iAPX 43203
VLSI INTERFACE PROCESSOR

PRELIMINARY

- **Fully Independent and Decentralized I/O**
- **Buffered Data Path for High-Speed Burst-Mode Transfers**
- **Initialization/Diagnostic Interface to 432 Systems**
- **Multiple 43203's per System Provide Incremental I/O Capacity**
- **Protected I/O Interface to 432 Memory**
- **Silicon Operating System Instruction Set Extensions for Attached iAPX Processors**
- **Multibus™ System Compatible Interface**
- **Functional Redundancy Checking Mode for Hardware Error Detection**

The Intel 43203 Interface Processor (IP) provides I/O facilities in iAPX 432 micromainframe systems employing peripheral subsystems. An IP maps a portion of the peripheral subsystem address space into iAPX 432 system memory. As any iAPX 432 processor, the IP operates in an object-oriented, capability-based, multiprocessing environment.

The 43203 is a VLSI device, fabricated with Intel's highly reliable +5 volt, depletion load, N-channel, silicon gate HMOS technology, and is packaged in a 64-pin Quad In-Line Package (QUIP). Refer to Figure 1 for the QUIP representation of the 43203 pin configuration.

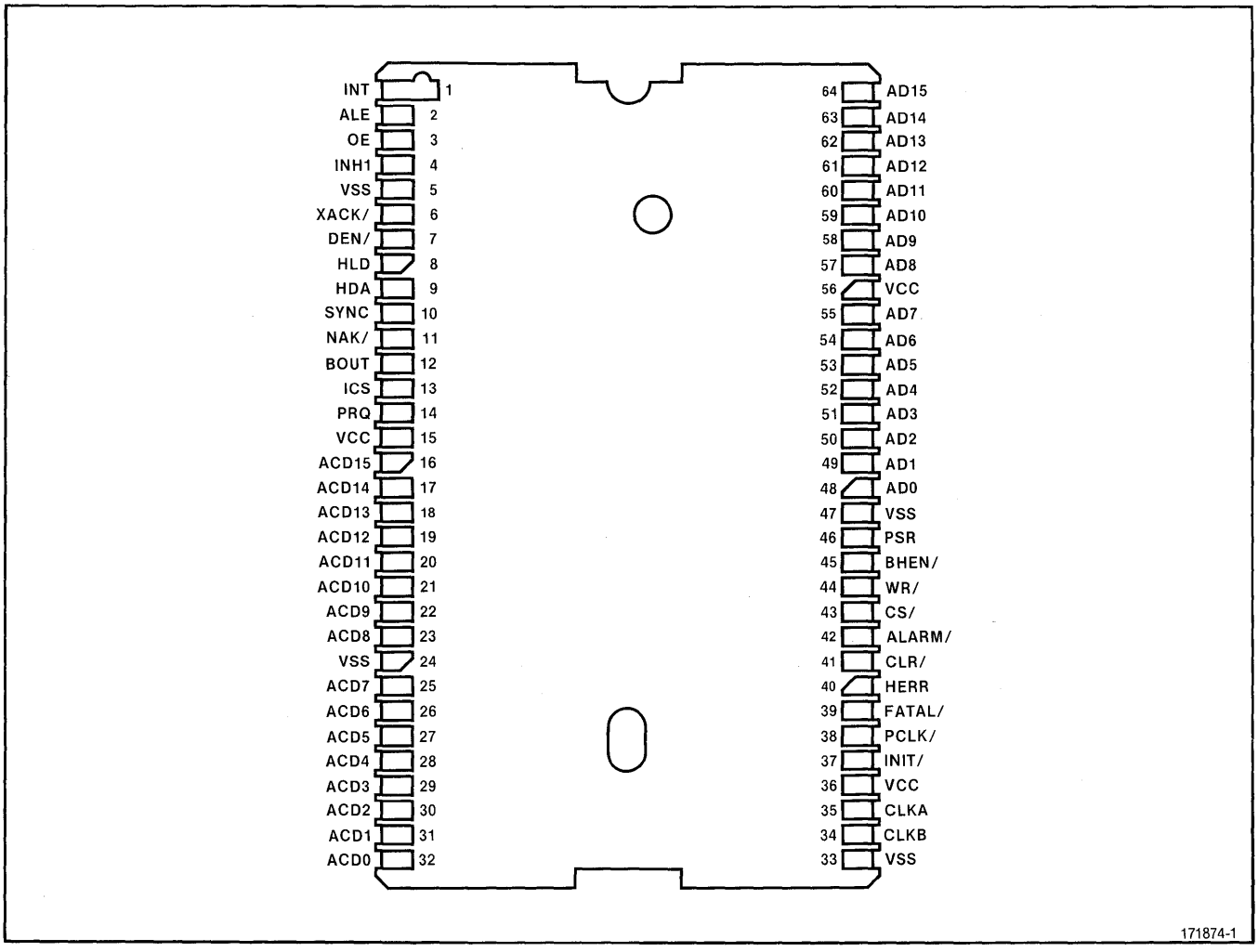


Figure 1. iAPX 43203 Interface Processor Pin Configuration

FUNCTIONAL DESCRIPTION

The block diagram shown in Figure 2 illustrates the internal architecture of the Interface Processor. Figure 3 represents the Interface Processor as a logical device and illustrates the signal interface to the Processor Packet bus (left side) and the peripheral subsystem (right side). The Interface Processor (IP) operates in conjunction with an Attached Processor (AP) to form the logical I/O processor of an iAPX 432 system.

The IP acts as a slave to the AP, and maps a portion of the AP's peripheral subsystem address space into iAPX 432 system memory with the same protection mechanisms as any iAPX 432 processor. Five peripheral subsystem (PS) memory subranges may be mapped into iAPX 432 memory segments. These five windows (labeled 0 through 4) allow the AP to reference iAPX 432 memory with logical addresses or, in special circumstances, with direct, 24-bit physical addresses.

A BASIC I/O MODEL

A typical application based on the iAPX 432 microprocessor family consists of a main system and one or more peripheral subsystems. Figure 4 illustrates a hypothetical configuration that employs two peripheral subsystems. The main system hardware is composed of one or more iAPX 432 general data processors (GDPs) and a common memory that is shared by these processors. The main system software is a collection of one or more processes that execute on the GDP(s).

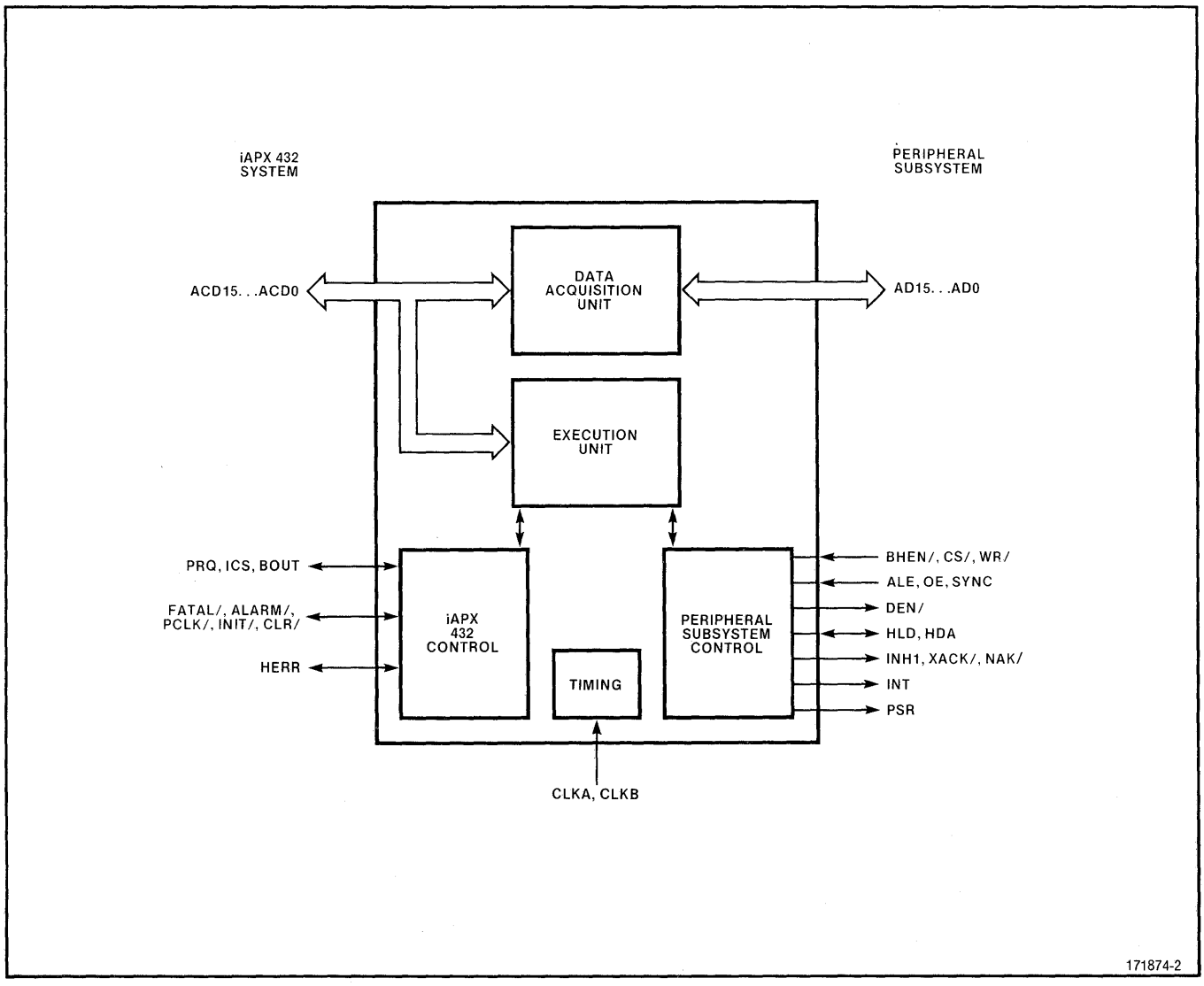


Figure 2. iAPX 43203 IP Functional Block Diagram

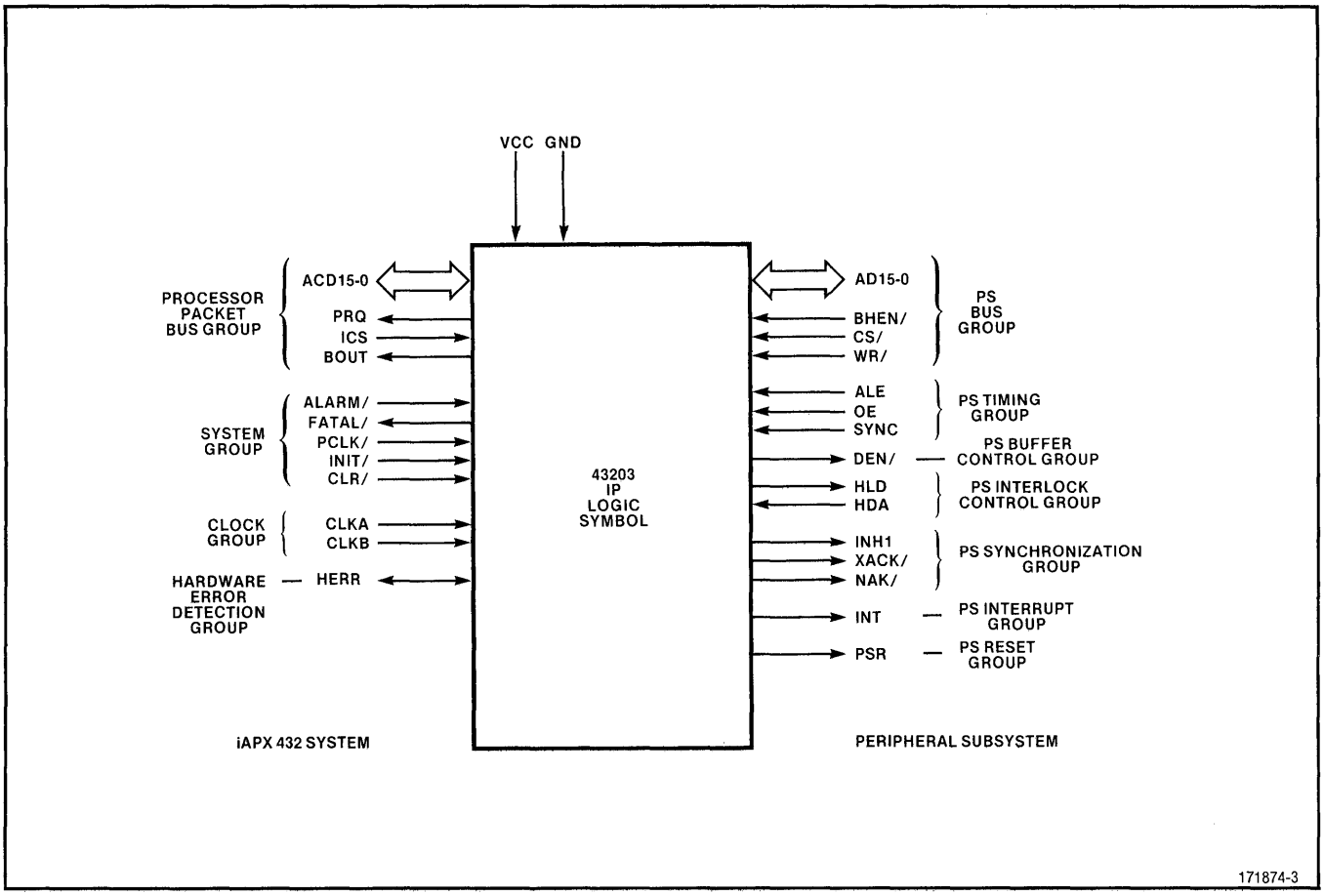


Figure 3. iAPX 43203 IP Logic Symbol

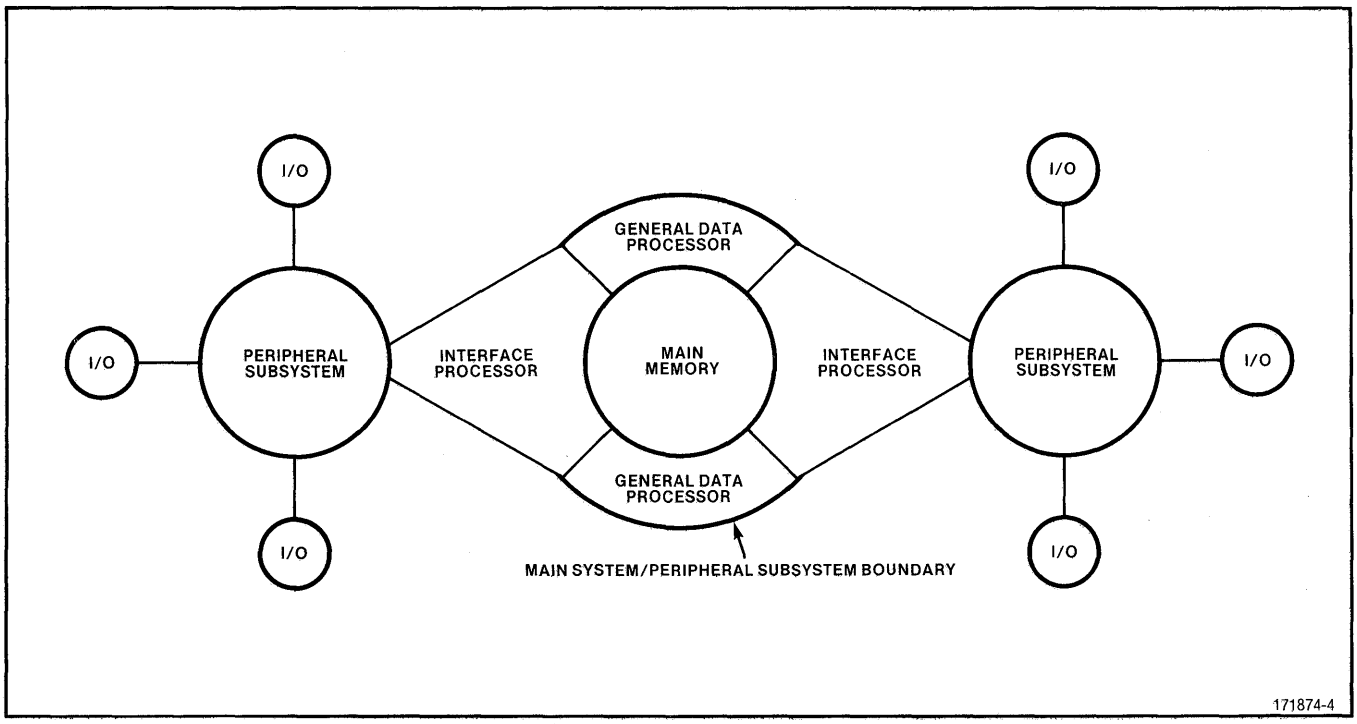


Figure 4. Main System and Peripheral Subsystem

A fundamental principle of the iAPX 432 architecture is that the main system environment is self-contained; neither processors nor processes have any direct contact with the "outside world." Conceptually, the main system is enclosed by a wall that protects objects in memory from possible damage by uncontrolled I/O operations.

In an iAPX 432-based system, the bulk of processing required to support input/output operations is delegated to peripheral subsystems; this includes device control, timing, interrupt handling and buffering. A peripheral subsystem is an autonomous computer system with its own local memory, I/O devices and controllers, at least one processor, and software. The number of peripheral subsystems employed in any given application depends on the I/O-intensiveness of the application, and may be varied with changing needs, independent of the number of GDPs in the system.

A peripheral subsystem resembles a mainframe channel in that it assumes responsibility for low-level I/O device support, executing in parallel with main system processor(s). Unlike a simple channel, however, each peripheral subsystem can be configured with a complement of hardware and software resources that precisely fits application cost and performance requirements. In general, any system that can communicate over a standard 8- or 16-bit microcomputer bus such as Intel's Multibus design may serve as an iAPX 432 peripheral subsystem.

A peripheral subsystem is attached to the main system by means of an IP. At the hardware level, the IP presents two separate bus interfaces. One of these is the standard iAPX 432 Processor Packet bus and the other is a very general interface.

To support the transfer of data through the wall that separates a peripheral subsystem from the main system, the IP provides a set of software-controlled windows. A window is used to expose a single object in main system memory so that its contents may be transferred to or from the peripheral subsystem.

The IP also provides a set of functions that are invoked by software. While the operation of these functions varies considerably, they generally permit objects in main system memory to be manipulated as entities, and enable communication between main system processes and software executing in a peripheral subsystem.

It is important to note that both the window and function facilities utilize and strictly enforce the standard iAPX 432 addressing and protection systems. Thus, a window provides protected access to an object, and a function provides a protected way to operate in the main system. The IP permits data to flow across the peripheral subsystem boundary while preserving the integrity of the main system.

As Figure 5 illustrates, input/output operations in an iAPX 432 system are based on the notion of passing messages between main system processes and device interfaces located in a peripheral subsystem. A device interface is considered to be the hardware and software in the peripheral subsystem that is responsible for managing an I/O device. An I/O device is considered to be a "data repository," which may be a real device (e.g., a terminal), a file, or a pseudo-device (e.g., a spooler).

A message sent from a process that needs an I/O service contains information that describes the requested operation (e.g., "read file XYZ"). The device interface interprets the message and carries out the operation. If an operation requests input data, the device interface returns the data as a message to the originating process. The device interface may also return a message to positively acknowledge completion of a request.

A very general and very powerful mechanism for passing messages between processes is inherent in the iAPX 432 architecture. A given peripheral subsystem may, or may not, have its own message facility, but in any case, such a facility will not be directly compatible with the iAPX 432's. By interposing a peripheral subsystem interface at the subsystem boundary, the standard IP communication system can be made compatible with any device interface (see Figure 6).

iAPX 432 SYSTEM INTERFACE

The IP exists in both the protected environment of the iAPX 432, and the conventional environment of the peripheral subsystem. Because of this, an IP is able to provide a pathway over which data may flow between the iAPX 432 system and the external subsystem. The IP operates at the boundary between the systems, providing compatibility and protection. In this position, the Interface Processor presents two different views of itself, one to software and processors in the iAPX 432 environment and another to its attached processor.

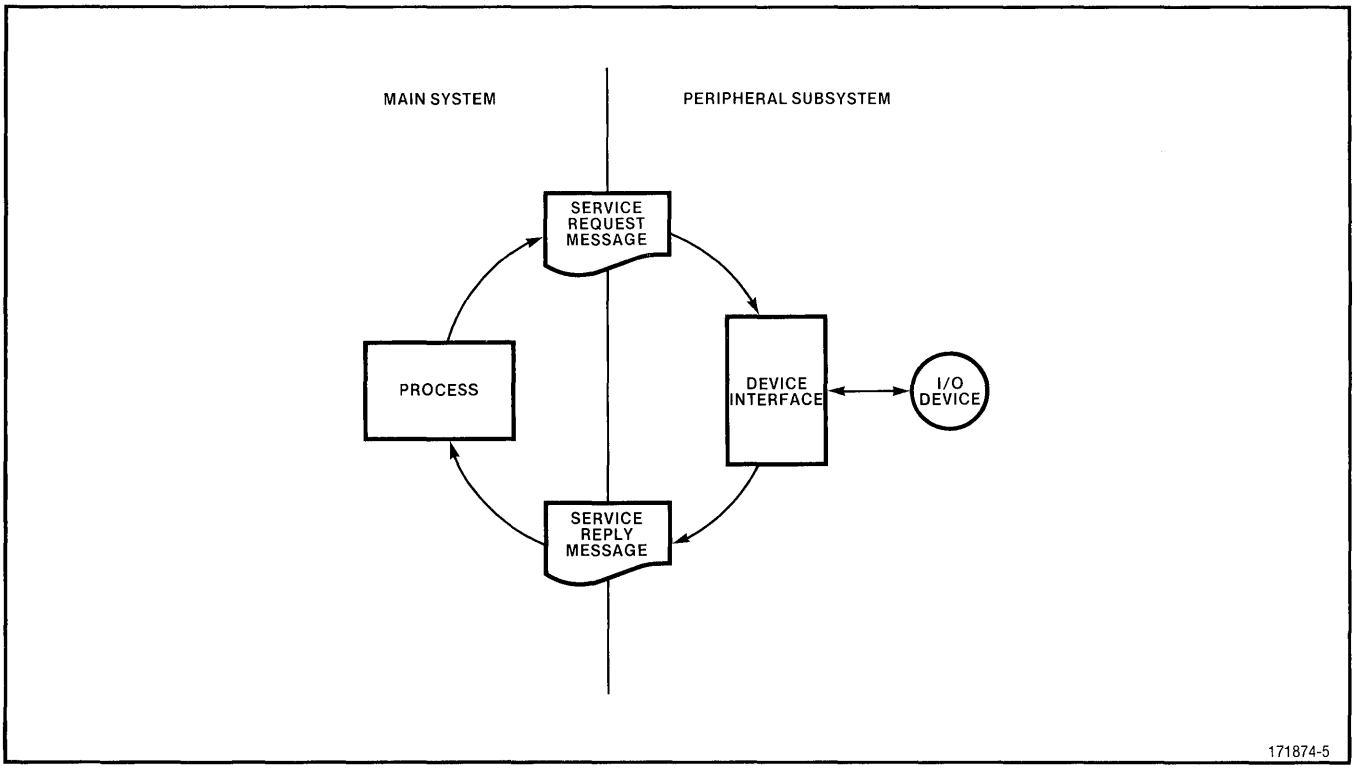


Figure 5. Basic IO Service Cycle

From the iAPX 432 side, an IP looks and behaves very much like any other processor. It attaches to the Processor Packet bus in the same way as a GDP. Within the iAPX 432 memory, the IP supports an execution environment that is compatible with, and largely identical to, the GDP. Thus, the IP recognizes and manipulates system objects representing processors, processes, ports, etc. It supports and enforces the iAPX 432's access control mechanisms, and provides full interprocess and interprocessor communication facilities.

The principal difference between the two processors is that the GDP manipulates its environment in response to the instruction it fetches, while the IP operates under the direction of its attached processor. Indeed, the IP may be said to extend the instruction set of the Attached Processor (AP) so that it may function in the environment of the iAPX 432 system.

PERIPHERAL SUBSYSTEM INTERFACE

A peripheral subsystem interface (PSI) is a collection of hardware and software that acts as an adapter that enables message-based communication between a process in the main system and a device interface in a peripheral subsystem (see Figure 6). Viewed from the iAPX 432 side, the peripheral subsystem interface appears to be a process. The peripheral subsystem interface may be designed to present any desired appearance to a device interface. For example, it may look like a collection of tasks, or like a collection of subroutines.

Hardware

The PSI hardware consists of an IP, an AP, and local memory (see Figure 7). To improve performance, these may be augmented by a DMA controller. The AP and the IP work together as a team, each providing complementary facilities. Considered as a whole, the AP/IP pair may be thought of as a logical I/O processor that supports software operations in both the main system and the peripheral subsystem.

ATTACHED PROCESSOR

Almost any general-purpose CPU, such as an 8085, an iAPX 86 or an iAPX 88 can be used as an AP. The AP need not be dedicated exclusively to working with the IP.

It may, for example, also execute device interface software. Thus, the AP may be the only CPU in the peripheral subsystem, or it may be one of several.

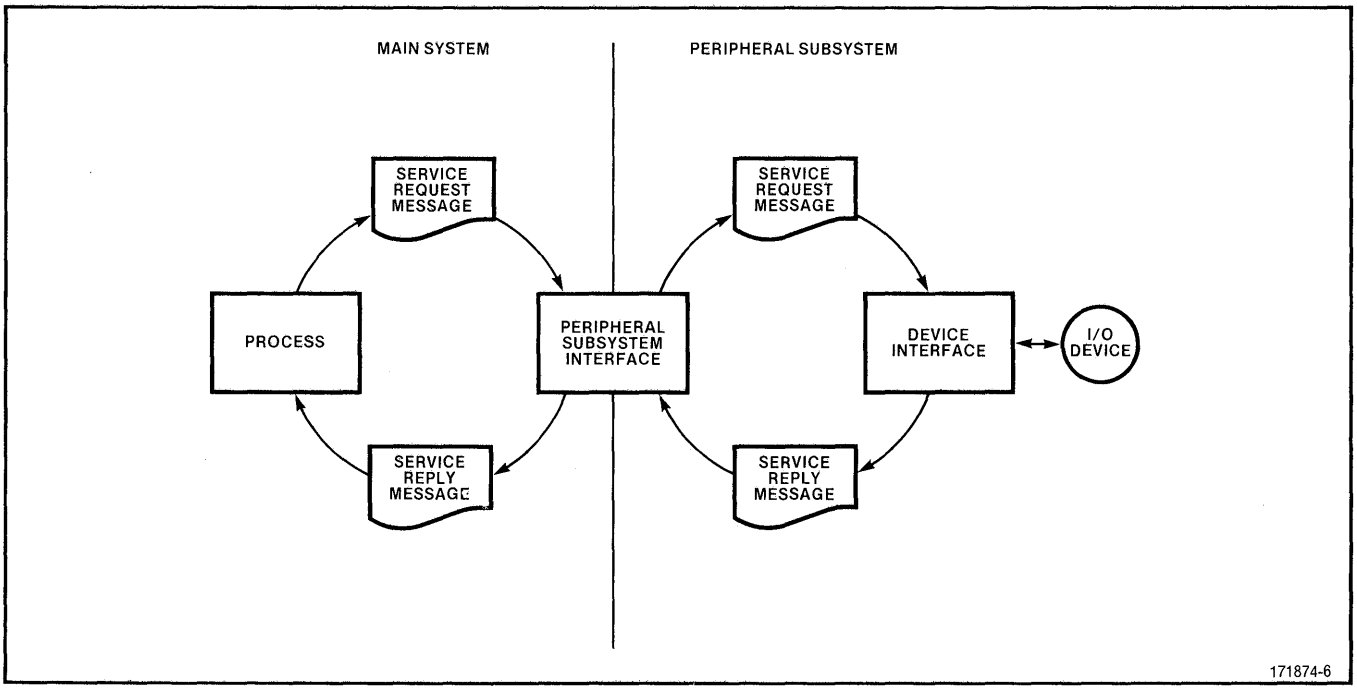


Figure 6. Peripheral Subsystem Interface

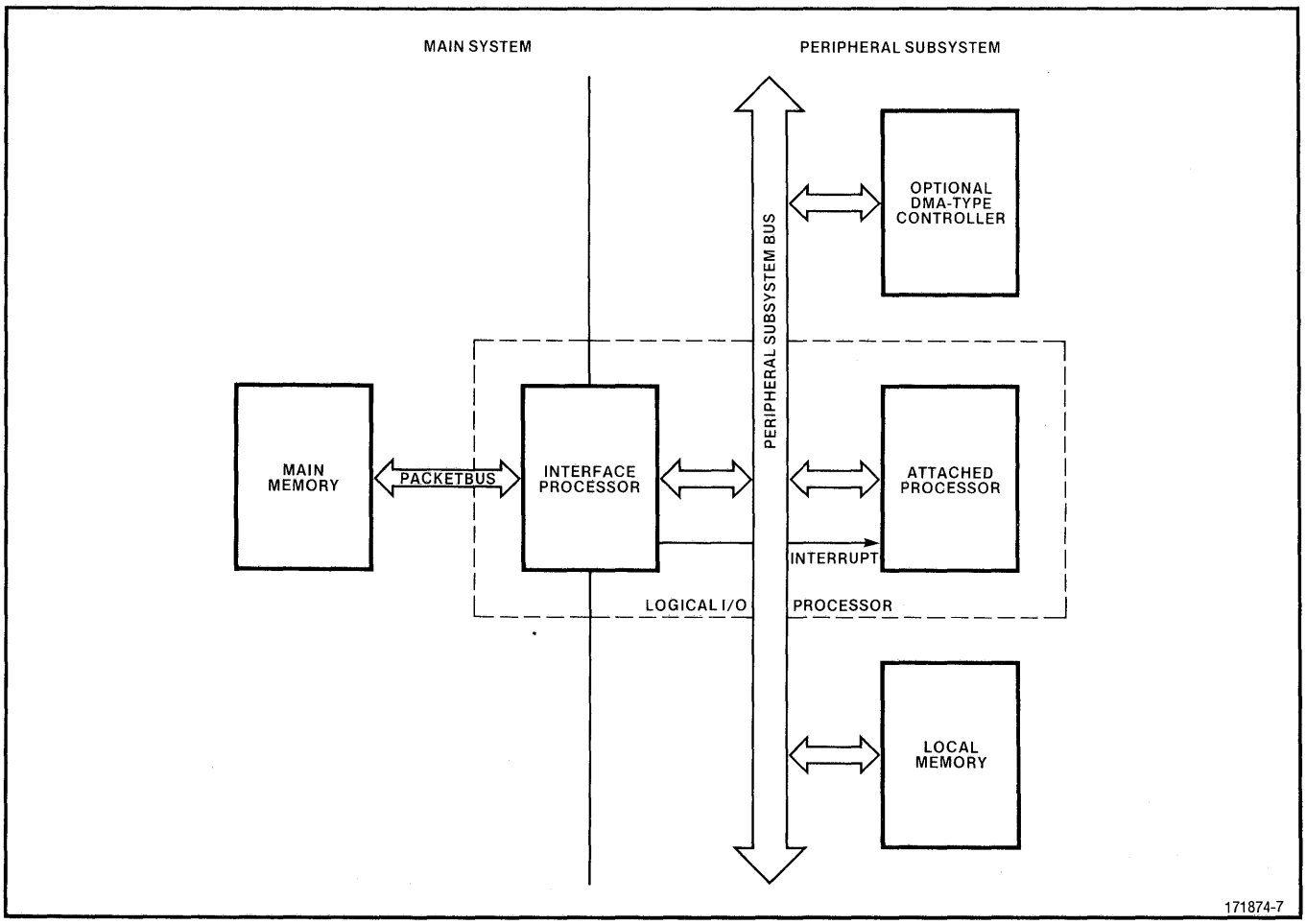


Figure 7. Peripheral Subsystem Interface Hardware

As Figure 7 shows, the AP is "attached" to the interface processor in a logical sense only. The physical connections are standard bus signals and one interrupt line (which would typically be routed to the AP via an interrupt controller).

Continuing the notion of the logical I/O processor, the AP fetches instructions, and provides the instructions needed to alter the flow of execution, and to perform arithmetic, logic and data transfer operations within the peripheral subsystem.

INTERFACE PROCESSOR

The IP completes the logical I/O processor by providing data paths between the peripheral subsystem and the main system, and by providing functions that effectively extend the AP's instruction set so that software running on the logical I/O processor can operate in the main system. Since these facilities are software-controlled, they are discussed in the next section.

As Figure 7 shows, the IP presents both a peripheral subsystem bus interface and a standard iAPX 432 Processor Packet bus interface. By bridging the two buses, the IP provides the hardware link that permits data to flow under software control between the main system and the peripheral subsystem.

The IP connects to the main system in exactly the same way as a GDP. Thus, in addition to being able to access main memory, the IP supports other iAPX 432 hardware-based facilities, including processor communication, the alarm signal and functional redundancy checking.

The IP is connected to the peripheral subsystem bus as if it were a memory component; it occupies a block of memory addresses up to 64K bytes long. Like a memory, the IP behaves passively within the peripheral subsystem (except as noted below). It is driven by peripheral subsystem memory references that fall within its address range.

While the IP generally responds like a memory component, it also provides an interrupt request signal. The interface processor uses this line to notify its AP that an event has occurred which requires its attention.

To summarize, the AP and the IP interact with each other by means of address references generated by the AP and interrupt requests generated by the IP. Since the IP responds to memory references, other active peripheral subsystem agents (bus masters), such as DMA controllers, may obtain access to main system memory via the IP.

Software

IP CONTROLLER

The peripheral subsystem interface is managed by software, referred to as the IP controller. The IP controller executes on the AP and uses the facilities provided by the AP and the IP to control the flow of data between the main system and the peripheral subsystem.

While there are no actual constraints on the structure of the IP controller, organizing it as a collection of tasks running under the control of a multitasking operating system (such as an RMX-80 or iRMX-86 operating system) can simplify software development and modification. This type of organization supports asynchronous message-based communication within the IP controller, similar to the iAPX 432's intrinsic interprocessor communication facility. Extending this approach to the device interface as well results in a consistent, system-wide communication model. However, communication within the IP controller and between the IP controller and device interfaces, is completely application-defined. It may also be implemented via synchronous procedure calls, with "messages" being passed in the form of parameters.

However it is structured, the IP controller interacts with the main system through facilities provided by the interface processor. There are three of these facilities: execution environments, windows, and functions.

EXECUTION ENVIRONMENTS

The IP provides an environment within the main system that supports the operation of the IP controller in that system. This environment is embodied in a set of system objects that are used and manipulated by the IP. At any given time, the IP controller is represented in main memory by a process object and a context object. Like a GDP, the IP itself is represented by a processor object. Representing the IP and its controlling software like this creates an execution environment that is analogous to the environment of a process running on a GDP. This environment provides a standard framework for addressing, protection and communication within the main system.

Like a GDP, an IP actually supports multiple process environments. The IP controller selects the environment in which a function is to be executed. This permits, for example, the establishment of separate environments corresponding to individual device interface tasks in the peripheral subsystem. If an error occurs while the IP controller is executing a function on behalf of one device interface, that error is confined to the associated process, and processes associated with other device interfaces are not affected.

WINDOWS

Every transfer of data between the main system and a peripheral subsystem is performed with the aid of an IP window. A window defines a correspondence, or mapping, between a subrange of peripheral subsystem memory addresses (within the range of addresses occupied by the IP) and an object in main system memory (see Figure 8). When an agent in the peripheral subsystem (e.g., the IP controller) reads a local windowed address, it obtains data from the associated object; writing into a windowed address transfers data from the peripheral subsystem to the windowed object.

The action of the IP, in mapping the peripheral subsystem address to the main system object, is transparent to the agent making the reference; as far as it is concerned, it is simply reading or writing local memory.

Since a window is referenced like local memory, any individual transfer may be between an object and local memory, an object and a processor register, or an object and an I/O device. The

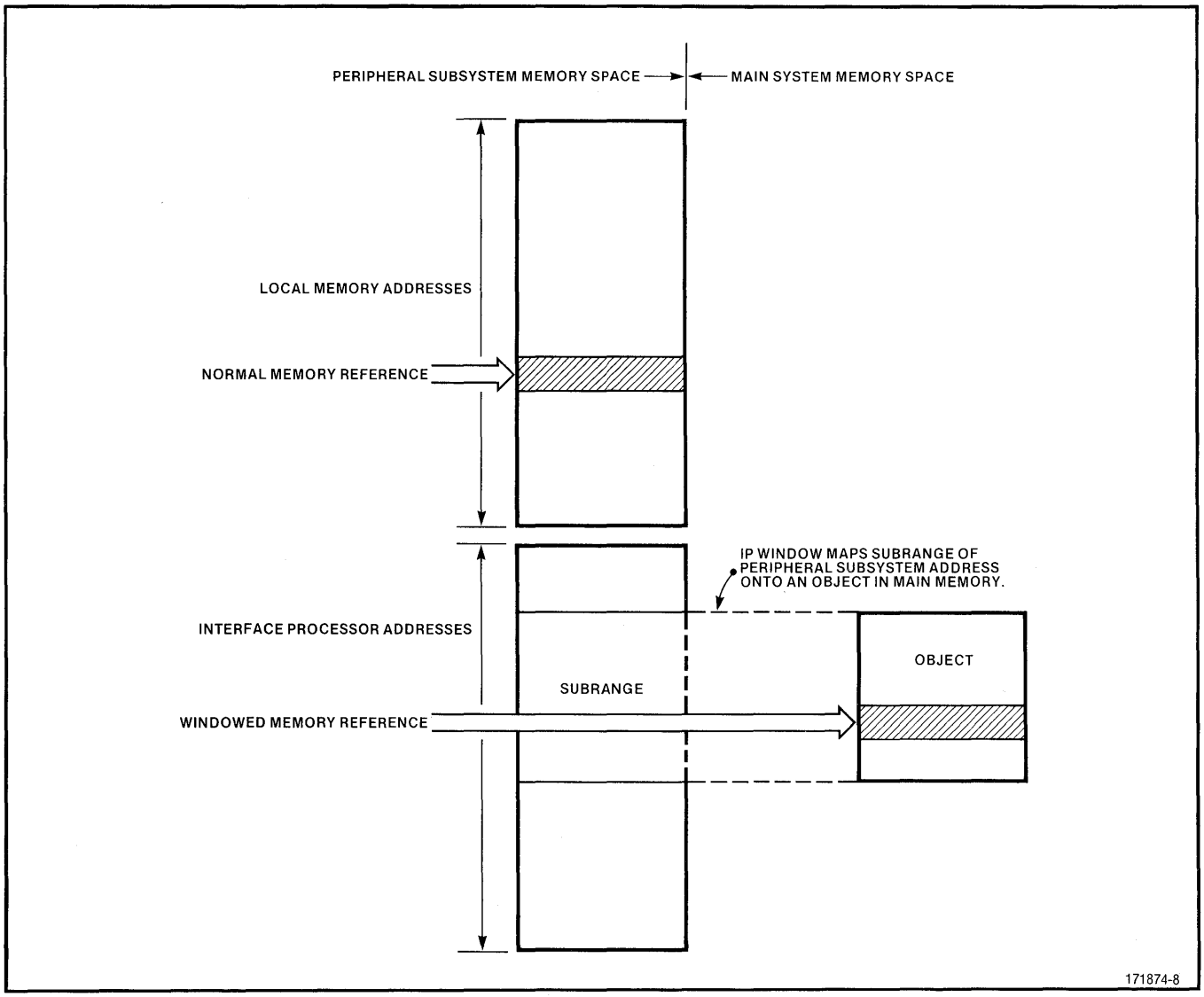


Figure 8. Interface Processor Window

latter may be appealing from the standpoint of "efficiency," but it should be considered with caution. Using a window to directly "connect" an I/O device and an object in main memory has the undesirable effect of propagating the real-time constraints imposed by the device beyond the subsystem boundary into the main system. It may seriously complicate error recovery as well. Finally, since there is a finite number of windows, most applications will need to manage them as scarce resources that will not always be instantly available. This means that at least some I/O device transfers will have to be buffered in local memory until a window becomes available. It may be simplest to buffer all I/O device transfers and use the windows to transfer data between local memory and main system memory.

There are four IP windows that may be mapped onto four different objects. The IP controller may alter the windows during execution to map different subranges and objects. References to windowed subranges may be interleaved and may be driven by different processors in the peripheral subsystem. For example, the AP and a DMA controller may be driving transfers concurrently, subject to the same bus arbitration constraints that would apply if they were accessing local memory.

FUNCTIONS

A fifth window provides the IP controller with access to the IP's function facility. By writing operands and an opcode into predefined locations in this window's subrange, the IP controller requests the IP to execute a function on its behalf. This procedure is very similar to writing commands and data to a memory-mapped peripheral controller (e.g., a floppy disk controller). Upon completion of the function, the IP provides status information that the IP controller can read through the window. The IP can perform transfer requests through the other four windows while it is executing a function.

The IP's function set permits the IP controller to:

- alter windows;
- exchange messages with GDP processes via the standard 432 communication facility;
- manipulate objects.

These functions may be viewed as instruction set extensions to the AP, which permit the IP controller to operate in the main system. The combination of the IP's function set and windows, the AP's instruction set, and possibly additional facilities provided by a peripheral subsystem operating system, permits the construction of powerful IP controllers that can relieve the main system of much I/O-related processing. At the same time, by utilizing only a subset of the available IP functions, relatively simple IP controllers can also be built (in cases where this approach is more appropriate).

SUPPLEMENTARY INTERFACE PROCESSOR FACILITIES

The preceding sections describe the IP as it is used most of the time. The IP provides two additional capabilities that are typically used less frequently, and only in exceptional circumstances. These are physical reference mode and interconnect access.

Physical Reference Mode

The IP normally operates in logical reference mode; this mode is characterized by its object-oriented addressing and protection system. There are times when logical referencing is impossible because the objects used by the hardware to perform logical-to-physical address development are absent (or, less likely, are damaged). In these situations, the IP can be used in physical reference mode.

In physical reference mode, the IP provides a reduced set of functions. Its windows operate as in logical reference mode, except that they are mapped onto memory segments (rather than objects) that are specified directly with 24-bit addresses. In this respect, physical reference mode is similar to traditional computer addressing techniques.

Physical reference mode is most often employed during system initialization to load binary images of objects from a peripheral subsystem into main memory. Once the required object images are available, processors can begin normal logical reference mode operations.

Interconnect Access

In addition to memory, the iAPX 432 architecture defines a second address space called the processor-memory interconnect address space. One of the IP windows is software-switchable to

either space. In logical reference mode, the interconnect space is addressed in the same object-oriented manner as the memory space, with the IP automatically performing the logical-to-physical address development. In physical reference mode, the interconnect space is addressed as an array of 16-bit registers, with a register selected by a 24-bit physical address.

iAPX 432 INFORMATION STRUCTURE

The following information describes the requirements placed on the logical structure of the iAPX 432 hardware environment. These requirements are concerned directly with the constraints of physical memory, the type of data transferred, and the structure of the data types. These requirements are common to the iAPX 432 family of processors. (Any pin notations called out in this information are described in the 43203 Pin Description section of this data sheet).

Any 432 processor in the system can access all the contents of physical memory. This section describes how information is represented and accessed.

Memory

The iAPX 432 implements a two-level memory structure. The software system exists in a segmented environment in which a logical address specifies the location of a data item. The processor automatically translates this logical address into a physical address for accessing the value in physical memory.

Physical Addressing

Logical addresses are translated by the processor into physical addresses. Physical addresses are transmitted to memory by a processor to select the beginning byte of a memory value to be referenced. A physical address is 24 binary bits in length. This results in a maximum physical memory of 16 Megabytes.

Data Formats

When a processor executes the instructions of an operation within a context, operands found in the logical address space of the context may be manipulated. An individual operand may occupy one, two, four, eight, or ten bytes of memory (byte, double byte, word, double word, or extended word, respectively). All operands are referenced by a logical address as described above. The displacement in such an address is the displacement in bytes from the base address of the data segment to the first byte of the operand. For operands consisting of multiple bytes, the address locates the low-order byte while the higher-order bytes are found at the next higher consecutive addresses.

DATA REPRESENTATION

An iAPX 432 convention has been adopted for representing data operands stored in memory. The bits in a field are numbered by increasing numeric significance, with the least-significant bit shown on the right. Increasing byte addresses are shown from right to left. Examples of the five basic data lengths used in the iAPX 432 system are shown in Figure 9.

DATA POSITIONING

The data operand types shown in Figure 9 may be aligned on an arbitrary byte boundary within a data segment. Note that more efficient system operation may be obtained when multi-byte data structures are aligned on double-byte boundaries (if the memory system is organized in units of double bytes).

Requirements of an iAPX 432 Memory System

The multiprocessor architecture of the iAPX 432 places certain requirements on the operation of the memory system to ensure the integrity of data items that can potentially be accessed simultaneously. Indivisible read-modify-write (RMW) operations to both double-byte and word operands in memory are necessary for manipulating system objects. When an RMW-read is processed for a location in memory, any other RMW-reads from that location must be held off by the memory system until an RMW-write to that location is received (or until an RMW timeout occurs). Note that while the memory system is awaiting the RMW-write, any other types of reads and writes are allowed. Also, for ordinary reads and writes of double-byte or longer operands, the memory system must ensure the entire operand

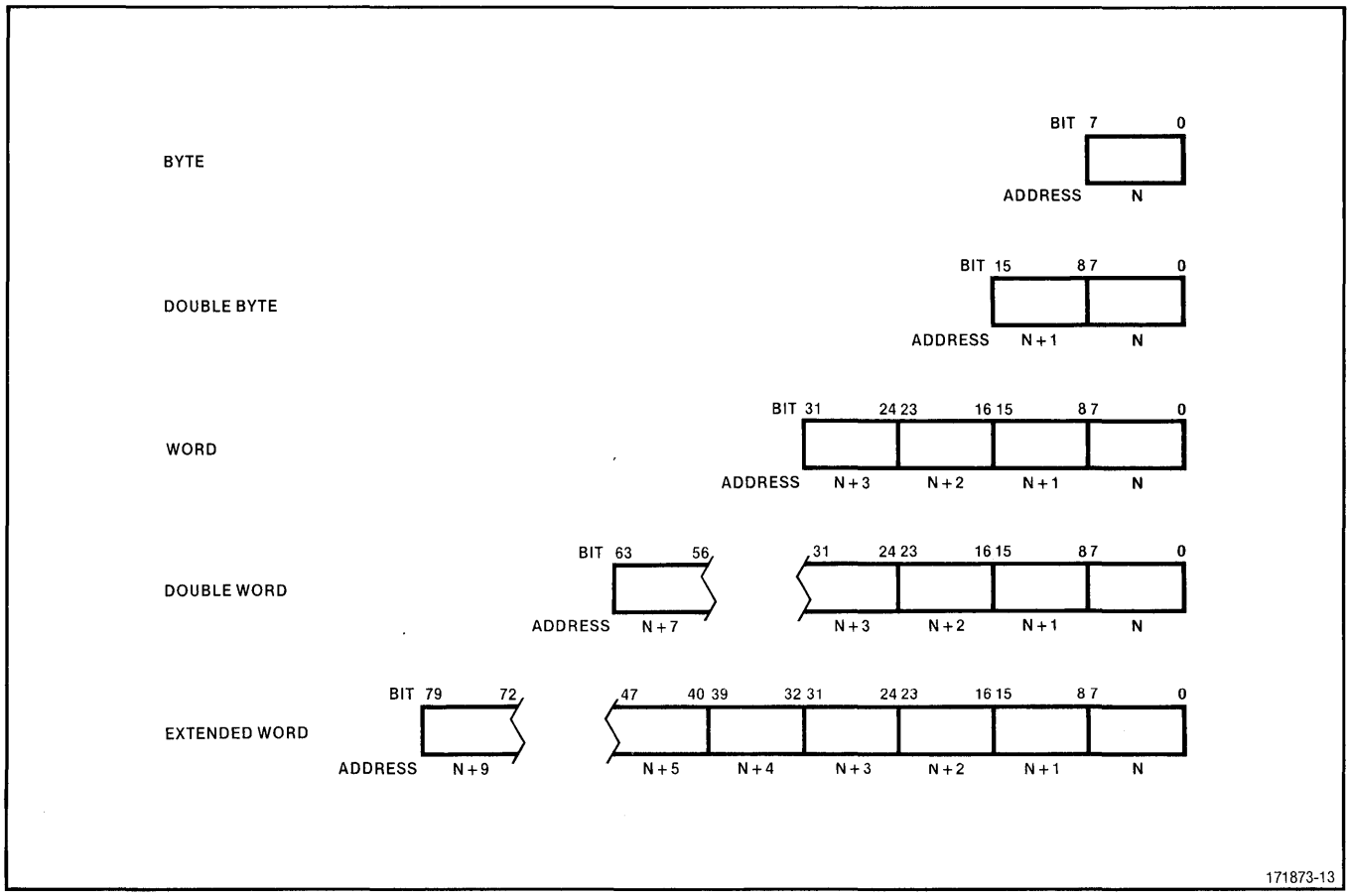


Figure 9. Basic iAPX 432 Data Lengths

has been either read or written before beginning to process another access to the same location; e.g., if two simultaneous writes to the same location occur, the memory system must ensure that the set of locations used to store the operand does not get changed to some interleaved combination of the two written values.

iAPX 432 HARDWARE ERROR DETECTION

iAPX 432 processors include a facility to support the hardware detection of errors by functional redundancy checking (FRC). At initialization time, each iAPX 432 processor is configured to operate as either a master or a checker processor (Figure 10). A master operates in the normal manner. A checker places all output pins that are being checked into a high-impedance state. Thus, those pins which are to be checked on a master and checker are parallel-connected, pin for pin, such that the checker is able to compare its master's output pin values with its own. Any comparison error causes the checker to assert HERR.

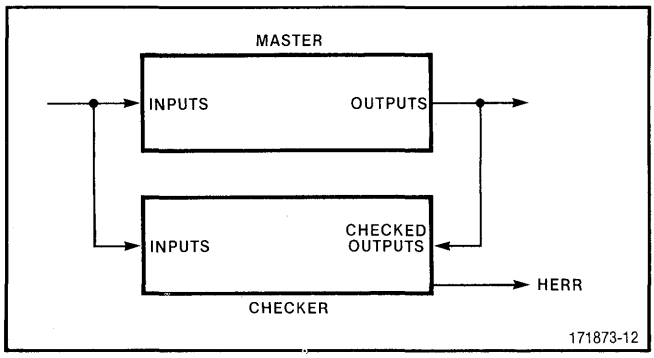


Figure 10. Hardware Error Detection

PROCESSOR PACKET BUS DEFINITION

This section describes and defines the significance of the 19 signal lines that make up the Processor Packet bus, and the general scheme by which timing relationships on these lines are derived. Although this section defines all legal bus activities, the processors do not necessarily perform all allowed activities. Slaves to the Processor Packet bus must support all state transitions to ensure compatibility.

The Processor Packet bus consists of 3 control lines:

- Processor Packet bus Request (PRQ),
- Enable Buffers for Output (BOUT),
- Interconnect Status (ICS).

This bus also includes sixteen 3-state Address-Control-Data lines (ACD15 through ACD0). PRQ has two functions whose use depends upon the application; i.e., PRQ either indicates the first cycle of a transaction on the Processor Packet bus or the cancellation of a transaction initiated in the previous cycle. Of the three control lines, BOUT has the simplest function, serving as a direction control for buffers in large systems requiring more electrical drive than the processor components can provide. The ICS signal has significance pertaining to one of three different system conditions and depends on the state of the Processor Packet bus transaction. The processor interprets the ICS input as an indication of one of the following:

- Whether or not an interprocessor communication (IPC) is waiting,
- Whether or not the slave requires more time to service the processor's request,
- Whether or not a bus ERROR has occurred.

The Address/Control/Data lines emit output specification information to indicate the type of cycle being initiated, e.g., addresses, data to be written, or control information. They also receive data returned to the processor during reads. Details of the ACD line operation and the associated control lines are summarized below.

ACD15-ACD0 (Address/Control/Data)

As shown in Figure 11, the first cycle, (T1 or Tvo) of a Processor Packet bus transaction (indicated by the rising edge of PRQ), the high-order 8 ACD bits (ACD15...ACD8) specify the type of the current transaction. In this first cycle, the low-order ACD bits (ACD7...ACD0) contain the least-significant eight bits of the 24-bit physical address.

During the subsequent cycle (T2), the remainder of the address is present on the ACD pins (aligned such that the most significant byte of the address is on ACD15 through ACD8, the mid-significant byte on ACD7 through ACD0). If PRQ is asserted during T2, the access is cancelled and the ACD lines are not defined.

During the third cycle (T3 or Tw) of a Processor Packet bus transaction the processor presents a high impedance to the ACD lines for read transactions and asserts write data for write transactions.

Once the bus has entered T3 or Tv, the sequence of state transactions depends on the type of cycle requested during the preceding T1 or Tvo. Accesses ranging in length from 1 to 32 bytes may be requested (see Table 1). If a transfer of more than one double byte has been requested, it is necessary to enter T3 for every double-byte that is transferred. The processor may simply enter T3 or it may first enter Tw for any number of cycles (as dictated by ICS).

After all data is transferred, the processor enters either Tv or Tvo. Tvo can be entered only when the internal state of execution is such that the processor is prepared to accomplish an immediate write transfer (overlapped access). During Tvo, the ACD lines contain address and specification information aligned in the same fashion as in T1. If the processor does not require an overlapped access, the bus state moves to Tv (the ACD lines will be high impedance). After Tv, a new bus cycle can be started with T1, or the processor may enter the idle state(Ti).

ICS (Interconnect Status)

ICS has three possible interpretations depending on the state of the bus transaction (see Table 2). Notice that under most conditions ICS has IPC significance for more than one cycle. It is important to note that a valid low during any cycle with IPC significance will signal the processor that an IPC or reconfiguration request has been received. An iAPX 432 processor is required to record and service only one IPC or reconfiguration request at a time. Logic in the interconnect system must record and sequence multiple (possibly simultaneous) IPC occurrences and reconfiguration requests to the processor. Thus the logic that forms ICS must accomodate global and local IPC arrivals and requests for reconfiguration as individual events:

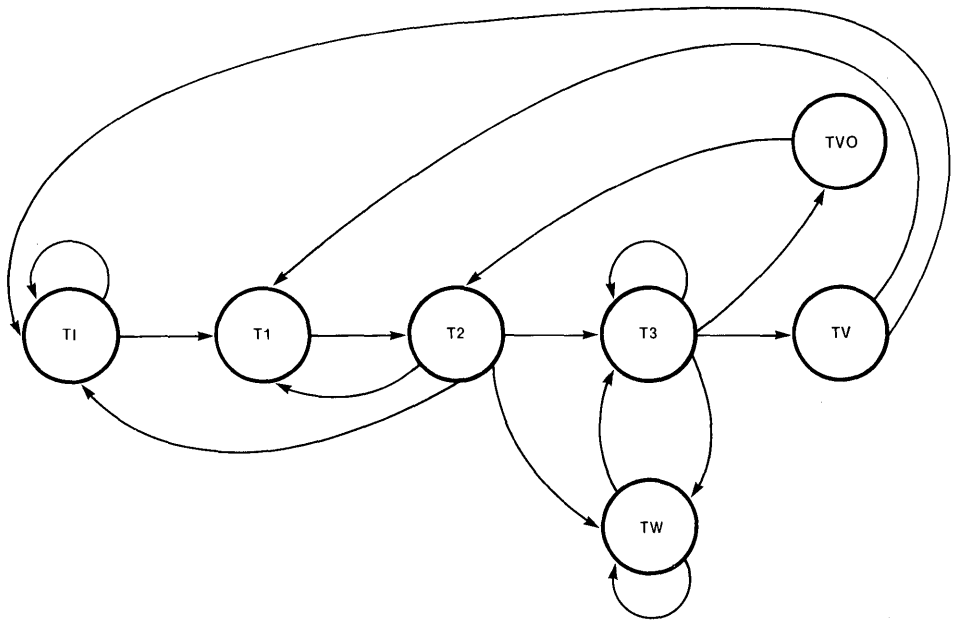
1. Assert IPC significance on ICS for the arrival of an IPC or reconfiguration request.

2. When the iAPX 432 processor reads interconnect address register 2, it will respond to one of the status bits for the IPC or reconfiguration request signalled on ICS in the following order:

   Bit 2 (1=reconfigure, 0=do not reconfigure)

   Bit 1 (1=global IPC arrived, 0=no global IPC)

   Bit 0 (1=local IPC arrived, 0=no local IPC)



171873-14

Initial State	Next State	Trigger
Ti	TI Ti	Bus cycle desired No bus cycle desired
TI	T2	Unconditional
T2	T3 Tw TI Ti	ICS high ICS low Cancelled, Access Pending Cancelled, No Access Pending
T3	T3 Tw Tv Tvo	Additional transfer required, ICS high Additional transfer required, ICS low All transfers completed, no overlapped access Current write with overlapped access
Tv	Ti TI	No access pending Access pending
Tvo	T2	Unconditional
Tw	Tw T3	ICS low ICS high

Figure 11. Processor Packet Bus State Diagram

Table 1. ACD Specification Encoding

ACD 15	ACD 14	ACD 13	ACD 12 ACD 11 ACD 10	ACD 9 ACD 8
Access	**Op**	**RMW**	**Length**	**Modifiers**
0 - Memory	0 - Read	0 - Nominal	000 - 1 Byte 001 - 2 Bytes 010 - 4 Bytes 011 - 6 Bytes 100 - 8 Bytes 101 - 10 Bytes 110 - 16 Bytes* 111 - 32 Bytes* * Not implemented	ACD 15 = 0: 00-Inst Seg Access 01-Stack Seg Access 10-Context Ctl Seg Access 11-Other ACD 15 = 1: 00-Reserved 01-Reserved 10-Reserved 11-Interconn Register
1 - Other	1 - Write	1 - RMW		

3. The logic in the interconnect system must clear the highest order status bit that was serviced by the iAPX 432 processor, and if additional IPC information has arrived, the interconnect system logic must signal an additional IPC indication to the iAPX 432 processor.

   The interconnect system must signal the second IPC by raising ICS high for at least one cycle and then setting ICS low for at least one cycle during IPC significance time.

Table 2. ICS Interpretation

	Level		State
	High	Low	
IPC	None	Waiting	Ti, T1, T2*
Stretch	Don't	Stretch	T3, Tw
Err	Bus Error	No Error	Tv, Tvo

* ICS has no significance in a cycle following a T2 where PRQ is asserted (cancelled access) or in any cycle during which CLR/ is asserted.

PRQ (Processor Packet Bus Request)

PRQ is normally low and can go high only during T1, T2 and Tvo. High levels during Tvo and T1 indicate the first cycle of an access. A high level during T2 indicates that the current cycle is to be cancelled. See Table 3.

Table 3. PRQ Interpretation

State	PRQ	Condition
Ti	0	Always
T1	1	Initiate access
T2	0	Continue access
	1	Cancel access
T3	0	Always
Tw	0	Always
Tv	0	Always
Tvo	1	Initiate overlapped access

BOUT (Enable Buffers for Output)

BOUT is provided to control external buffers when they are present. Table 4 and Figures 12 through 16 show its state under various conditions.

Processor Packet Bus Timing Relationships

All timing relationships on the processor packet bus are derived from a simple scheme and related to Table 5. Each timing diagram shown in the following pages (Figures 12 through 17) provides a separate table illustrating the various system states during the cycle. This approach to transfer timing was designed to allow maximum time for the transfer to occur and yet guarantee hold time. The solid lines in Figure 18 show the state transitions initiated by the IP.

Any agent connected to the processor packet bus is recognized as either a processor or a slave. Examples of processors are the GDP and the IP. A memory system provides an example of a slave.

In all tranfers between a processor and a slave, the data to be driven are clocked three-quarters of a cycle before they are to be sampled. This allows adequate time for the transfer and ensures sufficient hold time after sampling. The BOUT timing is unique because BOUT is intended as a direction control for external buffers.

Detailed set-up and hold times depend on the processor implementation and can be found in the ac characteristics section.

Table 4. BOUT Interpretation

BOUT	Always High	Low-to-High Transition or Low	High-to-Low Transition or Low	High-to-Low Transition or High
Write	T1, T2, T3, Tw, Tvo	Ti	None	Tv
Read	T1, T2	Ti, Tv	T3, Tw	None

Table 5. iAPX 432 Component Signaling Scheme

	Processor		Slave	
Inputs Sampled	ACD:	↓CLKA	All:	↑CLKB
	Others:	↑CLKA		
Outputs Driven	All (except BOUT):	↓CLKA	ACD:	↓CLKB
			Others:	↑CLKB
	BOUT:	↑CLKA		

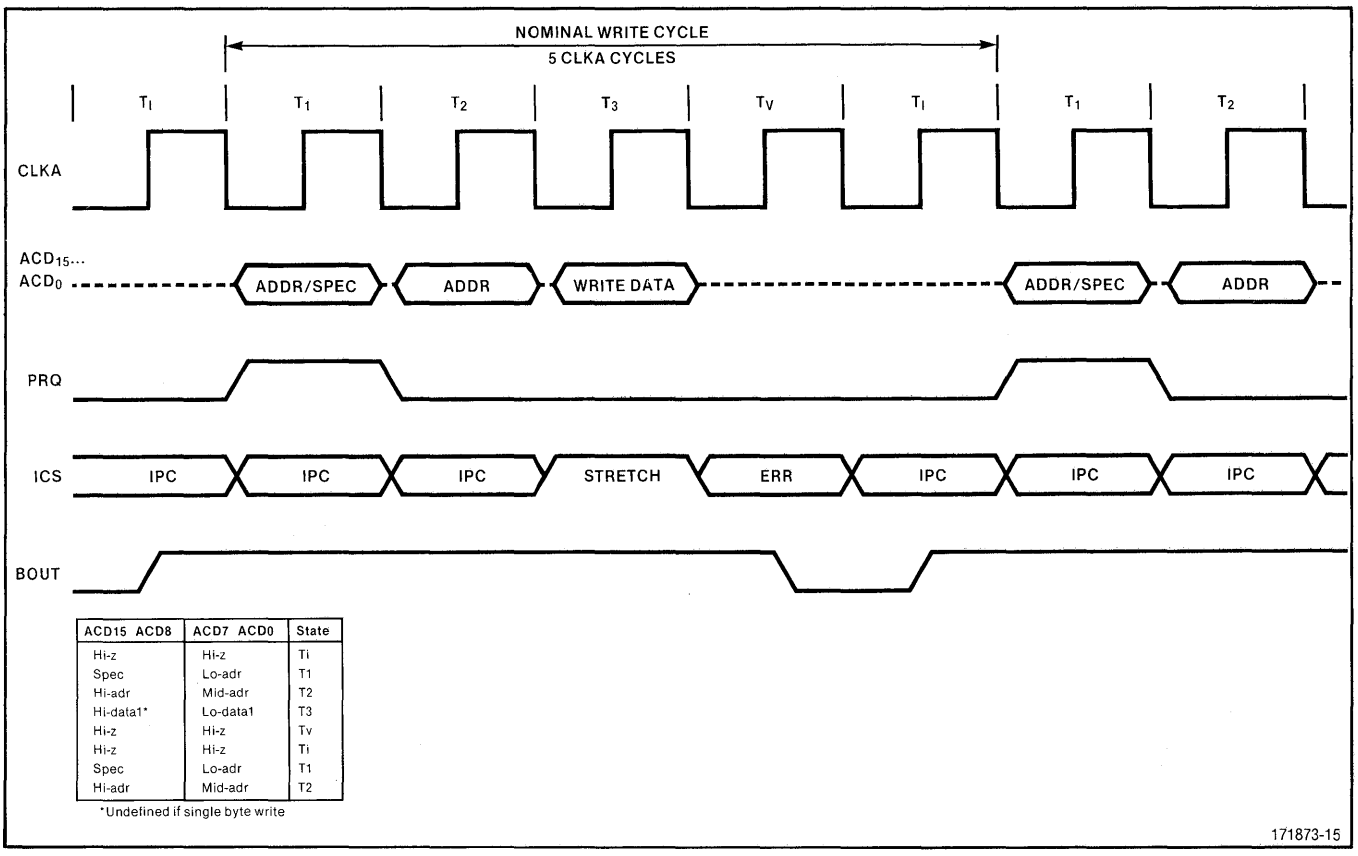
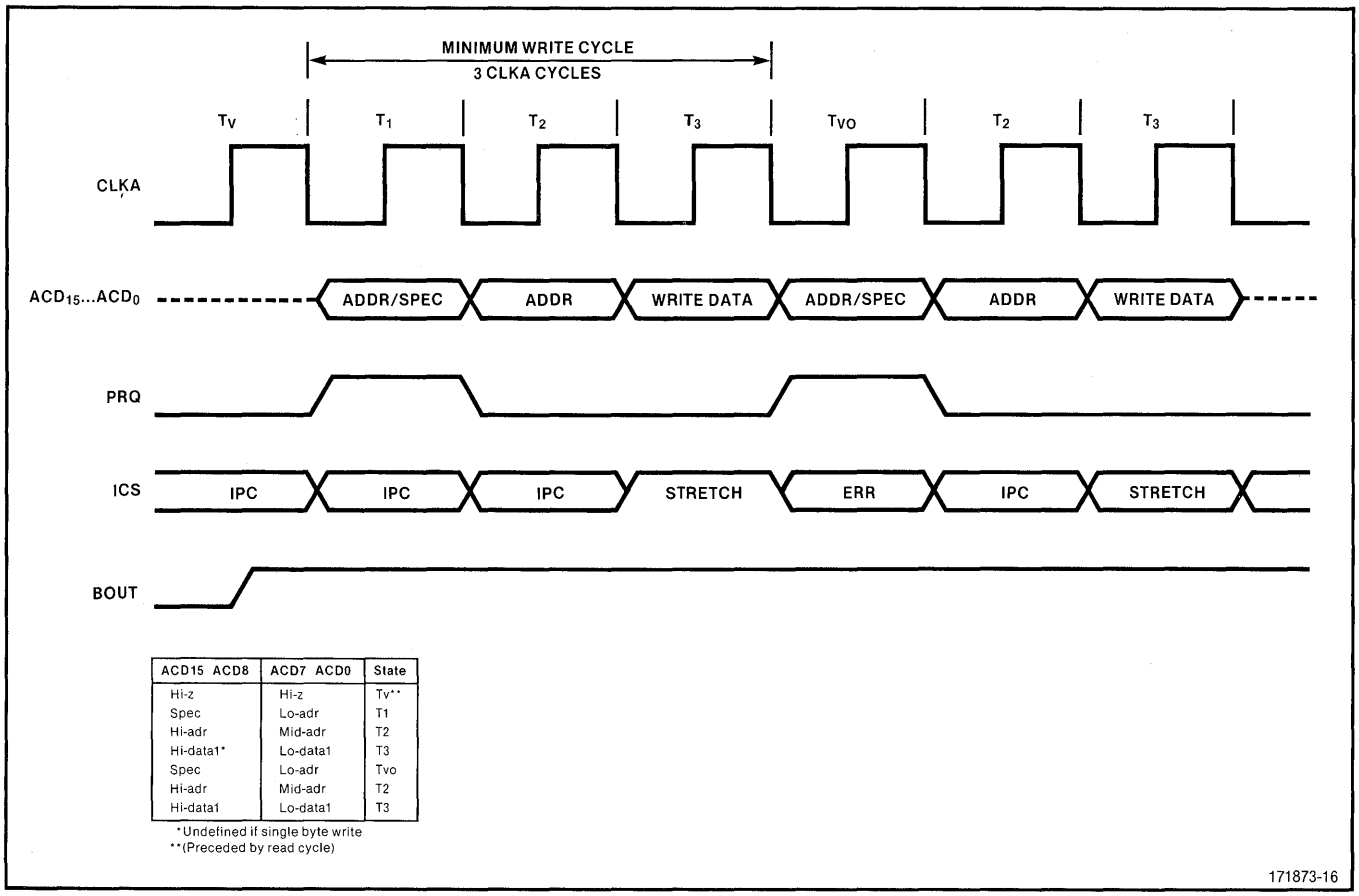


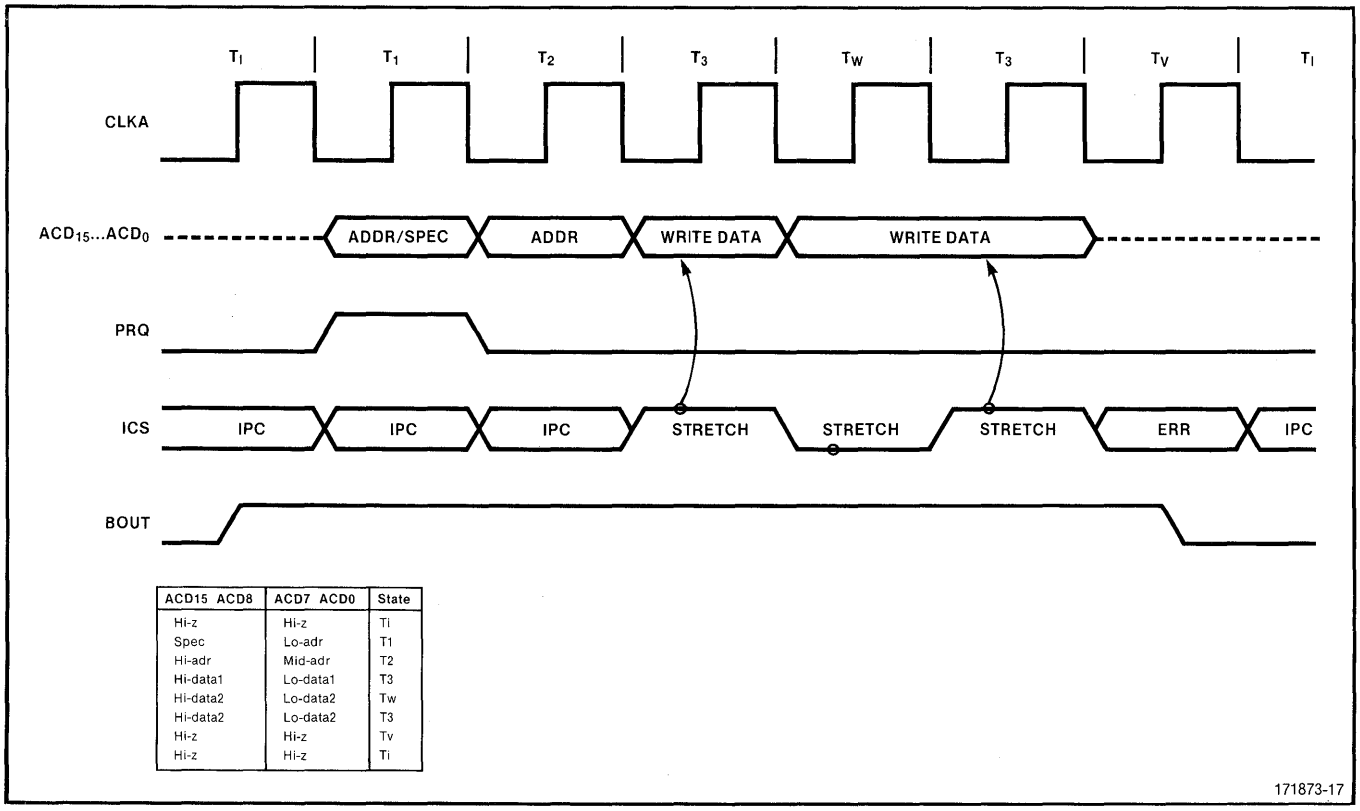
Figure 12. Nominal Write Cycle Timing



ACD15 ACD8	ACD7 ACD0	State
Hi-z	Hi-z	Tv**
Spec	Lo-adr	T1
Hi-adr	Mid-adr	T2
Hi-data1*	Lo-data1	T3
Spec	Lo-adr	Tvo
Hi-adr	Mid-adr	T2
Hi-data1	Lo-data1	T3

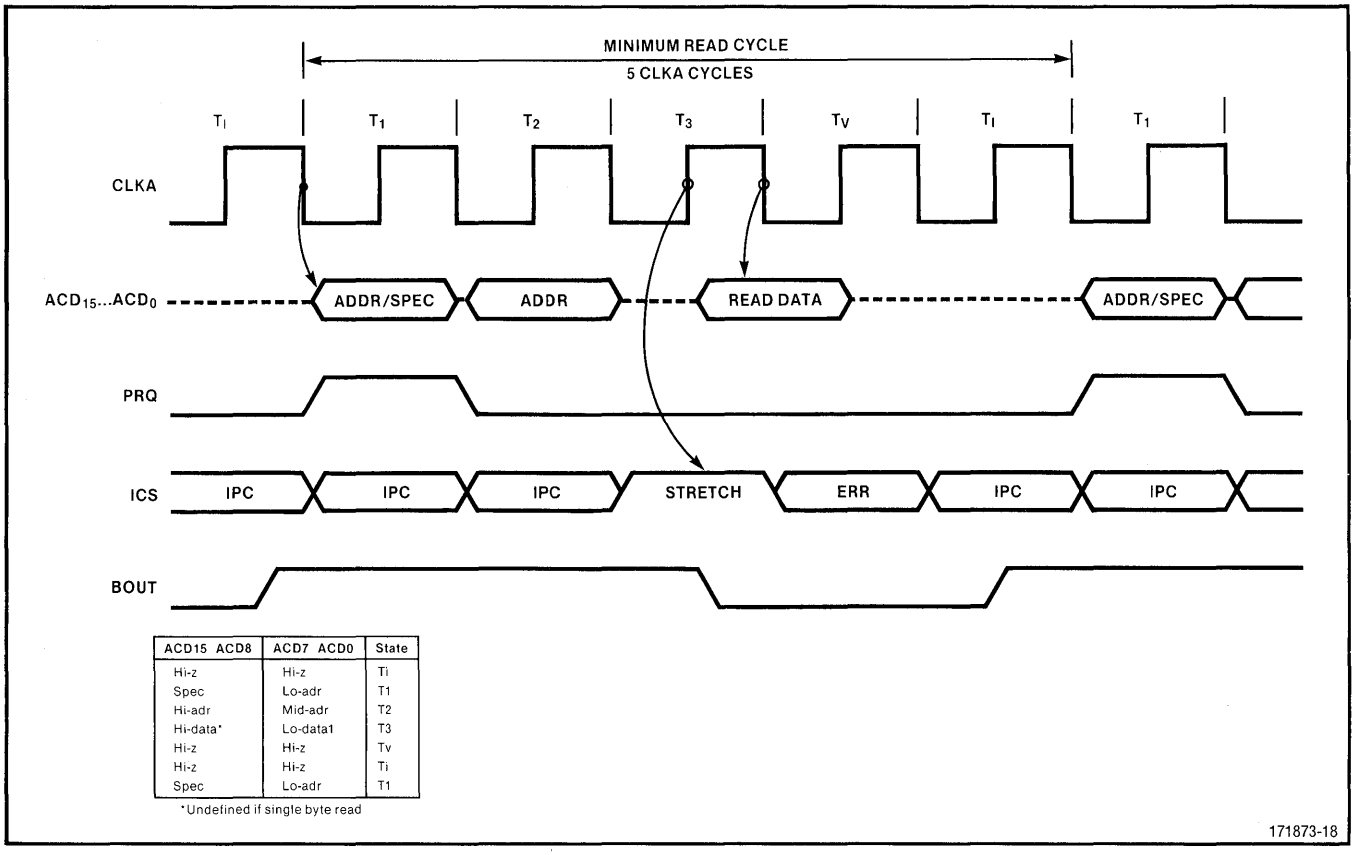
*Undefined if single byte write
**(Preceded by read cycle)

Figure 13. Minimum Write Cycle Timing



ACD15 ACD8	ACD7 ACD0	State
Hi-z	Hi-z	Ti
Spec	Lo-adr	T1
Hi-adr	Mid-adr	T2
Hi-data1	Lo-data1	T3
Hi-data2	Lo-data2	Tw
Hi-data2	Lo-data2	T3
Hi-z	Hi-z	Tv
Hi-z	Hi-z	Ti

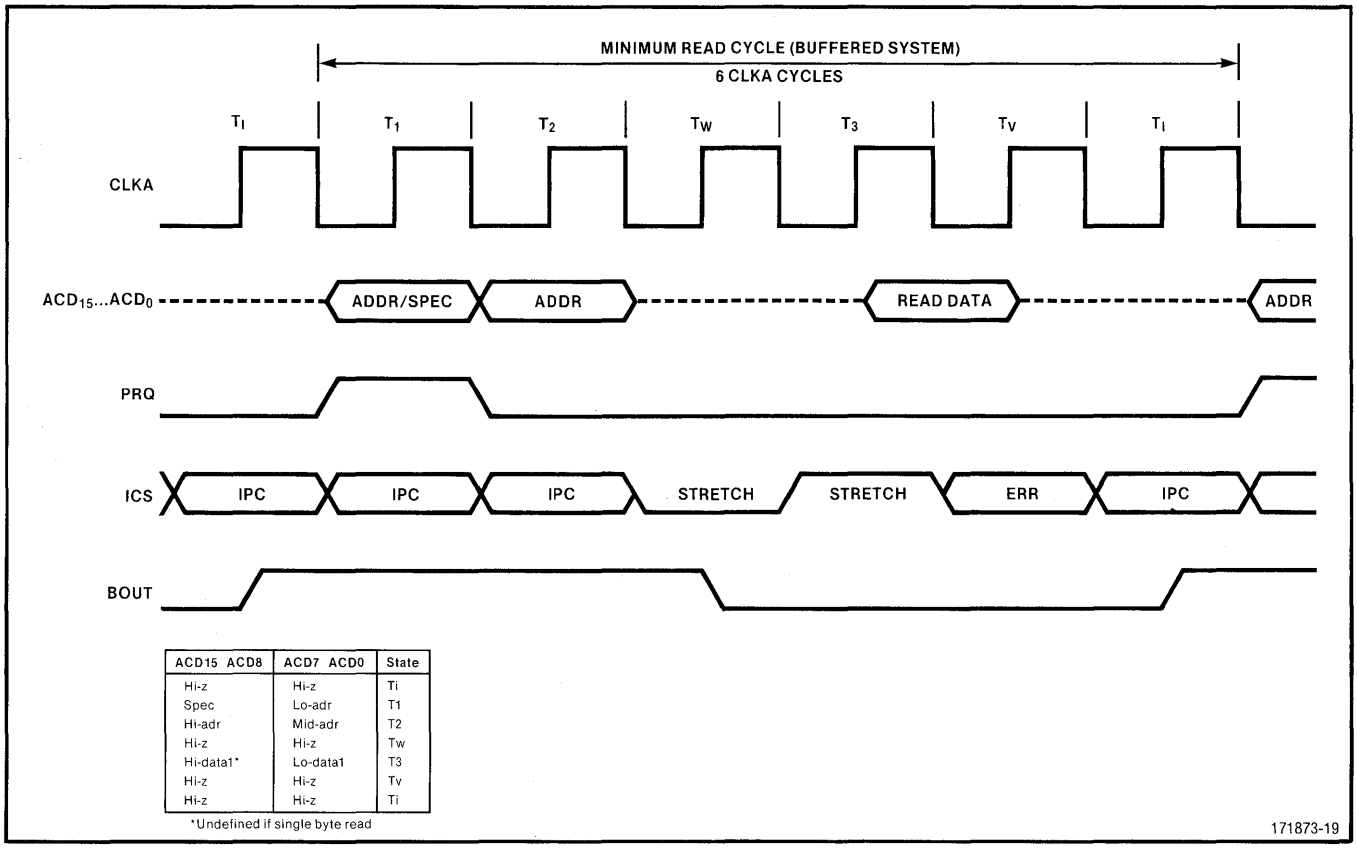
Figure 14. Stretched Write Cycle Timing



ACD15 ACD8	ACD7 ACD0	State
Hi-z	Hi-z	Ti
Spec	Lo-adr	T1
Hi-adr	Mid-adr	T2
Hi-data*	Lo-data1	T3
Hi-z	Hi-z	Tv
Hi-z	Hi-z	Ti
Spec	Lo-adr	T1

*Undefined if single byte read

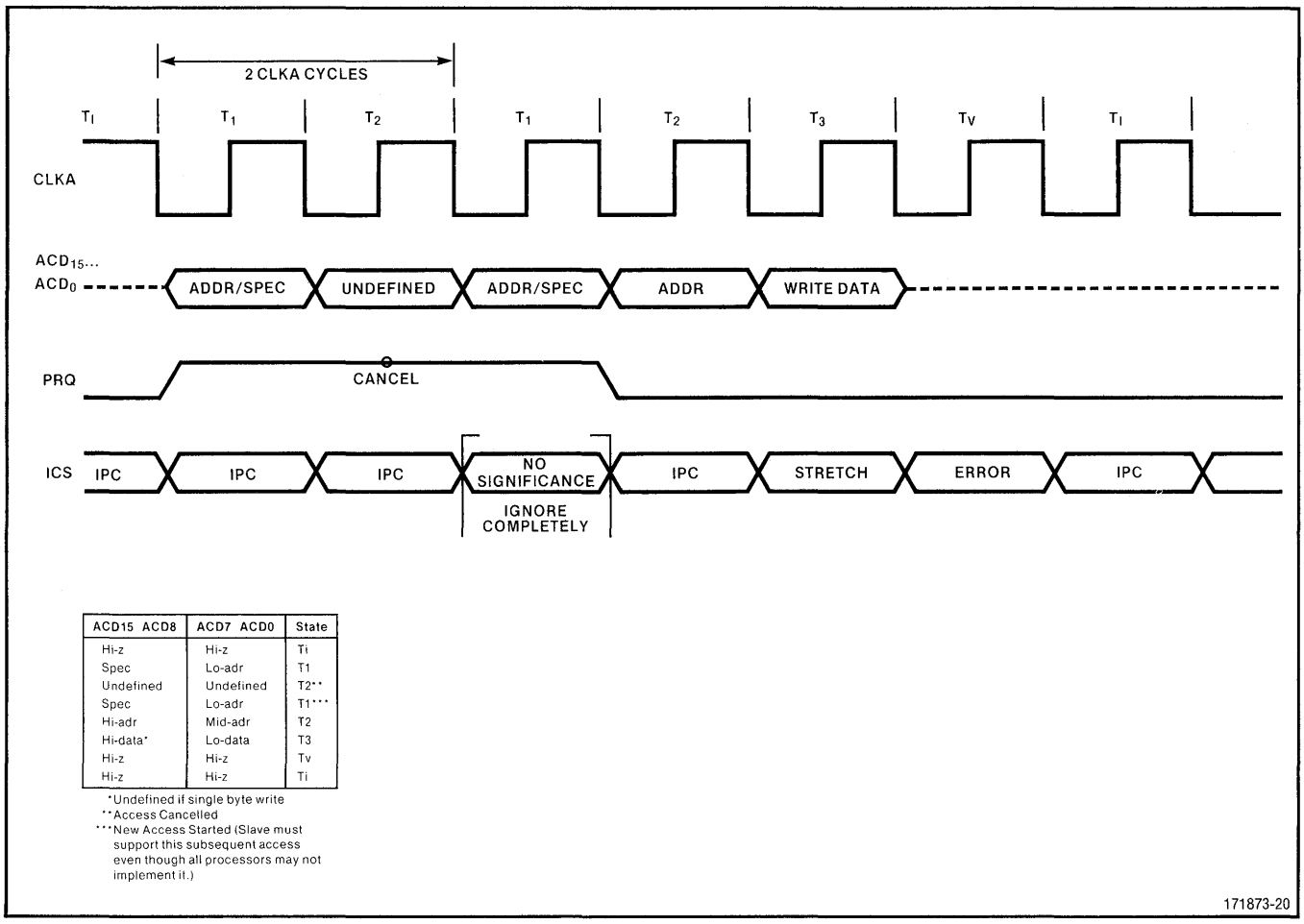
Figure 15. Minimum Read Cycle (Not Buffered)



ACD15 ACD8	ACD7 ACD0	State
Hi-z	Hi-z	Ti
Spec	Lo-adr	T1
Hi-adr	Mid-adr	T2
Hi-z	Hi-z	Tw
Hi-data1*	Lo-data1	T3
Hi-z	Hi-z	Tv
Hi-z	Hi-z	Ti

*Undefined if single byte read

Figure 16. Minimum Read Cycle (Buffered System)

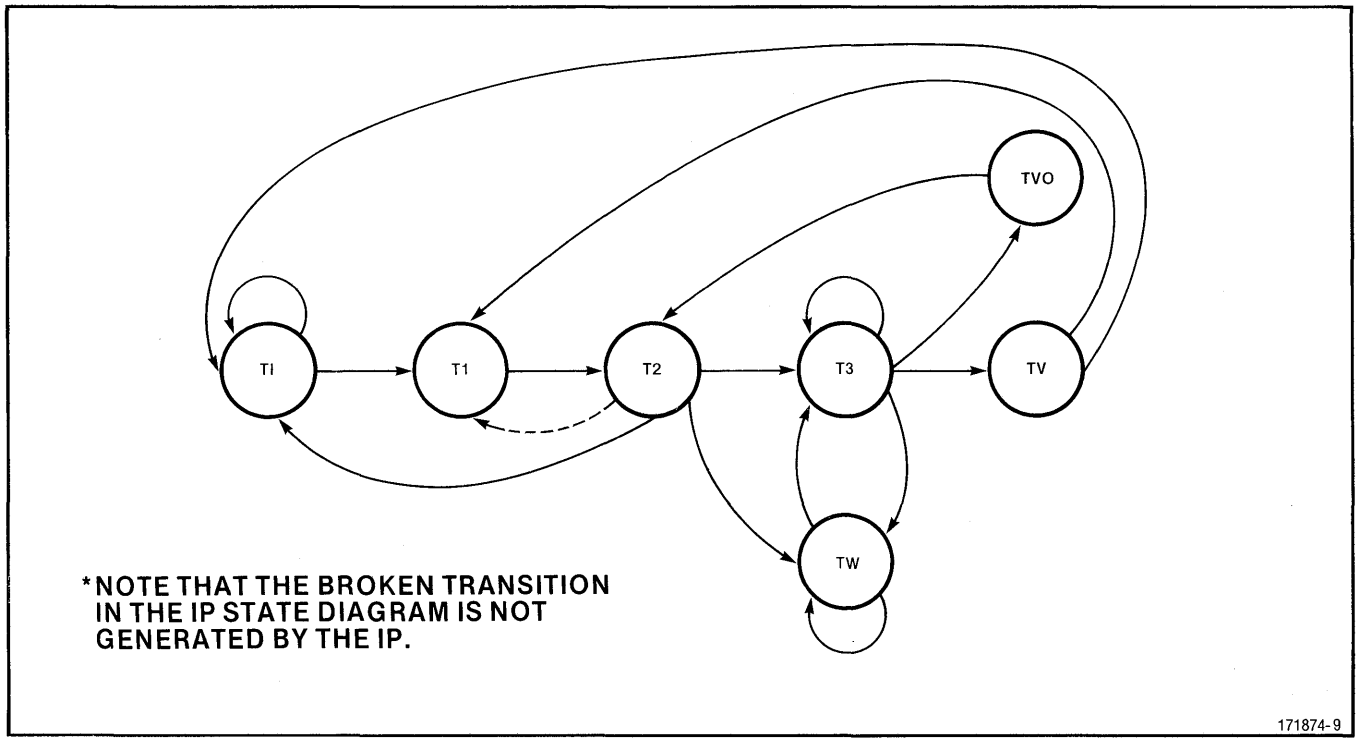


ACD15 ACD8	ACD7 ACD0	State
Hi-z	Hi-z	Ti
Spec	Lo-adr	T1
Undefined	Undefined	T2**
Spec	Lo-adr	T1***
Hi-adr	Mid-adr	T2
Hi-data*	Lo-data	T3
Hi-z	Hi-z	Tv
Hi-z	Hi-z	Ti

*Undefined if single byte write
**Access Cancelled
***New Access Started (Slave must support this subsequent access even though all processors may not implement it.)

171873-20

Figure 17. Minimum Faulted Access Cycle



171874-9

Figure 18. IP State Diagram

Table 6. iAPX 43203 Interface Processor Pin Summary

432 System Side			
Pin Group	**Pin Name**	**Direction**	**Hardware Error Detection**
PROCESSOR PACKET BUS GROUP	ACD15...ACD0	I/O	X
	PRQ	O	X
	ICS	I	
	BOUT	O	
SYSTEM GROUP	ALARM /	I	
	FATAL/	O (I at Initialization)	
	CLR/	I	
	PCLK/	I	
	INIT/	I	
CLOCK GROUP	CLKA	I	
	CLKB	I	
HARDWARE ERROR DETECTION GROUP	HERR	O (I at Initialization)	
Peripheral Subsystem Side			
Pin Group	**Pin Name**	**Direction**	**Hardware Error Detection**
PERIPHERAL SUBSYSTEM BUS GROUP	AD15... AD0	I/O	X
	BHEN/	I	
	CS/	I	
	WR/	I	
PS TIMING GROUP	ALE	I	
	OE	I	
	SYNC	I	
PS BUFFER CONTROL GROUP	DEN/	O	X
PS INTERLOCK GROUP	HLD	O	X
	HDA	I	
PS SYNCHRONIZATION GROUP	XACK/	O	X
	NAK/	O	X
	INH1	O	X
PS INTERRUPT GROUP	INT	O	X
PS RESET GROUP	PSR	O	X

43203 PIN DESCRIPTION

The following section provides detailed information concerning the 43203 pin description. Table 6 lists a summary of all signal groups, signal names and their active states, and whether or not they are monitored by the Hardware Error Detection circuitry.

Processor Packet Bus Group

ACD_{15}—ACD_0 (Address/Control/Data lines, Inputs or Three-state Outputs, high asserted)

The Processor Packet bus Address/Control/Data lines are the basic communication path between the IP and its environment. These pins are used three ways:

- They may indicate control information for bus transactions,
- They may issue physical addresses generated by the IP for an access, or
- They may transfer data (either direction).

When the 43203 is in checker mode, the ACD pins are monitored by the hardware error detection logic and are in the high impedance mode.

PRQ (Processor Packet bus Request, Three-state Output, high asserted)

PRQ is used to indicate the presence of a transaction between the IP and its external environment. Normally low, the PRQ pin is brought high during the same cycle as the first double-byte of address information is being driven onto the ACD pins. PRQ remains high for only one cycle during the access, unless an address development fault occurs. The 43203 will leave PRQ high for a second cycle to indicate the GDP has detected an addressing or segment rights fault in completing address generation. PRQ is checked by the hardware error detection logic. PRQ is in a high impedance state when the 43203 is in checker mode.

ICS (Interconnect Status, Input, high asserted)

ICS is an indication to the 43203 from the bus interface circuitry concerning the status of a bus transaction. The interpretation of the ICS state is dependent upon the present cycle of a bus transaction and may indicate:

- Interprocessor communication (IPC) message waiting,
- Input data invalid,
- Output data not taken,
- Bus error in external environment.

System Group

ALARM/ (Alarm, Input, low asserted)

The ALARM/ input monitors the occurrence of an unusual, system-wide condition such as power failure. ALARM/ is sampled on the rising edge of CLKA.

FATAL/ (Fatal, Output, low asserted) (Master, Input, low asserted)

FATAL/ is asserted by the IP under microcode control when the processor is unable to continue due to various error or fault conditions. Once FATAL/ is asserted, it can only be reset by assertion of INIT/. FATAL/ is not checked by the hardware error detection logic.

When INIT/ is asserted, the FATAL/ pin assumes an input role. Please refer to the INIT/ pin description for a discussion of this function.

Hardware Error Detection Group

HERR (Hardware Error Output, Open Drain Output, high asserted) (Master, Input, low asserted)

HERR is used to signal a discrepancy between a master and a checker (difference between the value internally computed in the checker and that output by the master). The sampling of errors occurs at the most appropriate time for the pin(s) being checked.

HERR is an open drain output which requires an external pullup resistor. Nominally the output is held low. HERR is released upon the detection of discrepancy. The timing of HERR depends on the source of the error. Once HERR is high it will remain high until external logic forces it to go low again. When HERR goes low again, the present HERR error condition is cleared and HERR is immediately capable of detecting and signaling another error.

When INIT/ is asserted, the HERR pin assumes an input role. Please refer to the INIT/ pin description for a discussion of this function.

PCLK/ (Processor Clock, Input, low asserted)

Assertion of PCLK/ for one clock cycle causes the system timer in the IP to decrement. Assertion of PCLK/ for two or more cycles causes the system timer to be reset. PCLK/ must be unasserted for at least 10 clock cycles before being asserted again.

CLR/ (Clear, Input, low asserted)

Assertion of CLR/ results in a microprogram trap which causes the IP to immediately terminate any bus transactions or internal operations which may be in progress at the time, reset to a known state, assert FATAL/, and await an IPC (which resets the IP to the same state as INIT/ assertion does). The IPC will not be serviced for at least four clock cycles following CLR/ assertion.

Response to CLR/ is disabled by the first CLR/ assertion and is reenabled when the IP receives the first IPC (or INIT/ assertion).

CLR/ is sampled by the IP on the rising edge of CLKA.

INIT/ (Initialize, Input, low asserted)

Assertion of INIT/ causes the internal state of the IP to be reset and starts execution of the initialization microcode. INIT/ must be asserted for a minimum of 10 clock cycles. After the INIT/ pin is returned to its nonasserted state, IP microcode will initialize all of the internal registers and windows and will wait for a local IPC.

During INIT/ assertion, the FATAL/ and HERR pins are sampled by the IP to establish the mode in which the two bus interfaces of the IP are to participate in hardware error detection. Table 7 specifies the encoding of the master/checker modes.

Table 7. Representation of MASTER/CHECKER Modes at Initialization

FATAL/	HERR	iAPX 432 Side	Peripheral Subsystem Side
0	0	MASTER	MASTER
0	1	MASTER	CHECKER
1	0	CHECKER	MASTER
1	1	CHECKER	CHECKER

Clock Group

CLKA, CLKB (Clock A, Clock B, Inputs)

CLKA provides the basic timing reference for the IP. CLKB follows CLKA by one-quarter cycle and is used to assist internal timings.

Peripheral Subsystem Bus Group

AD_{15}—AD_0 (Address/Data, Input/Output)

These pins constitute a multiplexed address and data input/output bus. When the attached processor bus is idle or during the first part of an access, these pins normally view the bus as an address. The address is asynchronously checked to see if it falls within (matches) any one of the five window address ranges. The address is latched on the falling edge of ALE thereby maintaining the state of a match or no match for the remainder of the access cycle. The addresses are then unlatched on the falling edge of OE.

Once SYNC has pulsed high, the AD_{15}—AD_0 pins become data input and output pins. When WR/ is high (read mode), data is now accessed in the IP and the output buffers are enabled onto the AD pins if the OE is asserted. When WR/ is low (write mode), data is sampled by the IP after the rising edge of SYNC during the CLKA high time.

The address is always a 16-bit, unsigned number. Data may be either 8 bits or 16 bits as defined by BHEN/ and AD_0. The 8-bit data may be transferred on either the high (AD_{15}—AD_8) or the low (AD_7—AD_0) byte. When 8-bit data is transferred on the high or low byte, the opposite byte is 3-stated.

Twenty-bit addresses are accommodated by the external decoding of the additional address bits and are incorporated in the external CS/ logic.

During the clock in which write data is sampled, data must be set up before the rising edge of CLKA and must be held until the falling edge of that CLKA. Read data is driven out from a CLKA high and should be sampled on the next rising edge of CLKA.

Hardware error detection sampling is not done synchronously to CLKA. It is sampled by the falling edge of the OE pin. The internal AD pin hardware error detection signal is then clocked and output on the HERR pin. At this point it may still not be synchronous with CLKA and should be externally synchronized.

BHEN/ (Byte High Enable, Input, low asserted)

This pin, together with AD_0, determines whether 8 or 16 bits of data are to be accessed, and if it is 8 bits, whether it is to be accessed on the upper or lower byte position. This pin is latched by the falling edge of ALE and unlatched by the falling edge of OE. BHEN/ and AD_0 decode as shown in Table 8.

Table 8. Bus Data Controls

BHEN/	AD_0	Description
0	0	16-bit access
0	1	8 bits on upper byte, lower byte tristated
1	0	8 bits on lower byte, upper byte tristated
1	1	8 bits on lower byte, upper byte tristated

CS/ (Chip Select, Input, low asserted)

Chip Select specifies that this IP is selected and that a read or write cycle is requested. This pin is latched by the falling edge of ALE and unlatched by the falling edge of OE.

WR/ (Write, Input, low asserted)

This pin specifies whether the access is to be a read or a write. WR/ is asserted high for a read and asserted low for a write. This pin is latched by the falling edge of ALE and unlatched by the falling edge of OE.

PS Timing Group

ALE (Address Latch Enable, Input, Rising- and Falling-Edge-Triggered)

The rising edge of ALE sets a flip-flop which enables Transfer Acknowledge (XACK/) to become active. The falling edge of ALE latches the address on the AD_{15}-AD_0 pins and latches WR/, BHEN/ and CS/. Figure 19 shows two styles of ALE.

OE (Data Output Enable, Input, high asserted)

During a read cycle the OE pin enables read data on to the AD_{15}-AD_0 pins when it is asserted. During a read or write cycle the falling edge of OE signifies the end of the access cycle. Specifically, the falling edge of OE does three things:

1. Resets the XACK/ enable flip-flop, thereby terminating XACK/.
2. Terminates DEN/ (if read cycle).
3. Opens address latches WR/, BHEN/, and CS/.

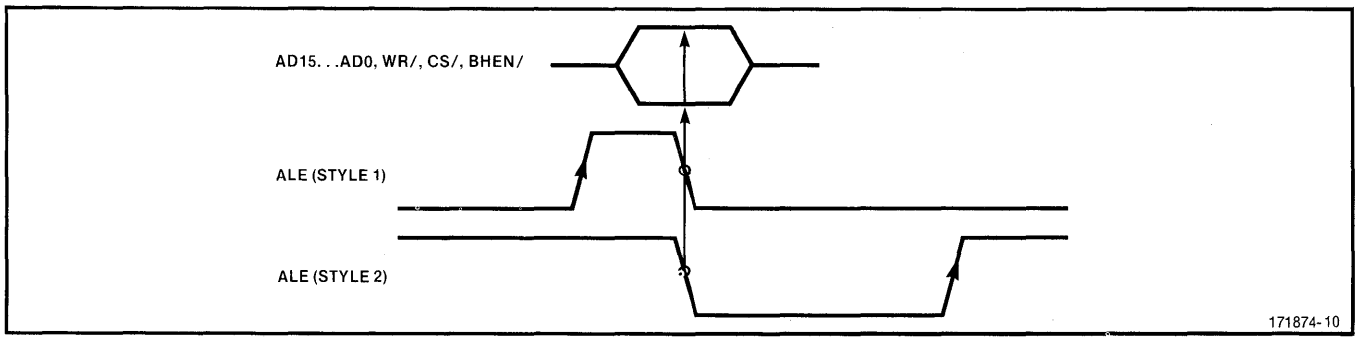


Figure 19. Two styles of ALE

SYNC (Synchronized Qualifier Signal, Input, high asserted)

A rising edge on this signal must be synchronized to the IP CLKA falling edge. This signal qualifies the address, BHEN/, CS/ and WR/, indicating a valid condition. SYNC also initiates any internal action on the IP's part to process an access. It starts the request for data to the IP in a read access. In a write access, data is expected one or two CLKA's after SYNC pulses high. At initialization time, IP microcode sets the write sample delay to the slowest operation (two CLKA's after SYNC). However, this can be modified to one clock cycle by making a function request to the IP to change the write sample delay.

When the hold/hold-acknowledge mechanism of the IP is used, and once HDA has pulsed high, a SYNC pulse is required to qualify the hold acknowledge since the HDA pin can be asynchronous.

PS Buffer Control Group

DEN/ (Data Enable, Output, low asserted)

This pin enables external data buffers which would be used in systems where the address and data are not multiplexed (e.g., a Multibus system). DEN/ assertion begins no sooner than the CLKA high time of the first clock of SYNC assertion if a valid, mappable address range is detected. It is terminated with the falling edge of OE. In a write access, it is also terminated after XACK/ assertion.

Hardware error detection occurs during the first clock of SYNC assertion.

PS Interlock Group

HLD (Hold Request, Output, high asserted)

The hold/hold-acknowledge mechanism is an interlocking mechanism between the peripheral subsystem and the IP. Hold is used by the IP to gain control of the subsystem bus to ensure that no subsystem processors will make an access to the IP while it alters internal registers.

This signal is put out synchronously with the rising edge of CLKA. Hardware error detection sampling occurs during CLKA low time.

In special cases it may not be necessary to use the HLD function interlocking. In this case HDA can be tied high and no SYNC pulse will be required for HDA qualification. The hardware detects this condition by noting that the HDA pin was high a half clock before HLD requests a hold. In this mode the HLD output still functions and can be monitored if desired.

HDA (Hold Acknowledge, Input, high asserted)

HDA is asserted by the peripheral subsystem when the IP's request for a hold has been granted. This pin need only be a high pulse and can be asynchronous to CLKA. This pin must be followed by a SYNC pulse in order to synchronously qualify it.

PS Synchronization Group

XACK/ (Transfer Acknowledge, Output, low asserted)

XACK/ is used to acknowledge that a data transfer has taken place.

For random or local accesses, XACK/ indicates that a transfer to or from iAPX 432 memory has been completed.

For buffered accesses where the XACK-Delay is not in the advanced mode, XACK/ signifies that the transfer from/to the prefetch/postwrite buffer in the IP has been completed.

For buffered accesses which use advanced acknowledge mode (XD=0) the formation of an advanced XACK/ signal is requested. This allows the possibility of interfacing to the peripheral subsystem without wait states. The acknowledge will be advanced if the access is a read operation and the buffer contains the required data or the access is a write operation and the buffer contains sufficient space to accept the write data. In addition, the access must be valid.

If XACK/ is preceded by a low pulse on NAK/, then XACK/ signifies that the access encountered a fault. If the access was a random access, other than window #4, the window will be placed in the faulted state and any further accesses to this window will be ignored by the IP.

If the IP is programmed to be in advanced acknowledge mode (XD=0) and XACK/ is not returned before the peripheral subsystem issued SYNC, then XACK/ will be postponed until valid data has been established on the AD_{15}-AD_0 bus.

Five conditions affecting XACK/ behavior are:

- XACK-Delay, user programmable through an IP function request. This parameter establishes the minimum operating XACK-delay with respect to the SYNC signal. Table 9 displays the representation of the XACK-delay codes.
- XACK-enable-flip-flop, set by the rising edge of the ALE signal and reset by the falling edge of the OE signal.
- Internal IP Registers. These are used to determine validity of the peripheral subsystem access and establish access modes.
- Type of access behavior: Random or Buffered, Memory or Interconnect.
- Bus Faults, nonexistent memory, etc.

Hardware error detection occurs during the first clock of SYNC assertion.

NAK/ (Negative Acknowledge, Output, low asserted)

This signal precedes XACK/ by one-half clock cycle in order to qualify it as a negative acknowledge. This pin pulses low for only one clock period.

When the IP is in physical mode and making an interconnect access, negative acknowledge may be used to indicate that the access was made to a nonexistent interconnect address space. This will allow determination of the system configuration by a subsystem processor at system initialization time.

This pin could be used to set a status bit and cause a special interrupt to transmit the information back to the subsystem.

This signal is synchronously driven from the falling edge of CLKA. Hardware error detection occurs during CLKA high time.

INH1 (Inhibit, Output, high asserted)

This pin is asynchronously asserted by non-clocked logic when a valid mappable address range is detected. It can be used to override other memories in the peripheral subsystem whose address space is overlapped by an IP window. After initialization, the microcode sets the INH1 mode for each window by loading registers in the IP for each window. Once the subsystem is allowed to make a function request, it can selectively disable or enable the inhibit mode on each window. This pin is gated off by CS/.

The selection of the inhibit mode for window 0, when in buffered mode, causes a corresponding built-in XACK-delay which delays the acknowledge from going active until two clock periods after the rising edge of SYNC. This was done to facilitate most Multibus systems using INH1, as they require that the acknowledge be delayed. When the Advanced XACK/ mode is programmed, the inhibit mode should not be used on window 0 when in buffered mode, since the acknowledge will not be effectively delayed.

Hardware error detection occurs during the first clock of SYNC assertion.

Table 9. XACK/ Timing Parameters

Inhibit Mode	WR/	XD_1	XD_0	XACK/ Formation
0	X	0	0	Advanced Acknowledge (XACK/ can occur before SYNC)
0	1	0	1	Rising edge of SYNC
0	0	0	1	Rising edge of SYNC plus 1 Clock
0	1	1	0	Rising edge of SYNC plus 1 Clock
0	0	1	0	Rising edge of SYNC plus 2 Clocks
1	X	1	0	Rising edge of SYNC plus 2 Clocks
1	X	0	1	Rising edge of SYNC plus 2 Clocks
X	X	1	1	Illegal condition

Note: X=don't care condition

PS Interrupt Group

INT (Interrupt, Output, high asserted)

This output is a pulse 2 CLKA's wide, and is synchronously driven from the rising edge of CLKA. Hardware error detection occurs during CLKA low time.

PS Reset Group

PSR (Peripheral Subsystem Reset, Output, high asserted)

PSR is asserted by the IP under microprogram control. When asserted, the peripheral subsystem should be reset. In a debug type of control, it may be desirable to use this pin to set a status bit in an external register or possibly cause a special interrupt. This pin is normally asserted by the IP when the peripheral subsystem is believed to be faulty and would not respond to other means of control.

This signal is put out synchronously with the rising edge of CLKA. Hardware error detection sampling occurs during CLKA low time.

43203 ELECTRICAL CHARACTERISTICS

Tables 10 through 12 and Figures 20 through 25 provide electrical specification information and include input/output timing, read and write timing, and component maximum ratings.

Instruction Set Comparison

Refer to Table 13 for a GDP/IP operator comparison.

Table 10. 43203 Absolute Maximum Ratings

Absolute Maximum Ratings	
Ambient Temperature Under Bias	0° C to 70° C
Storage Temperature	−65° C to +150° C
Voltage on Any Pin with Respect to GND	−1 V to +7 V
Power Dissipation	2.5 Watts

Table 11. iAPX 43203 DC Characteristics

VCC = 5V±10%		Ta = 0°C to 70°C		
Spec	**Description**	**Min**	**Max**	**Units**
Vilc	Clock Input Low Voltage	−0.3	+0.5	V
Vihc*	Clock Input High Voltage	3.5	VCC+0.5	V
Vil	Input Low Voltage	−0.3	0.8	V
Vih	Input High Voltage	2	VCC+0.5	V
Icc	Power Supply Current	–	450	mA
Iil	Input Leakage Current	–	±10	uA
Io	Output Leakage Current	–	±10	uA
Iol	@0.45 Vol			
	HERR	–	8	mA
	FATAL/	–	4	mA
	AD15...AD0	–	4	mA
	OTHER	–	2	mA
Ioh	@2.4 Voh	–	−0.1	mA

* For operation at 5 MHz or slower, the 43203 may be operated with a Vihc minimum of 2.7 volts.

Table 12. iAPX 43203 AC Characteristics

VCC = 5 ± 10% Ta = 0°C to 70°C Loading: AD15...AD0 20 to 100pf
OTHER 20 to 70pf

Symbol	Description	8 MHz.		5MHz.		Unit
		Min	Max	Min	Max	
	GLOBAL TIMING REQUIREMENTS					
tcy	Clock Cycle Time	125	1000	200	1000	nsec.
tr,tf	Clock Rise and Fall Time	–	10	–	10	nsec.
t1,t2						
t3,t4	Clock Pulse Widths	26	250	45	250	nsec.
tis	INIT/ to Signal Hold Time	15	–	20	–	nsec.
tsi	Signal to INIT/ Setup Time	10	–	10	–	nsec.
tie	INIT/ Enable Time	10	–	10	–	tcy
	SYSTEM SIDE TIMING REQUIREMENTS					
tdc	Signal to CLOCK Setup Time	5	–	5	–	nsec.
tcd	Clock to Signal Delay Time	–	55	–	85	nsec.
tdh	Clock to Signal Hold Time	25	–	35	–	nsec.
toh	Clock to Signal Output Hold Time	15	–	20	–	nsec.
ten	Clock to Signal Output Enable Time	15	–	20	–	nsec.
tdf	Clock to Signal Data Float Time	–	55	–	75	nsec.
	PERIPHERAL SUBSYSTEM SIDE TIMING REQUIREMENTS					
tas	AD15...AD0,CS/,WR/,BHEN/ Setup Time to ALE Low	0	–	0	–	nsec.
tah	AD15...AD0,CS/,WR/,BHEN/ Hold Time to ALE Low	32	–	35	–	nsec.
tss	SYNC High Setup Time to CLKA High	50	–	60	–	nsec.
tsh	SYNC Low Hold Time to CLKA High	30	–	40	–	nsec.
tsw	SYNC High Pulse Width	50	tss + 1.5tcy	60	tss + 1.5tcy	nsec.
tds	Write Data Setup to Sampling CLKA High	10	–	20	–	nsec.
tdx*	Write Data Hold to Sampling CLKA Low (Advanced XACK/)	10	–	20	–	nsec.
tdhx	Write Data Hold to XACK/	5	–	5	–	nsec.
tasy	AD15...AD0,CS/,WR/,BHEN/ Setup to SYNC	120	–	160	–	nsec.
	PERIPHERAL SUBSYSTEM TIMING RESPONSES					
tsdh	CLKA High to HLD,INT,PSR	–	75	–	90	nsec.
taih	Valid AD15...AD0,CS/ to Chip INH1 Valid Delay	–	80	–	85	nsec.
tede	OE to DEN/ Delay	–	65	–	70	nsec.
tead	OE to Enable AD15...AD0 Buffers Delay (Read Cycle)	–	70	–	75	nsec.
tdad	OE to Disable AD15...AD0 Buffers Delay (Read Cycle)	–	52	–	52	nsec.
tced	CLKA High to Enable AD15..AD0 Buffers Delay	–	70	–	75	nsec.
tcvd	CLKA High to Valid Read Data Delay	–	80	–	90	nsec.
tox	OE Inactive to XACK/ Inactive Delay	–	80	–	90	nsec.
tdds	AD15...AD0 Disable Setup to DEN/ High	0	–	0	–	nsec.
txde	XACK/ Low to DEN/ High (Write Cycle)	–	35	–	40	nsec.
tcde	CLKA High to DEN/ Low	–	70	–	75	nsec.

Table 12. iAPX 43203 AC Characteristics (Cont'd.)

Symbol	Description	8 MHz.		5MHz.		Unit
		Min	Max	Min	Max	
	XACK/ TIMING CHARACTERISTICS					
	Buffered Accesses with XD= 0					
tax	ALE High to XACK/ Valid	0	65	0	70	nsec.
tdsx	AD15...AD0 Read Data Valid Setup to XACK/ Valid (When internal state does not allow XACK/ before SYNC)	20	–	20	–	nsec.
tadx	Valid AD15...AD0 to XACK/ Valid (When internal state allows XACK/ before SYNC)	–	120	–	140	nsec.
	Buffered Accesses (With XD=1 or XD=2) or Random Accesses					
tdsx	AD15...AD0 Read Data Valid Setup to XACK/	20	–	20	–	nsec.
	Faulted Accesses					
tsdl	CLKA Low to NAK/	–	75	–	90	nsec.
tsnx	Setup of NAK/ to XACK/	50	–	50	–	nsec.

Note: All timing parameters are measured at the 1.5 Volt level except for CLKA and CLKB which are measured at the 1.8 Volt level.

* Write data is sampled for only one clock cycle. The PS must meet the t_{DHX} specification thereby guaranteeing t_{DX}.

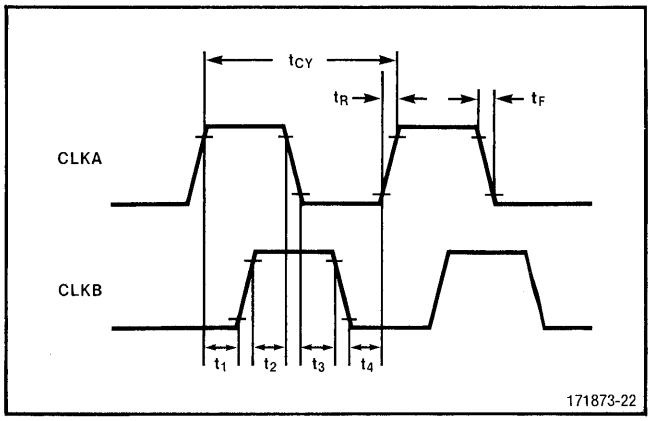


Figure 20. 43203 Clock Input Specification

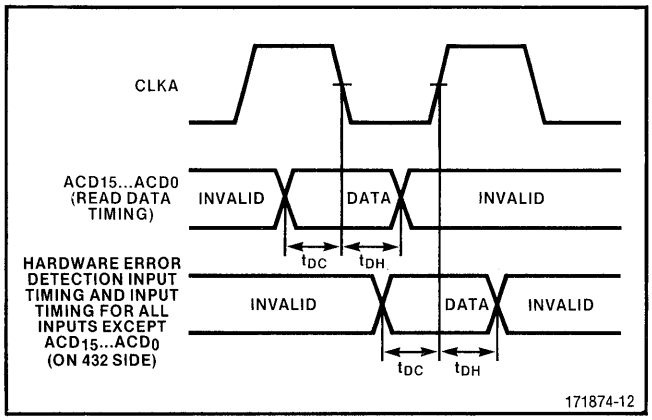


Figure 22. 43203 Input Timing Specification

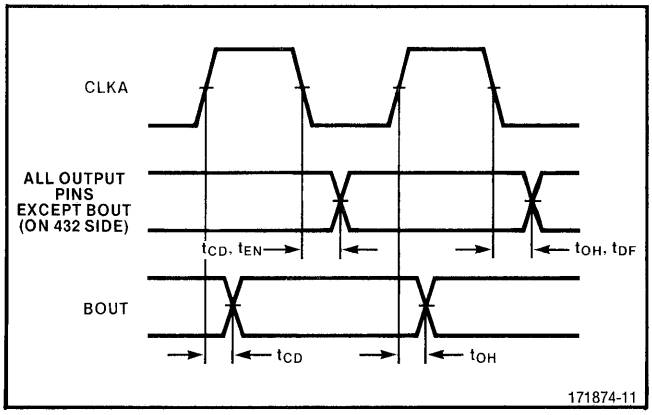


Figure 21. 43203 Output Timing Specification

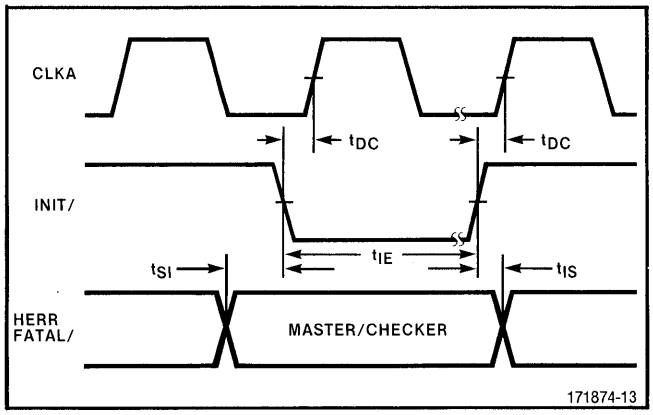


Figure 23. 43203 Initialization Timing

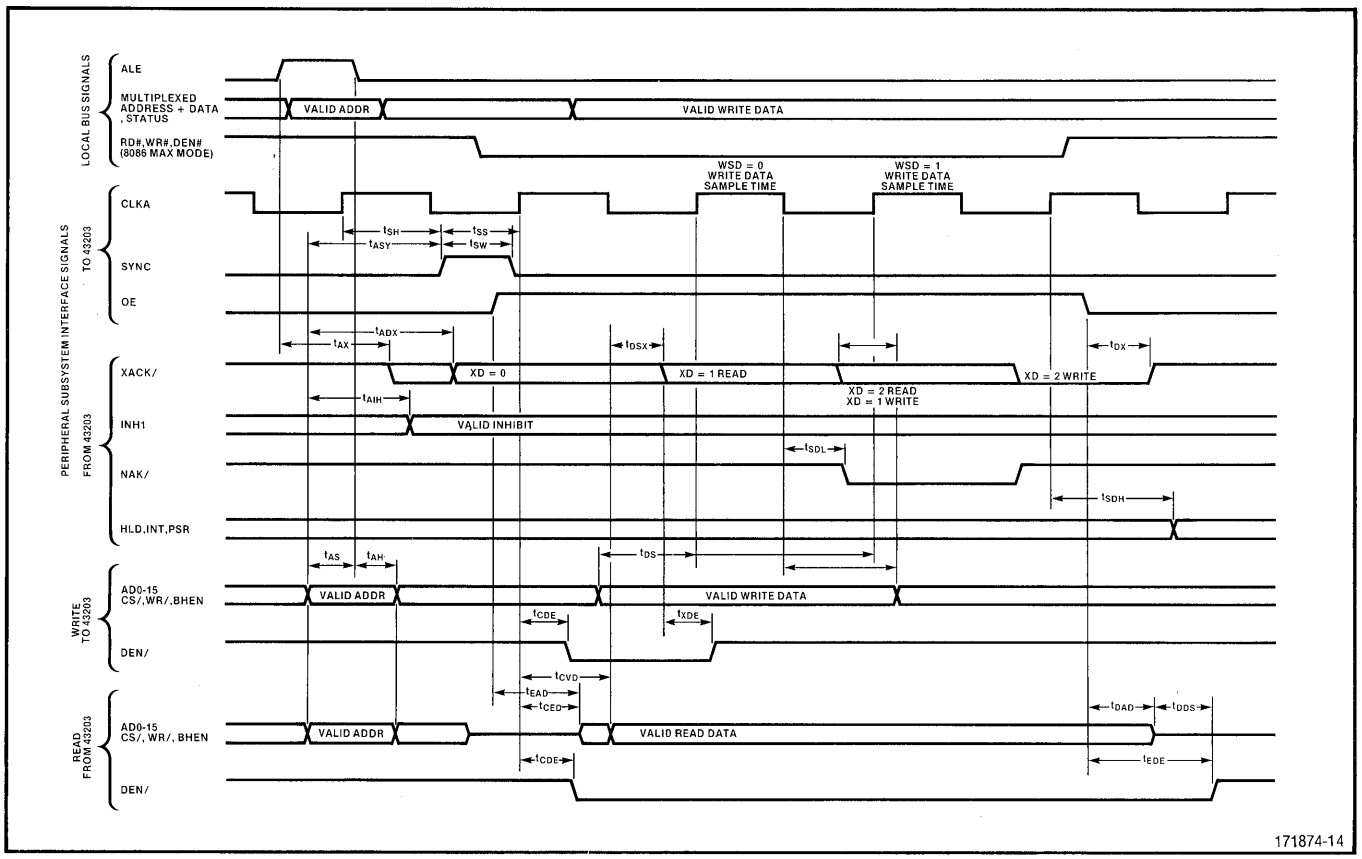


Figure 24. Local Processor Bus Timing

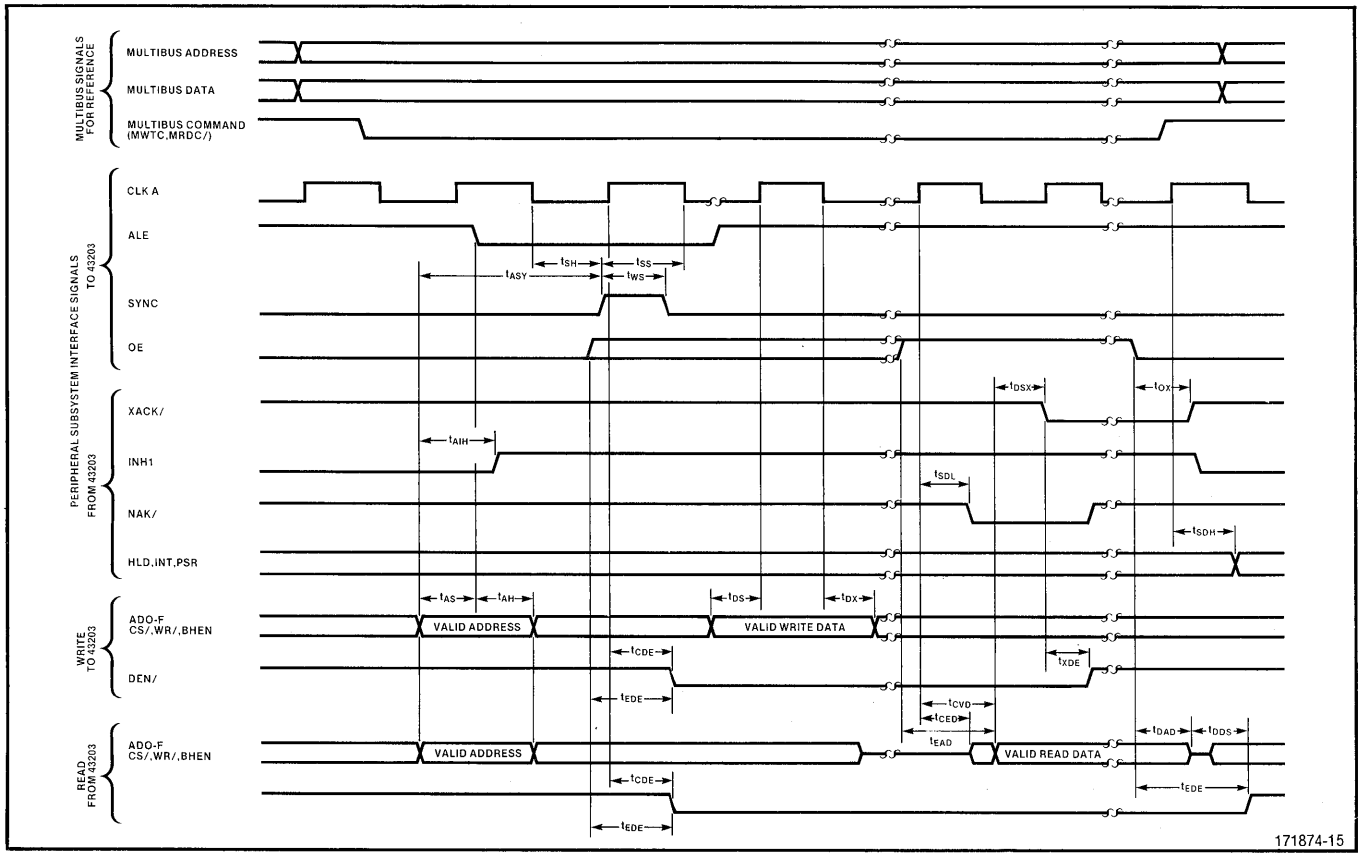


Figure 25. Multibus™ Interface Timing

Table 13. GDP/IP Operator Comparison

OPERATOR	IP IMPLEMENTATION
Window Definition Operator	
Update Window	+
Access Descriptor Movement Operators	
Copy Access Descriptor	=
Null Access Descriptor	=
Rights Manipulation Operators	
Amplify Rights	=
Restrict Rights	=
Type Definition Manipulation Operators	
Create Public Type	–
Create Private Type	–
Retrieve Public Type	=
Retrieve Type	=
Retrieve Type Definition	=
Refinement Operators	
Create Generic Refinement	=
Create Typed Refinement	=
Retrieve Refinement	=
Segment Creation Operators	
Create Data Segment	–
Create Access Segment	–
Create Typed Segment	–
Create Access Descriptor	–
Access Path Inspection Operators	
Inspect Access Descriptor	=
Inspect Access Path	=
Object Interlock Operators	
Lock Object	=
Unlock Object	=
Indivisibly Add Short Ordinal	–
Indivisibly Add Ordinial	–
Indivisibly Insert Short Ordinal	–
Indivisibly Insert Ordinal	–
Context Communication Operators	
Enter Access Segment	=
Enter Process Globals Access Segment	=
Set Context Mode	/
Call	–
Call Context with Message	–
Return	–

Table 13. GDP/IP Operator Comparison (Cont'd.)

OPERATOR	IP IMPLEMENTATION
Process Communication Operators	
Send	=
Receive	=
Conditional Send	=
Conditional Receive	=
Surrogate Send	=
Surrogate Receive	=
Delay	=
Read Process Clock	=
Processor Communication Operators	
Send to Processor	=
Broadcast to Processors	=
Read Processor Status and Clock	=
Move to Interconnect	–
Move from Interconnect	–
Branch Operators	–
Character Operators	–
Short-Ordinal Operators	–
Short-Integer Operators	–
Ordinal Operators	–
Integer Operators	–
Short-Real Operators	–
Real Operators	–
Temporary-Real Operators	–

Legend:
= IP and GDP identical implementation
+ GDP does not implement operator
– IP does not implement operator
/ While conceptually similar, IP implements operator differently than GDP

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara 95051 (408) 734-8102 x598
Printed in U.S.A./Y-33/0281/50K/PS/GFH