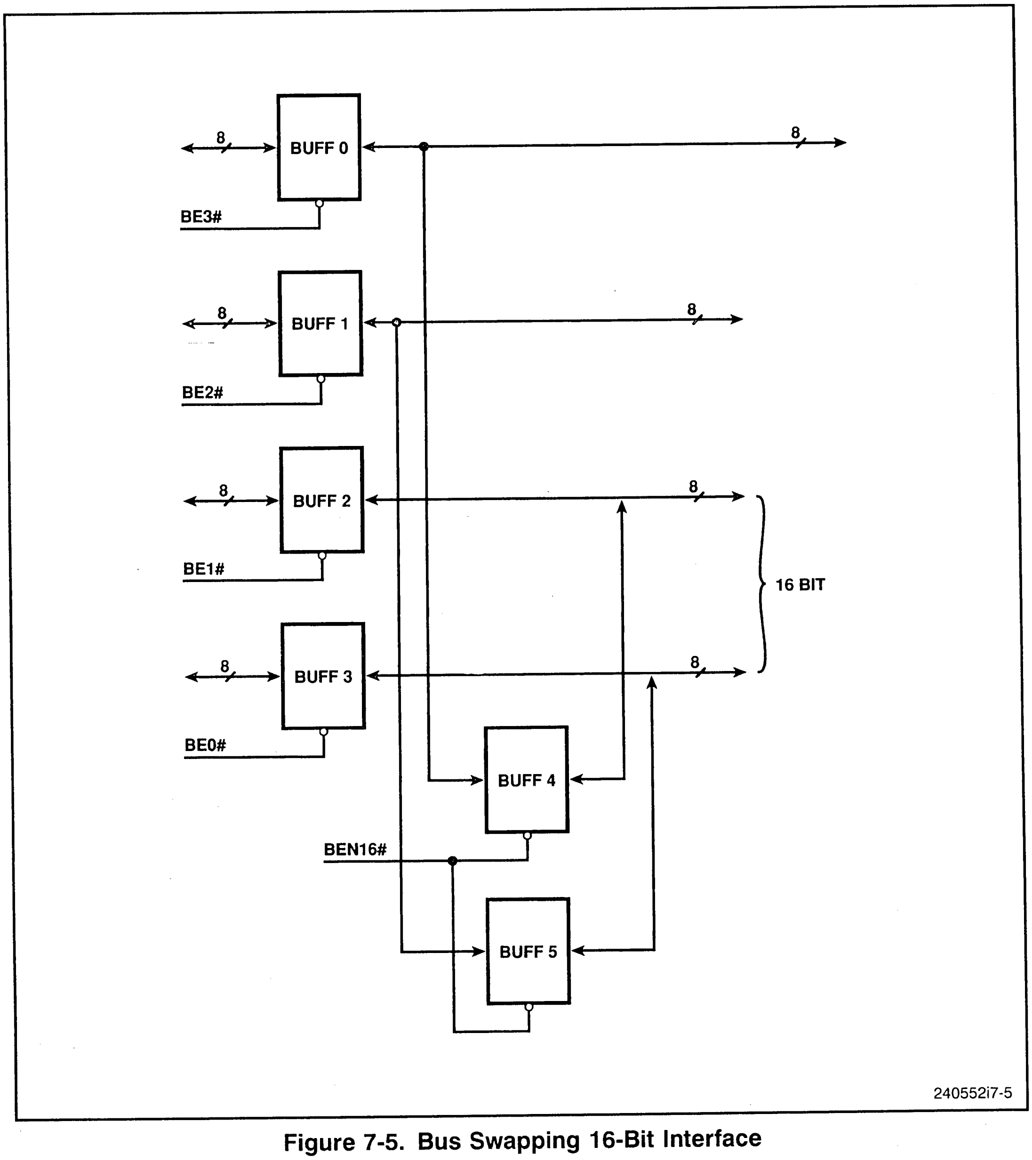

Figure 7-5. Bus Swapping 16-Bit Interface

Table 7-4 shows the truth table for 32-to-16-bit Bus Swapping logic and A0, A1 and BEH# generation.

The PLD equation used to implement 32-bit-to-16-bit byte swap/steer logic is shown below:

PLD INPUT SIGNALS:

BS16#: Either from a 16-bit device or from system logic which indicates a 16-bit transfer.

BE0#–BE3#: Byte enable inputs from i486 CPU. In 16-bit mode, the external logic should look at BE0# and BE1# only.

ADS#: Address strobe from an i486 CPU indicating a valid CPU cycle.

PLD OUTPUT SIGNALS:

BEN16: Word enable for 16-bit interface.

EQUATION:

```
BEN16 = ADS * BE2 * /BE1 * /BE0 * BS16 * /BS8        ;swapping upper 16-bits
        + ADS * BE3 * /BE1 * /BE0 * BS16 * /BS8
        + /ADS * BEN16
```

Table 7-4. 32-Bit to 16-Bit Bus Swapping Logic Truth Table

i486™ Microprocessor				16-Bit Interface						
BE3#	BE2#	BE1#	BE0#	BEN16#	BEN8UH#	BEN8UL#	BEN8H#	BHE#	A1	A0
0	0	0	0	H	H	H	H	L	L	L
1	0	0	0	H	H	H	H	L	L	L
0	1	0	0*	H	H	H	H	X	X	X
1	1	0	0	H	H	H	H	L	L	L
0	0	1	0*	H	H	H	H	X	X	X
1	0	1	0*	H	H	H	H	X	X	X
0	1	1	0*	H	H	H	H	X	X	X
1	1	1	0	H	H	H	H	H	L	L
0	0	0	1	H	H	H	H	H	L	H
1	0	0	1	H	H	H	H	H	L	H
0	1	0	1*	H	H	H	H	X	X	X
1	1	0	1	H	H	H	H	H	L	H
0	0	1	1	L	H	H	H	L	H	L
1	0	1	1	L	H	H	H	H	H	L
0	1	1	1	L	H	H	H	H	H	H
1	1	1	1*	H	H	H	H	X	X	X
← Inputs →				← Outputs →						

*A non-occurring pattern of byte enables; either none are asserted or the pattern has byte enables asserted for non-contiguous bytes.

The logic needed to generate the byte swapping control signals for 32-bit-to-8-bit and 32-bit-to-16-bit data transfer can be implemented in PLDs. Propagation delay of the PLD and the bidirectional Buffer propagation delay of 9 ns max must be taken into consideration. This adds into data set-up time for CPU Read cycle and data valid delay for the CPU Write cycle. The byte-swapping and address bit generation logic is shown in Figure 7-6.

7.1.2.1.4 32-Bit Interface

A simple 32-bit I/O interface is shown in Figure 7-7. The example uses only four 8-bit wide bidirectional buffers which are enabled by BE3#–BE0#. Table 7-2 provides different combinations of BE3#–BE0#. To provide greater flexibility in I/O interface implementation, the design should include interfaces for 32-, 16- and 8-bit devices. The truth table for 32-to-32-bit interface is shown in Table 7-5.

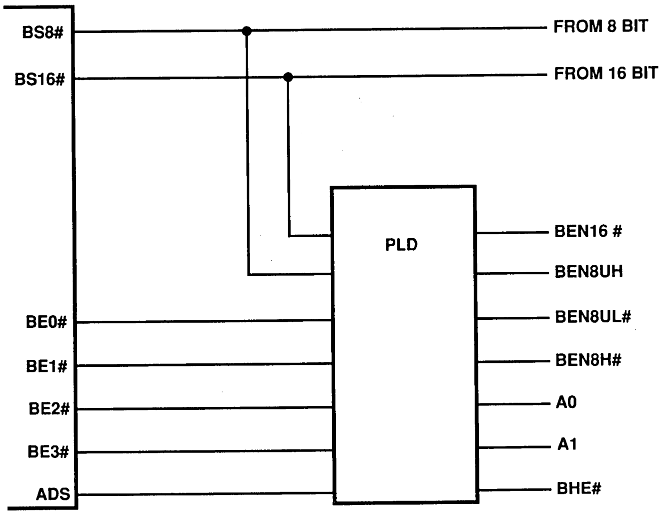

Figure 7-6. Bus Swapping and Low Address Bit Generating Control Logic

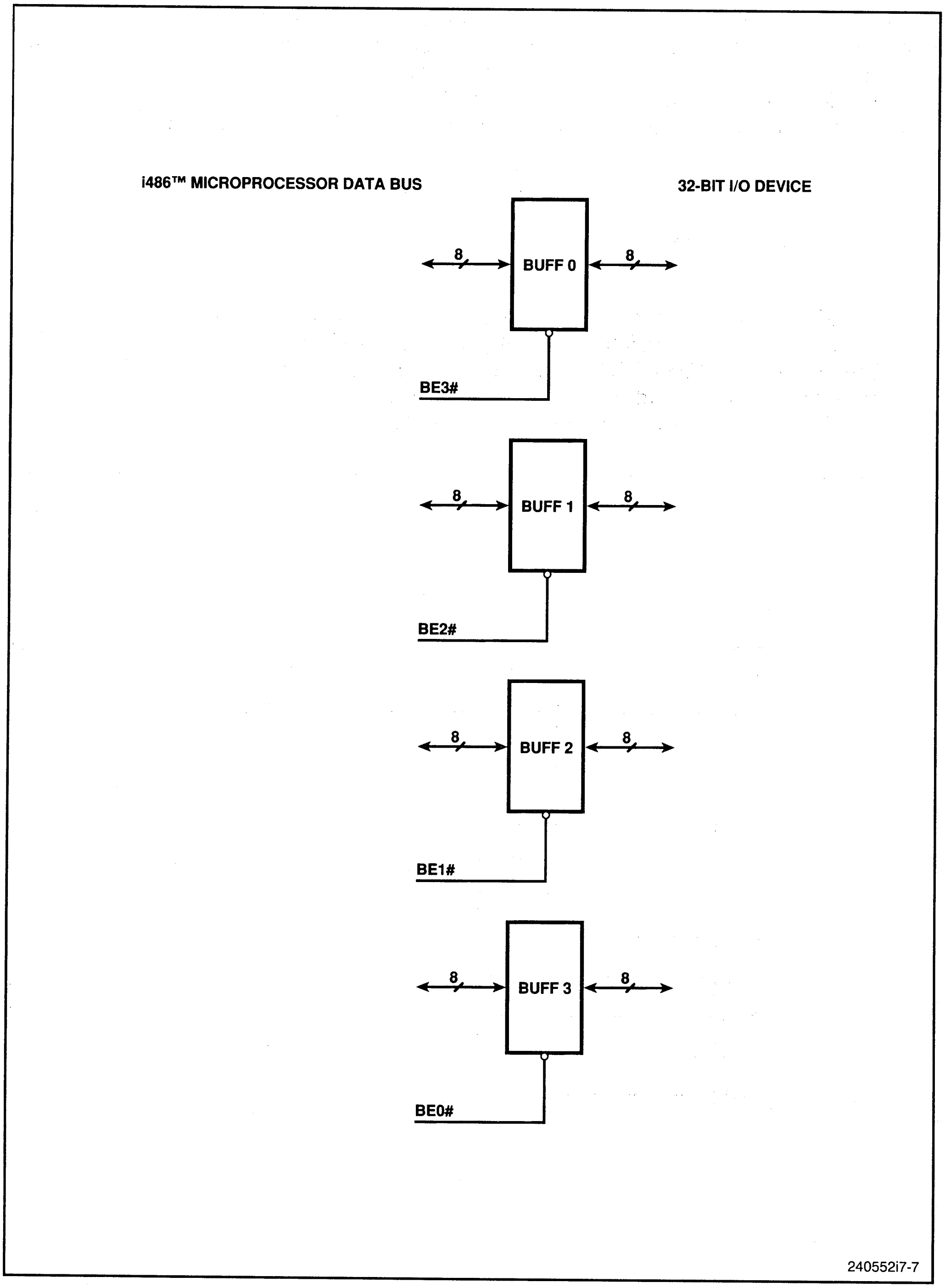

Figure 7-7. 32-Bit I/O Interface

Table 7-5. 32-Bit to 32-Bit Bus Swapping Logic Truth Table

i486™ CPU				32-Bit Interface						
BE3#	BE2#	BE1#	BE0#	BEN16#	BEN8UH#	BEN8UL#	BEN8H#	BHE#	A1	A0
0	0	0	0	H	H	H	H	X	X	X
1	0	0	0	H	H	H	H	X	X	X
0	1	0	0*	H	H	H	H	X	X	X
1	1	0	0	H	H	H	H	X	X	X
0	0	1	0*	H	H	H	H	X	X	X
1	0	1	0*	H	H	H	H	X	X	X
0	1	1	0*	H	H	H	H	X	X	X
1	1	1	0	H	H	H	H	X	X	X
0	0	0	1	H	H	H	H	X	X	X
1	0	0	1	H	H	H	H	X	X	X
0	1	0	1*	H	H	H	H	X	X	X
1	1	0	1	H	H	H	H	X	X	X
0	0	1	1	H	H	H	H	X	X	X
1	0	1	1	H	H	H	H	X	X	X
0	1	1	1	H	H	H	H	X	X	X
1	1	1	1*	H	H	H	H	X	X	X
← Inputs →				← Outputs →						

*A non-occurring pattern of byte enables; either none are asserted or the pattern has byte enables asserted for non-contiguous bytes.

7.2 BASIC PERIPHERAL SUBSYSTEM

All microprocessor systems include a microprocessor, memory and I/O devices which are linked by the address, data and control buses. Figure 7-8 illustrates the system block diagram of a typical i486 microprocessor-based system.

A typical system consists of four subsystems. The heart of the system is the processor core. The memory subsystem is also important and must be efficient and optimized to provide peak system level performance. As described in Chapter 5, it is necessary to utilize the burst-bus feature of the i486 microprocessor for the DRAM control implementation. Cache subsytem as described in Chapter 6 also plays an important role in overall system performance. For many systems however, the on-chip cache provides sufficient performance.

A high-performance i486 microprocessor-based system, requires an efficient peripheral subsystem. This section describes the elements of this system which includes the I/O devices on the expansion bus (the memory bus) and the Local I/O bus. In a typical system a number of slave I/O devices can be controlled through the same local bus interface. Complex peripheral devices which can act as bus masters may require more complex interface.

Although the i486 microprocessor can interface to the peripherals of the earlier architecture (Section 7.5), Intel has added new members to the peripheral subsystem family as shown in Table 7-6.