# iAPX 43201 iAPX 43202 PRELIMINARY
# VLSI GENERAL DATA PROCESSOR

- **Self-Dispatching Processors for Software-Transparent Multiprocessing**
- **Hardware Implemented Inter-Process Communication and Dynamic Storage Allocation**
- **High-Level Language Directed Instruction Set with 0-3 Operand References**
- **Functional Redundancy Checking Mode for Hardware Error Detection**
- **Capability-Based Addressing and Protection**
- **$2^{40}$ Bytes of Virtual Address Space**
- **Object-Based Architecture for Improved Programmer Productivity**
- **Symmetrical Support of All 8-, 16-, and 32-Bit Scalar Data Types and Proposed IEEE Standard 32-, 64-, and 80-Bit Floating Point**

The Intel iAPX 432 General Data Processor (GDP) is a 32-bit microprocessor that consists of two VLSI devices, the 43201 and the 43202. These companion devices (shown in Figures 1 and 2) provide the general data processing facility of the iAPX 432 Micromainframe. The combination of VLSI technology and advanced architecture in the iAPX 432 system results in mainframe functionality with a microcomputer form factor. The new object-based architecture significantly reduces the cost of large software systems and enhances their reliability and security.

Software-transparent multiprocessing allows the user to configure systems matched to the required performance and provides an easy growth path. Hardware support for operating systems and high-level languages eases their implementation.

The GDP provides $2^{40}$ bytes of virtual address space with capability-based addressing and protection. In addition, a hardware-implemented functional redundancy checking mode is provided for the detection of hardware errors.

The iAPX 43201 and iAPX 43202 are fabricated with Intel's highly reliable +5-volt, depletion load, N-channel, silicon gate HMOS technology and each is packaged in a 64-pin Quad In-Line Package (QUIP).

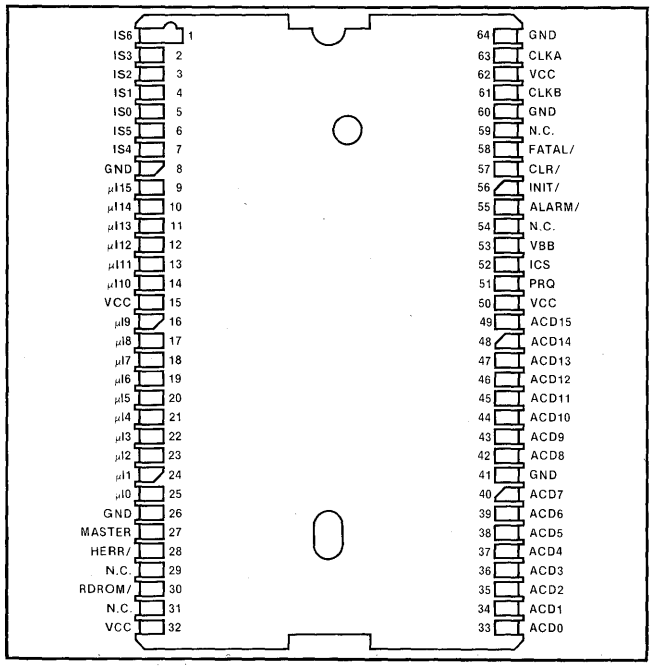



**Figure 1. 43201 Pin Assignment Instruction Decoder/Microinstruction Sequencer** 171873-1

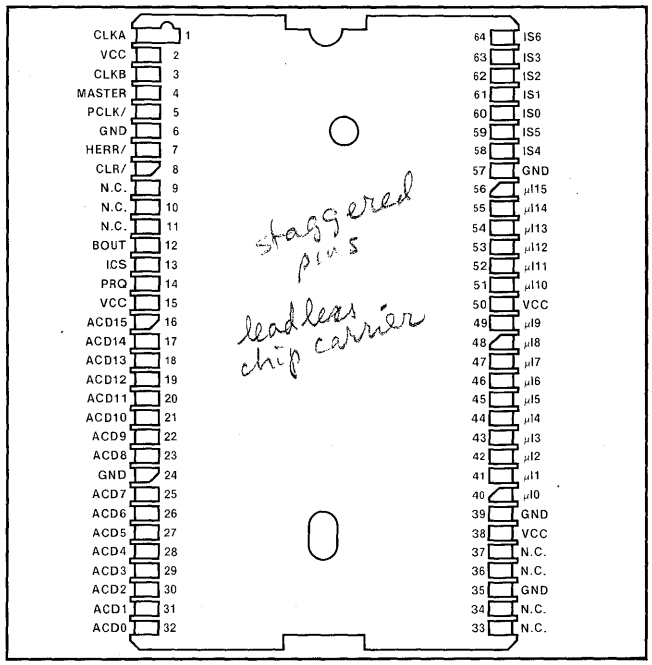



**Figure 2. 43202 Pin Assignment Execution Unit** 171873-2

## iAPX 432 GDP FUNCTIONAL DESCRIPTION

The general data processor is organized internally as a three-stage microprogram-controlled pipeline. The first stage is the instruction decoder (ID); the second stage is the microinstruction sequencer (MS); and the third stage is the execution unit (EU).

The first two stages of the pipeline are physically located on the 43201 (Figure 3). Each stage of the pipeline can be considered an independent subprocessor which operates until the pipeline is full and then halts and waits for more work to do.

### Instruction Decoder

The first subprocessor of the pipeline is the ID, which performs the following functions:

- Receives macroinstructions
- Processes variable-length fields
- Extracts logical addresses
- Generates starting addresses for the microinstruction procedures
- Generates microinstructions for simple operations

The general task facing the Instruction Decoder is to interpret the macro-instruction stream to determine which micro-instruction sequence should be initiated next and to extract logical address data. The major sub-tasks involved in accomplishing the larger goal are:

- iAPX 432 instructions contain a variable number of bits, depending upon the complexity of the instruction. Instructions may range from a few bits long to several hundred bits long. Instructions may extend over many words in memory. The Instruction Decoder requests words from memory as they are needed.
- A GDP instruction is composed of a variable number of fields and each field may contain a variable number of bits. In most cases, the

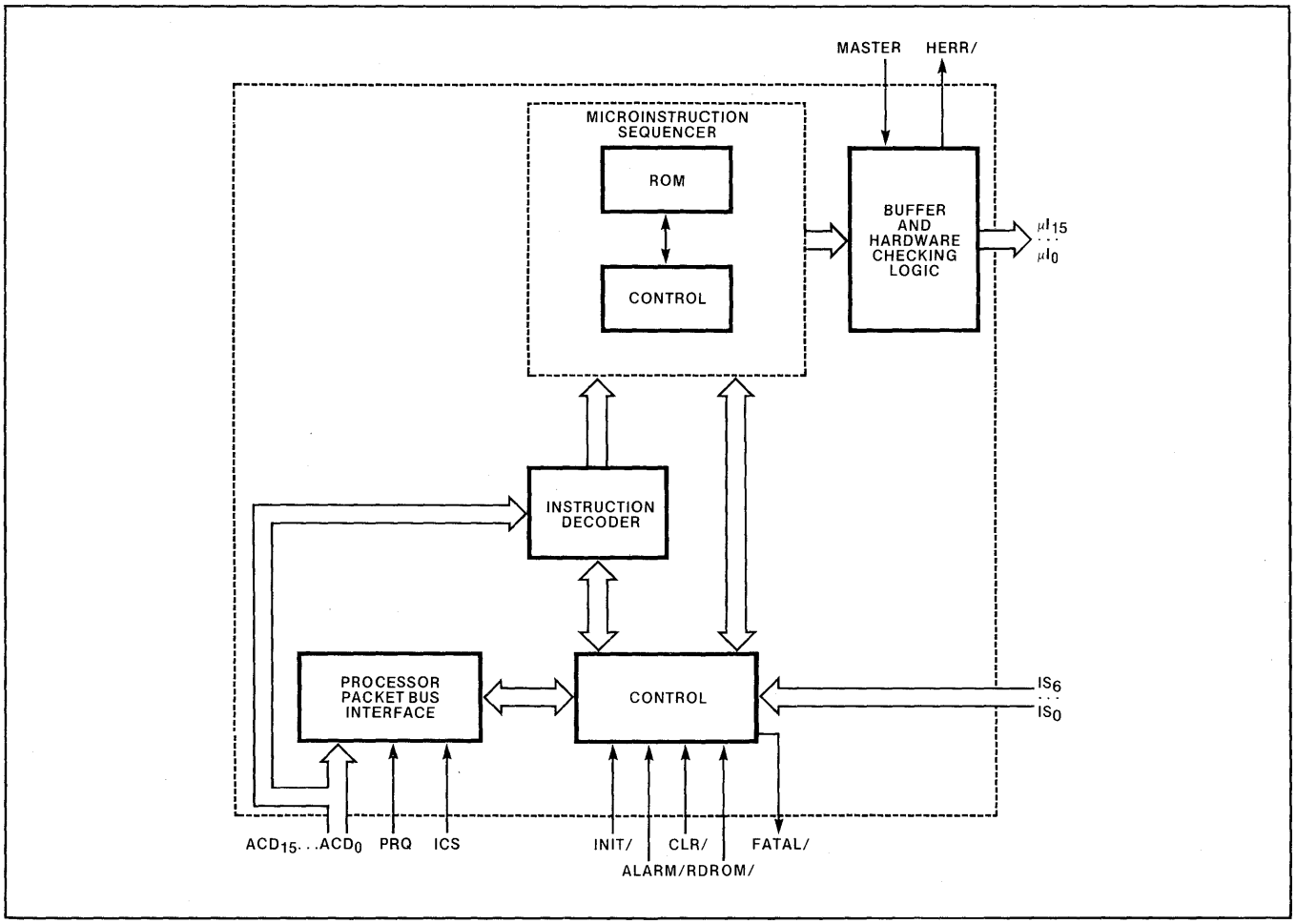



**Figure 3. 43201 Block Diagram**

171873-3

encoding of a field specifies its length. The Instruction Decoder interprets a field to find its length.

- The Instruction Decoder determines when an instruction boundary has been reached so that it may properly begin decoding the next instruction.
- In some cases, the interpretation of one field may depend upon the value of some previous field. In particular, the interpretation of the opcode (the last field of an instruction) depends on the value of the class field of that instruction (the first field). The Instruction Decoder saves enough information about the instruction to properly interpret every field in the instruction.
- A GDP instruction may contain an explicit reference to some location in memory. This logical address information must be transferred to the Reference Generation Unit so that the correct physical address of the operand may be generated. As with all fields of a GDP instruction, the logical address fields are also variable length fields. The Instruction Decoder has provisions for formatting the logical address information and storing it until needed by the Reference Generation Unit.
- The iAPX 432 instruction set contains several branch instructions. Since iAPX 432 instructions may start at any bit in the segment, the Instruction Decoder is able to start decoding at any point in the segment. Since branches occur fairly often in a typical instruction stream, it is also desirable to minimize the start-up time of the GDP after a branch has occurred.
- The Instruction Decoder provides a mechanism to recover from an instruction that faults. The information necessary for fault recovery will be retained by the Instruction Decoder until the instruction is successfully completed.

## Microinstruction Sequencer

The second subprocessor in the pipeline is the Microinstruction Sequencer (MS) which performs the following functions:

- Issues microinstructions to the Execution Unit (EU) (43202)
- Executes microcode sequences out of an on-chip, 4.0K x 16-bit microcode ROM
- Responds to the bus control signals
- Invokes macroinstruction fetches
- Initiates interprocessor communication and fault handling sequences

The role of the Microinstruction Sequencer is to decide which microinstruction should be sent to the Execution Unit for each cycle. The Microinstruction Sequencer must consider each of the following when generating microinstructions:

- There are two sources of microinstructions. They may come from either the Instruction Decoder or from a ROM contained in the Microinstruction Sequencer (MS). The MS must choose the appropriate source.
- The Microinstruction Sequencer must compute the address in ROM (if any) of the next microinstruction.
- The Execution Unit may require variable lengths of time to complete some microinstructions. The Microinstruction Sequencer waits for the Execution Unit to finish the requested operation.

## Execution Unit

The 43202 contains the third stage of the GDP pipeline—the Execution Unit (EU). (Refer to Figure 4.) This unit receives microinstructions from the 43201 and routes them to one of the two independent subprocessors that make up the EU. These two are the Data Manipulation Unit (DMU) and the Reference Generation Unit (RGU).

The EU executes most microinstructions in one clock cycle. However, each of the subprocessors has an associated sequencer that may run for many cycles in response to certain microinstructions. Those sequencers are invoked for complicated arithmetic operations (in the Data Manipulation Unit) and Processor Packet bus transactions (in the Reference Generation Unit).

The Data Manipulation Unit contains the registers and arithmetic capabilities to perform the following functions:

- Hardware recognition of nine (9) data types
- Built-in state machine for 16- and 32-bit multiply, divide, and remainder
- Control functions for 32-, 64-, and 80-bit floating point arithmetic

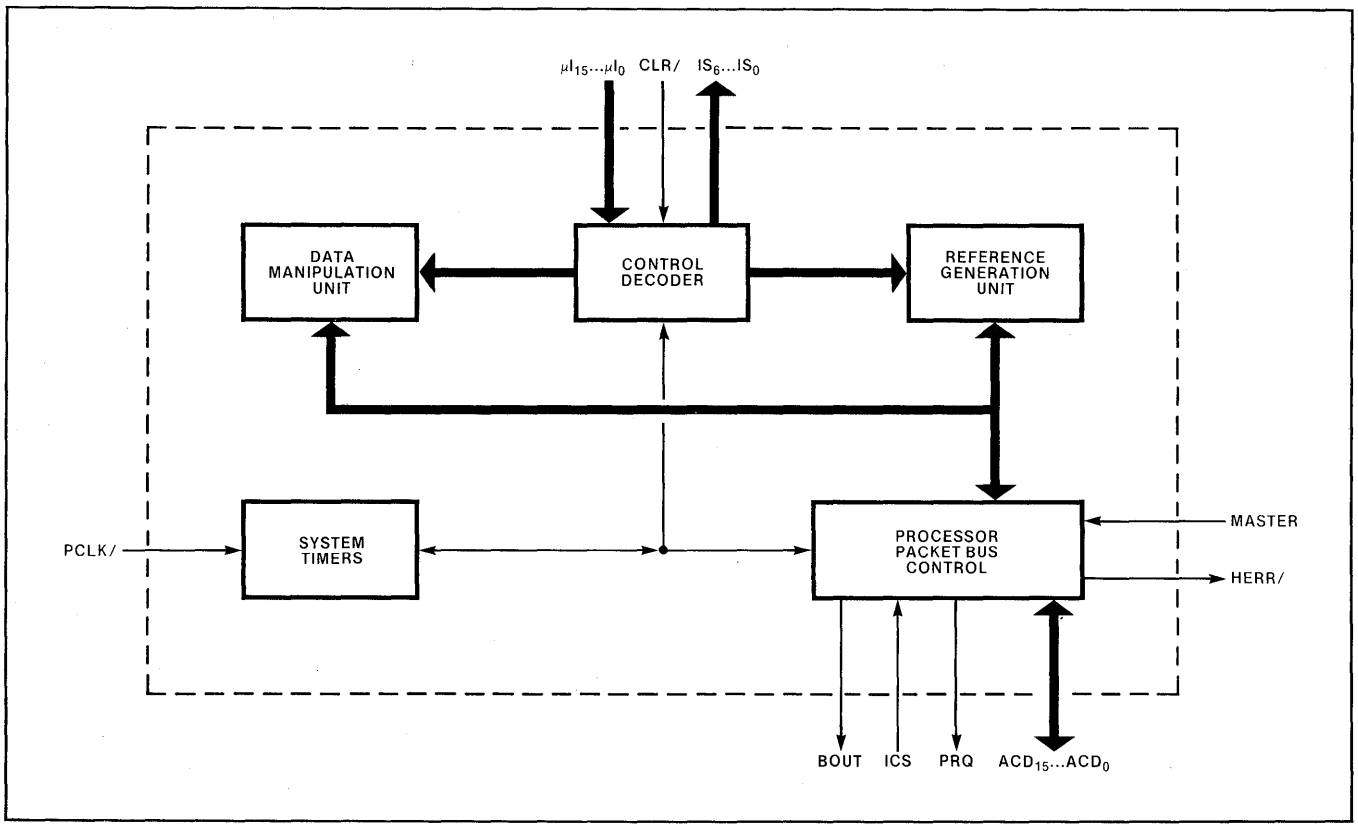



**Figure 4. 43202 Block Diagram** 171873-4

The Reference Generation Unit performs the following functions:

- Provides the translation of a 40-bit virtual address into a 24-bit physical address
- Provides for a hardware-enforced domain protection system (read, write, alter, accessed)
- Handles sequencing for 8-, 16-, 32-, 64-, and 80-bit memory accesses
- Controls on-chip top-of-stack register

The Execution Unit manipulates data and translates the logical addresses into physical addresses. The efficient performance of these tasks requires:

- While most microinstructions require only a single cycle to complete, there are some that require multiple and even variable numbers of cycles. As a result there are two sequences in the Execution Unit. One sequence is associated with the Data Manipulation Unit and is responsible for controlling multiple-cycle arithmetic operations. The other sequencer works in conjunction with the Reference Generation Unit and is responsible for running cycles on the Processor Packet bus.
- When a reference to a given memory segment has been translated from its logical representation to a physical address, there is a cache in the Reference Generation Unit that maintains the physical base address as well as the length of the segment. Future references to the same segment can use this cached information as the basis for logical to physical address translation.
- There is a hardware-implemented feature which uses least-recently-used algorithms for deciding which cached segment base-length pair to replace when a new segment is referenced.
- The top 16-bit element of the operand stack can be stored in a register in the Data Manipulation Unit.
- A circuit in the Reference Generation Unit checks every memory reference to see if it is within the length of its segment. Since the iAPX 432 architecture controls the type of access (read, write) as well, whether or not the access is allowed at all, this hardware also verifies that the reference is of the proper type.

The 43201 and 43202 components, described above, together form the GDP. Figure 5 is a block diagram that shows both units interfacing to the Packet bus as a single processor.

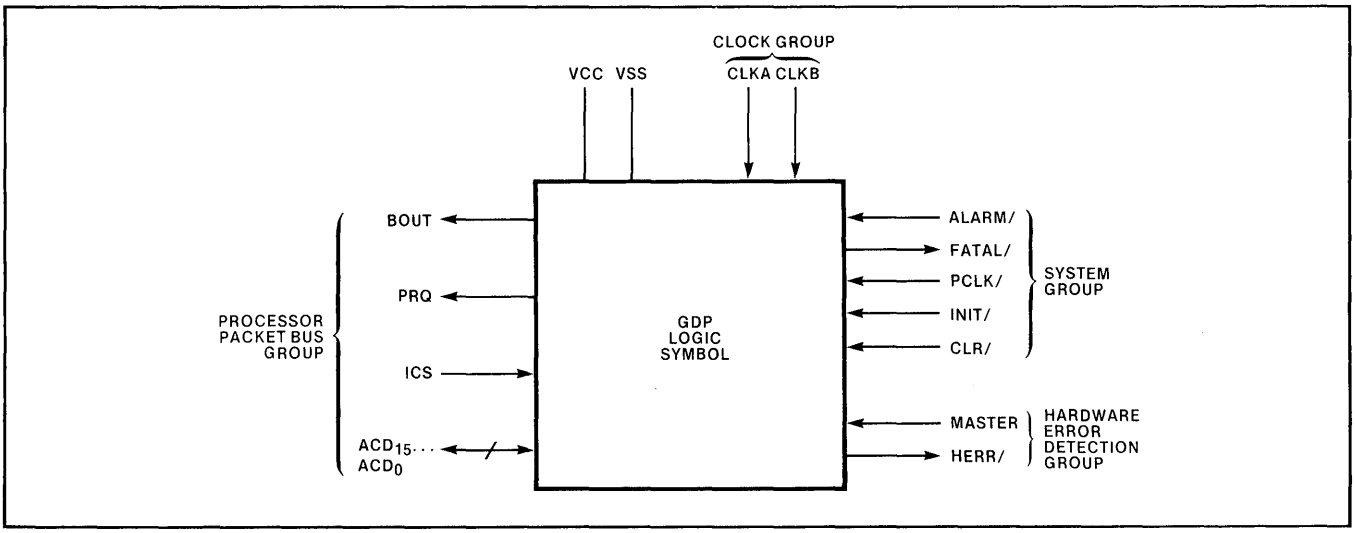



**Figure 5. GDP Block Diagram**

171873-5

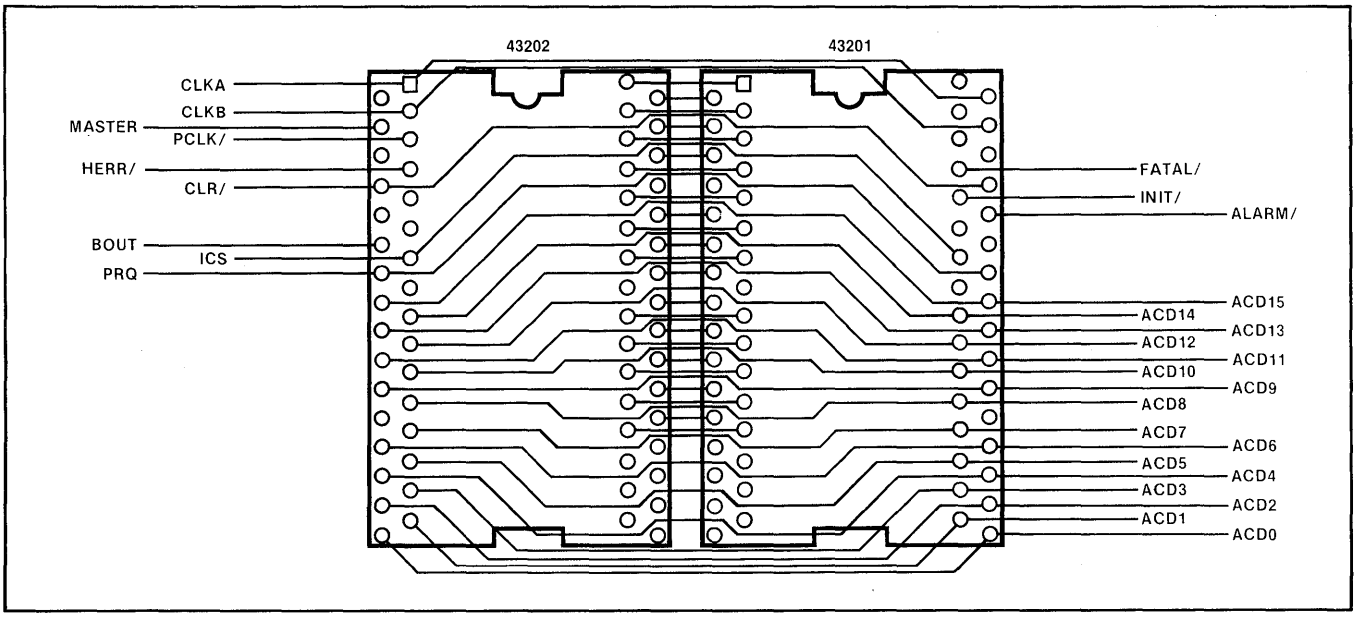



**Figure 6. GDP Layout**

171873-6

## iAPX 43201/43202 PHYSICAL INTERCONNECT

Figure 6 illustrates how the 43201 and 43202 are layed-out to form a GDP.

## 432 INSTRUCTIONS

Intel iAPX 432 instruction codes have been designed to minimize the space the instructions occupy in memory and still allow for efficient encoding. In order to achieve the ultimate in efficiency of storage, the instructions are encoded without regard for byte, word, or other artificial boundaries. The instructions may be viewed as a linear sequence of bits in memory, with each instruction occupying exactly the number of bits required for its complete specification.

iAPX 432 processors view these instructions as composed of fields of varying numbers of bits that are organized to present information to the Instruction Decoder in the sequence required for decoding. A unified form for all instructions allows instruction decoding of all instructions to proceed in the same fashion.

In general, GDP instructions consist of four main fields. These fields are called the class field, the format field, the reference field, and the opcode field. The reference field, in turn, may contain

several other fields, depending upon the number and complexity of the operand references in the instruction. The fields of a GDP instruction are stored in memory in the following format.

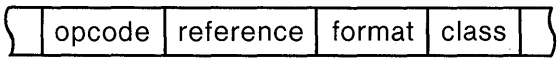
The class field is either 4- or 6-bits long, depending on its encoding. The class field specifies the number of operands required by the instruction and the primitive types of the operands. The class field may indicate 0, 1, 2 or 3 references.

If the class field indicates one or more references, a format field is required to specify whether the references are implicit or explicit and their uses.

In the case of explicit references the format field can indicate whether or not the reference is direct or indirect. Further, the format field may indicate that a single operand plays more than one role in the execution of the instruction. As an example, consider an instruction to increment the value of an integer in memory. This instruction contains a class field, which specifies that the operator is of order two and that the two operands both occupy a word of storage, followed by a format field, whose value indicates that a single reference specifies a logical address to be used both for fetching the source operand and for storing the result, followed by an explicit data reference to the integer to be incremented, and finally followed by an opcode field for the order-two operator INCREMENT INTEGER. It is possible for a format field to indicate that an instruction contains fewer explicit data references than are indicated by the instruction's class field. In such a case the other required data references are implicit references, and the corresponding source or result operands are obtained from or returned to the top of the operand stack. The use of implicit references is illustrated in the following example, which considers the high-level language statement

$$A=A+B*C$$

The instruction stream fragment for this statement consists of two instructions and has the following form:




Assume that A, B, and C are integer operands. The first class field (the rightmost field above) specifies that the operator requires three references and that all three references are to word operands.

The first format field contains a code specifying two explicit data references. These references are to supply only the two source operands. The destination is referenced implicitly so that the result of the multiplication is to be pushed onto the operand stack. The second class field is identical to the first and specifies three required references by the operator. In addition, all three references are to word operands. The second format field specifies one explicit data reference to be used for both the first source operand and the destination. The second source operand is referenced implicitly and is to be popped from the operand stack when the instruction is executed.

The reference fields themselves can be of various lengths and can appear in various numbers, consistent with their specification in the class and format fields. If implicit references are specified, reference fields for them will not appear. Direct references will require more bits to specify than indirect references.

Following the class, format, and reference fields, the opcode field appears. The opcode field specifies the operator to be applied to the operands specified in the preceding fields.

## Modes of Generation

Figures 7 and 8 illustrate the two iAPX 432 system modes of generation: Selector Generation and Displacement Generation.

The modes of Selector Generation are concerned with the object structure and how they are accessed by the operands. The four modes of Selector Generation shown are:

- Short Direct
- Long Direct
- Stack Indirect
- General Indirect

The modes of Displacement Generation specify the physical location and displacement of objects within a given segment or segment. The four modes of Displacement Generation are:

- Scalar Data Reference Mode
- Record Item Reference Mode
- Static Vector Element Reference Mode
- Dynamic Vector Element Reference Mode

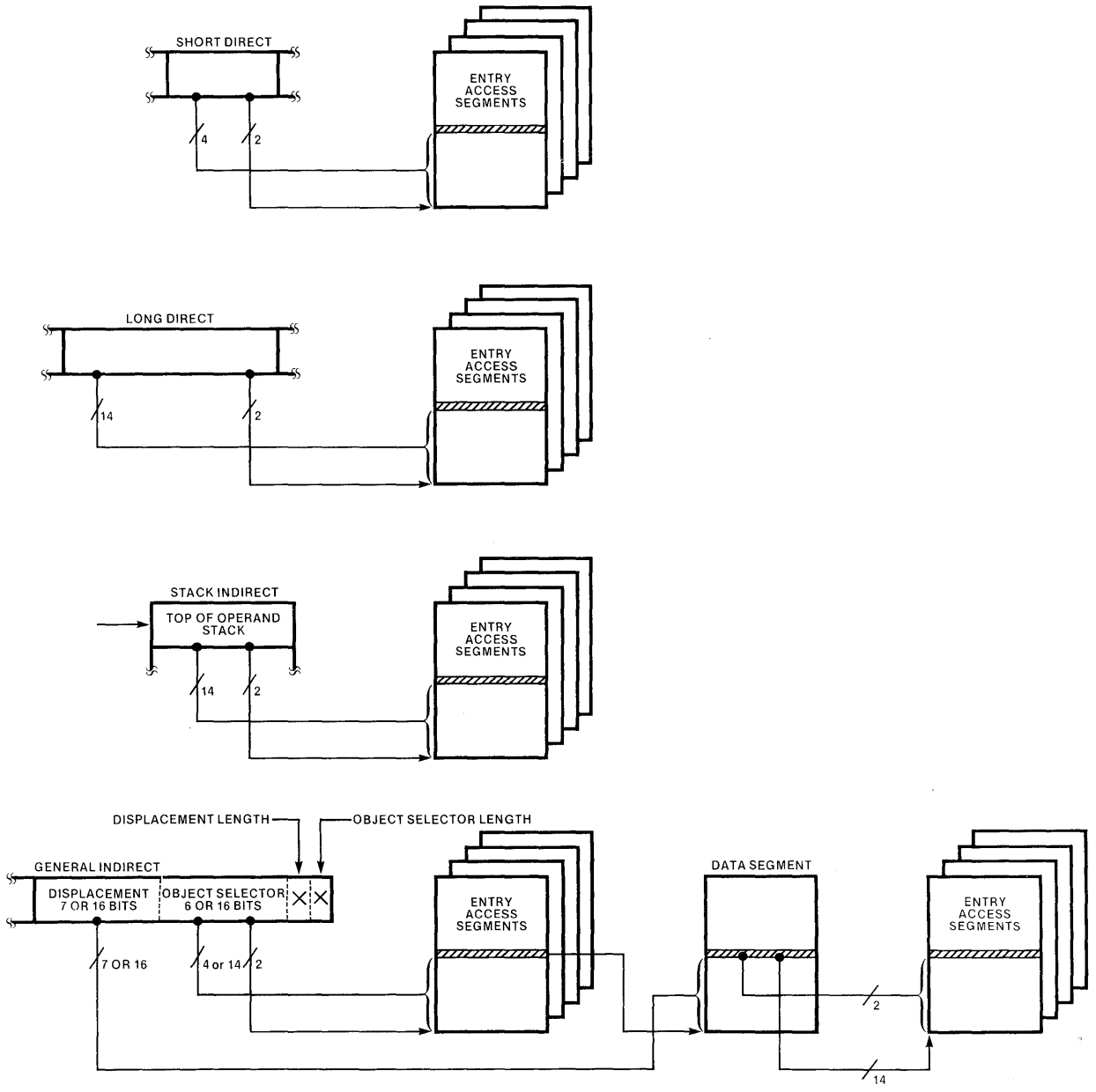
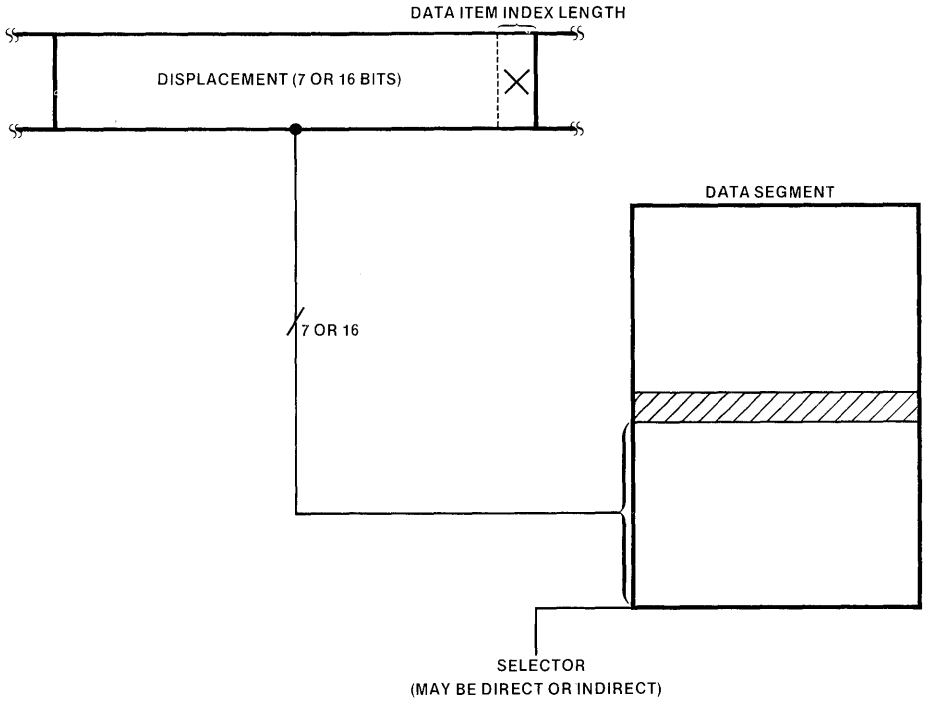
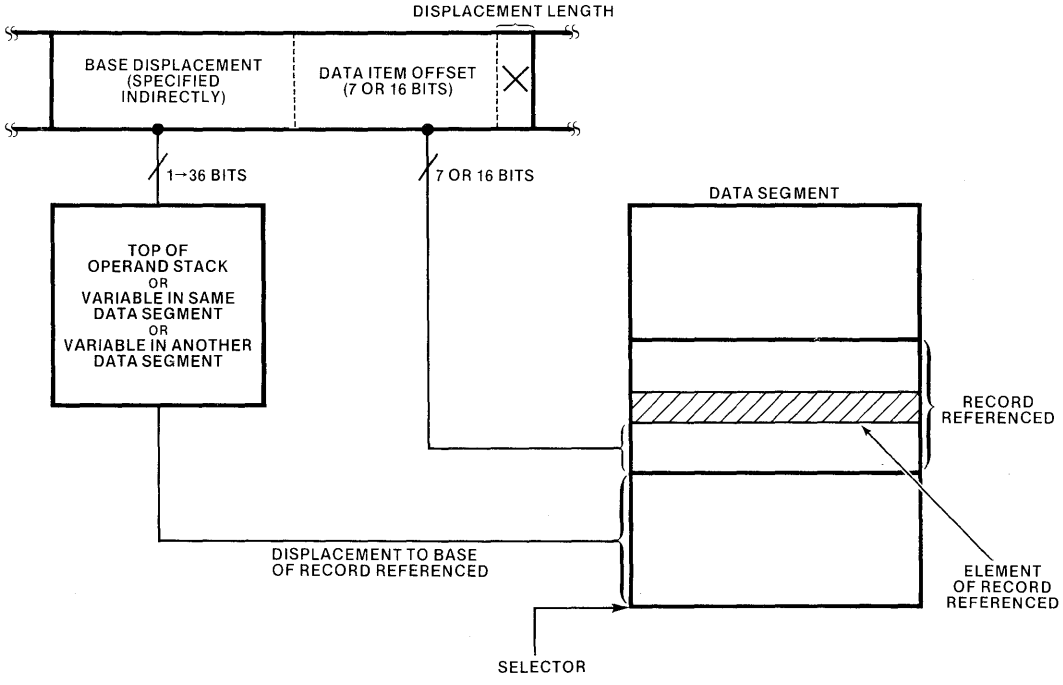



**Figure 7. Modes of Selector Generation**

171873-7

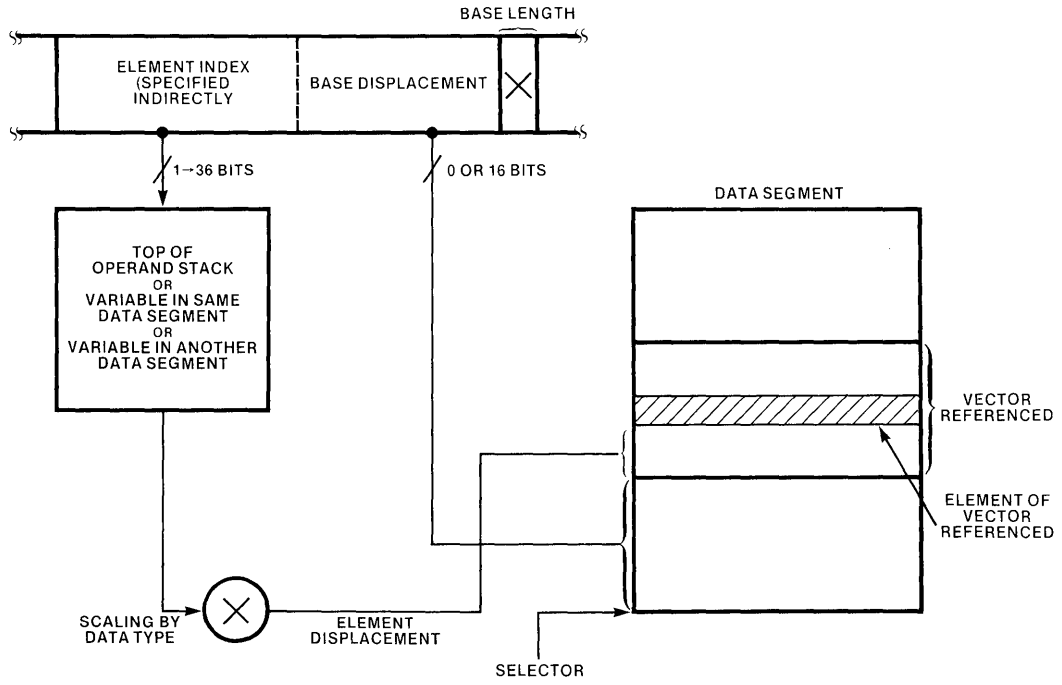



A. SCALAR DATA REFERENCE MODE 171873-8

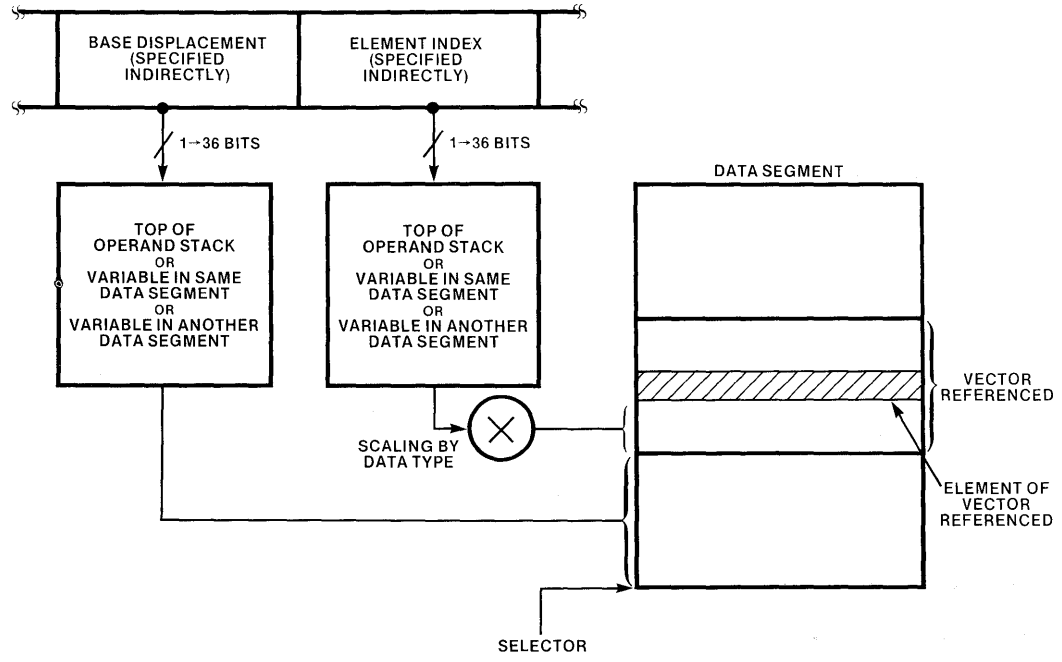



B. RECORD ITEM REFERENCE MODE 171873-9

**Figure 8. Modes of Displacement Generation**




C. STATIC VECTOR ELEMENT REFERENCE MODE 171873-10




D. DYNAMIC VECTOR ELEMENT REFERENCE MODE 171873-11

**Figure 8. Modes of Displacement Generation (Cont'd.)**

## HARDWARE ERROR DETECTION FOR iAPX 432 PROCESSORS

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. A master operates in the normal manner. A checker places all output pins that are being checked in the high-impedance state. Thus, those pins which are to be checked on a master and checker are parallel-connected, pin for pin, so the checker can compare its master's output values with its own. Any comparison error causes the checker to assert HERR/ (refer to Figure 9).

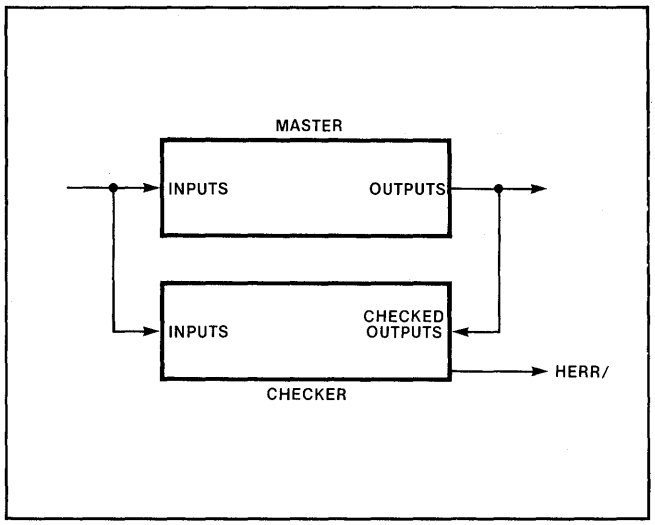



**Figure 9. Hardware Error Detection** 171873-12

## iAPX 432 INFORMATION STRUCTURE

The following section presents the information structure for an iAPX 432 system and includes a discussion of memory system requirements, physical addressing, data formats, and data representation. 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 10.

### Data Positioning

The data operand types shown in Figure 10 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

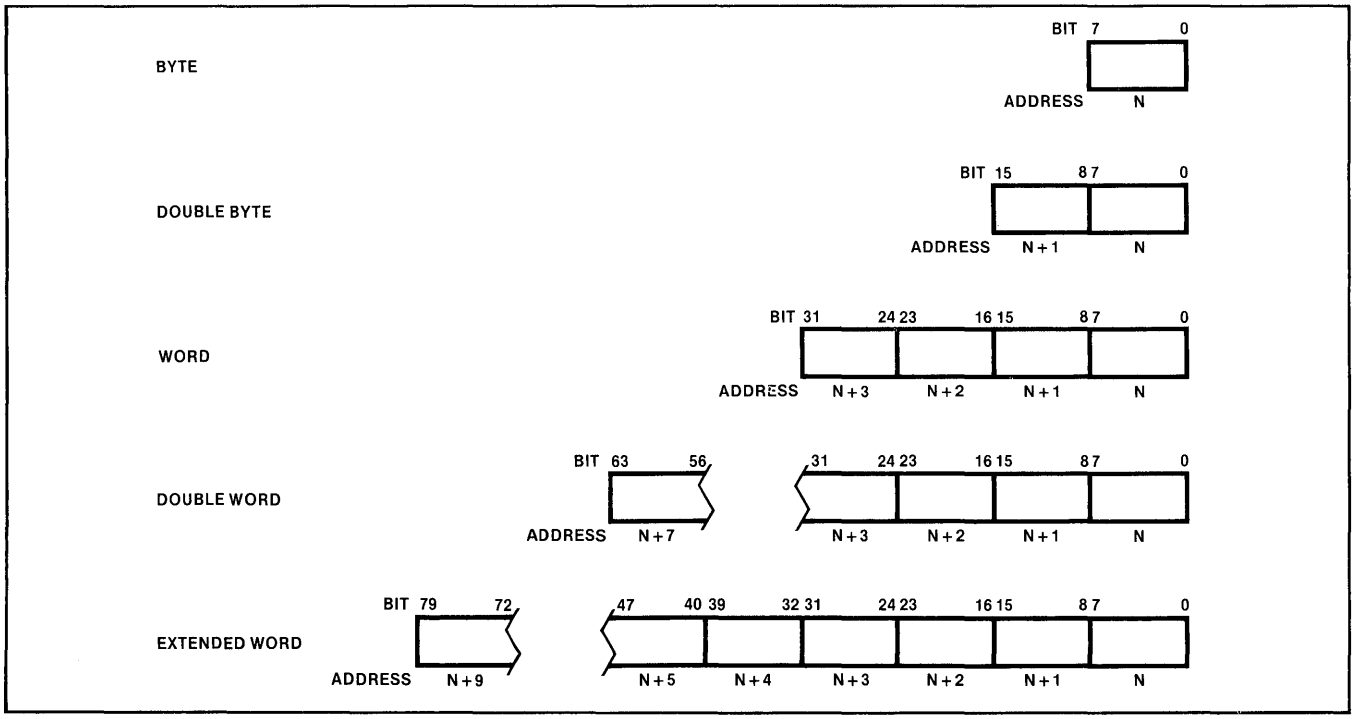



**Figure 10. Basic iAPX 432 Data Lengths**

171873-13

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 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.

## PROCESSOR PACKET BUS DEFINITION

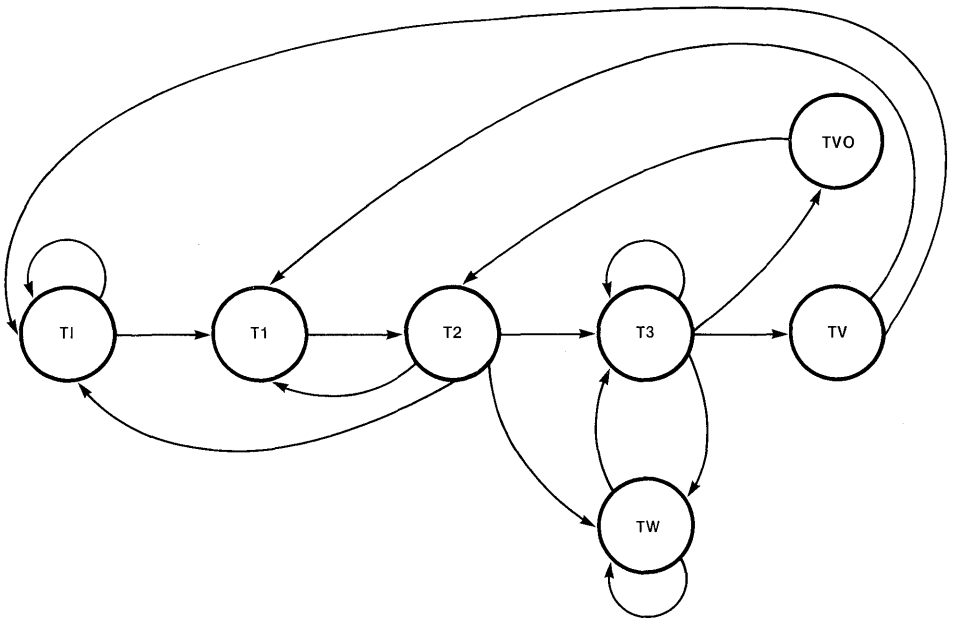
This section describes and defines the significance of the 19 signal lines that make up the Processor Packet bus, and the general scheme by which timing relationships on these lines are derived. Although this section defines all legal bus activities, the processors do not necessarily perform all allowed activities. Slaves to the Processor Packet bus must support all state transitions to ensure compatibility (refer to Figure 11 for Packet bus states).

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.




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

**Figure 11. Processor Packet Bus State Diagram**

17 1873-14

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)

During 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

**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<br>001 - 2 Bytes<br>010 - 4 Bytes<br>011 - 6 Bytes<br>100 - 8 Bytes<br>101 - 10 Bytes<br>110 - 16 Bytes*<br>111 - 32 Bytes* | ACD 15 = 0:<br>00-Inst Seg Access<br>01-Stack Seg Access<br>10-Context Ctl Seg Access<br>11-Other |
| 1 - Other | 1 - Write | 1 - RMW | * Not implemented | ACD 15 = 1:<br>00-Reserved<br>01-Reserved<br>10-Reserved<br>11-Interconn Register |

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:

- Assert IPC significance on ICS for the arrival of an IPC or reconfiguration request.
- 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)
- 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 indicate 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 GDP.

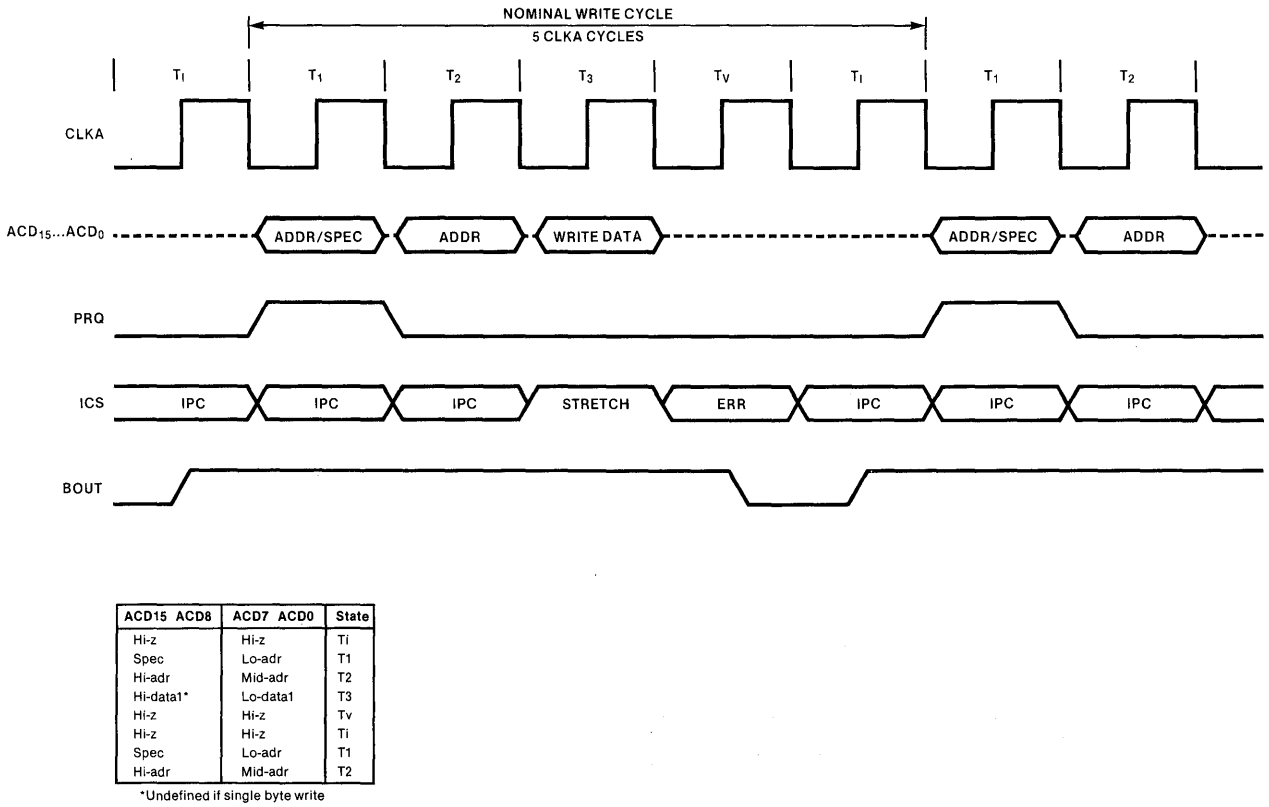
Any agent connected to the Processor Packet bus is recognized as either a processor or a slave. Examples of processors are the GDP and the IP. A memory system provides an example of a slave.

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 |




| 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-z | Hi-z | Tv |
| Hi-z | Hi-z | Ti |
| Spec | Lo-adr | T1 |
| Hi-adr | Mid-adr | T2 |

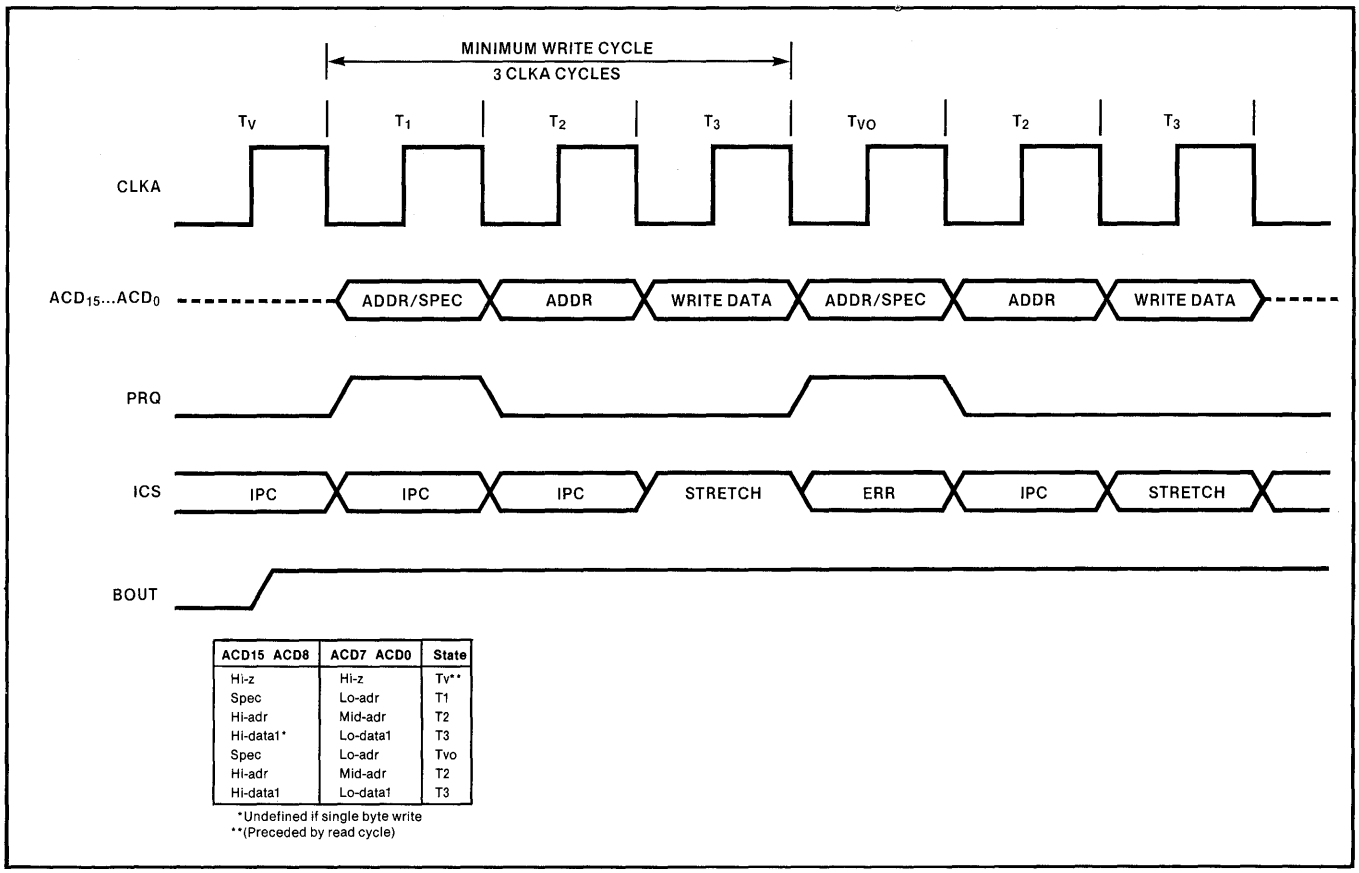
*Undefined if single byte write

**Figure 12. Nominal Write Cycle Timing**

171873-15

**Table 5. iAPX 432 Component Signaling Scheme**

| | Processor | | Slave | |
|---|---|---|---|---|
| Inputs Sampled | ACD:<br>Others: | ↓CLKA<br>↑CLKA | All: | ↑CLKB |
| Outputs Driven | All (except BOUT):<br><br>BOUT: | ↓CLKA<br><br>↑CLKA | ACD:<br>Others: | ↓CLKB<br>↑CLKB |

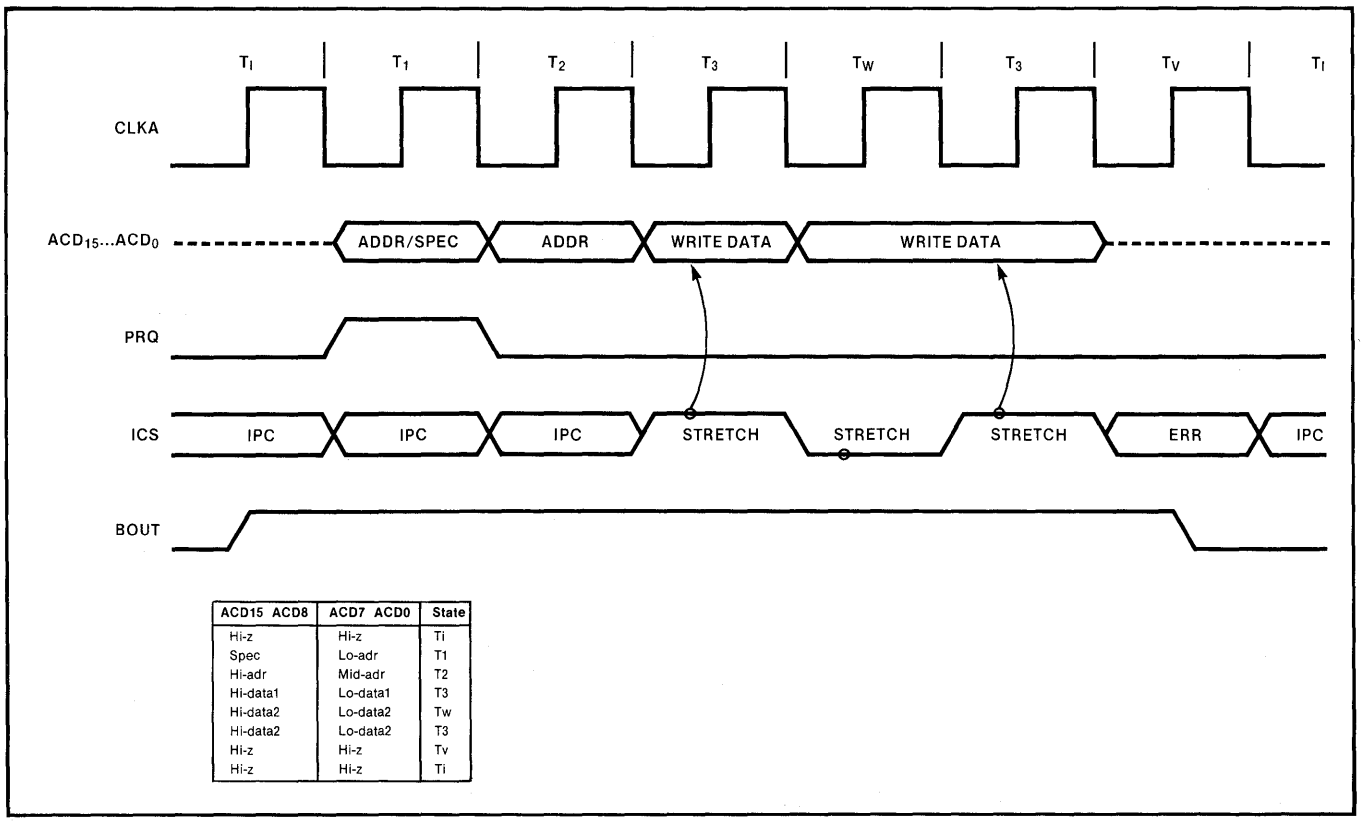



| ACD15 ACD8 | ACD7 ACD0 | State |
|---|---|---|
| Hi-z | Hi-z | Tv** |
| Spec | Lo-adr | T1 |
| Hi-adr | Mid-adr | T2 |
| Hi-data1* | Lo-data1 | T3 |
| Spec | Lo-adr | Tvo |
| Hi-adr | Mid-adr | T2 |
| Hi-data1 | Lo-data1 | T3 |

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

**Figure 13. Minimum Write Cycle Timing**

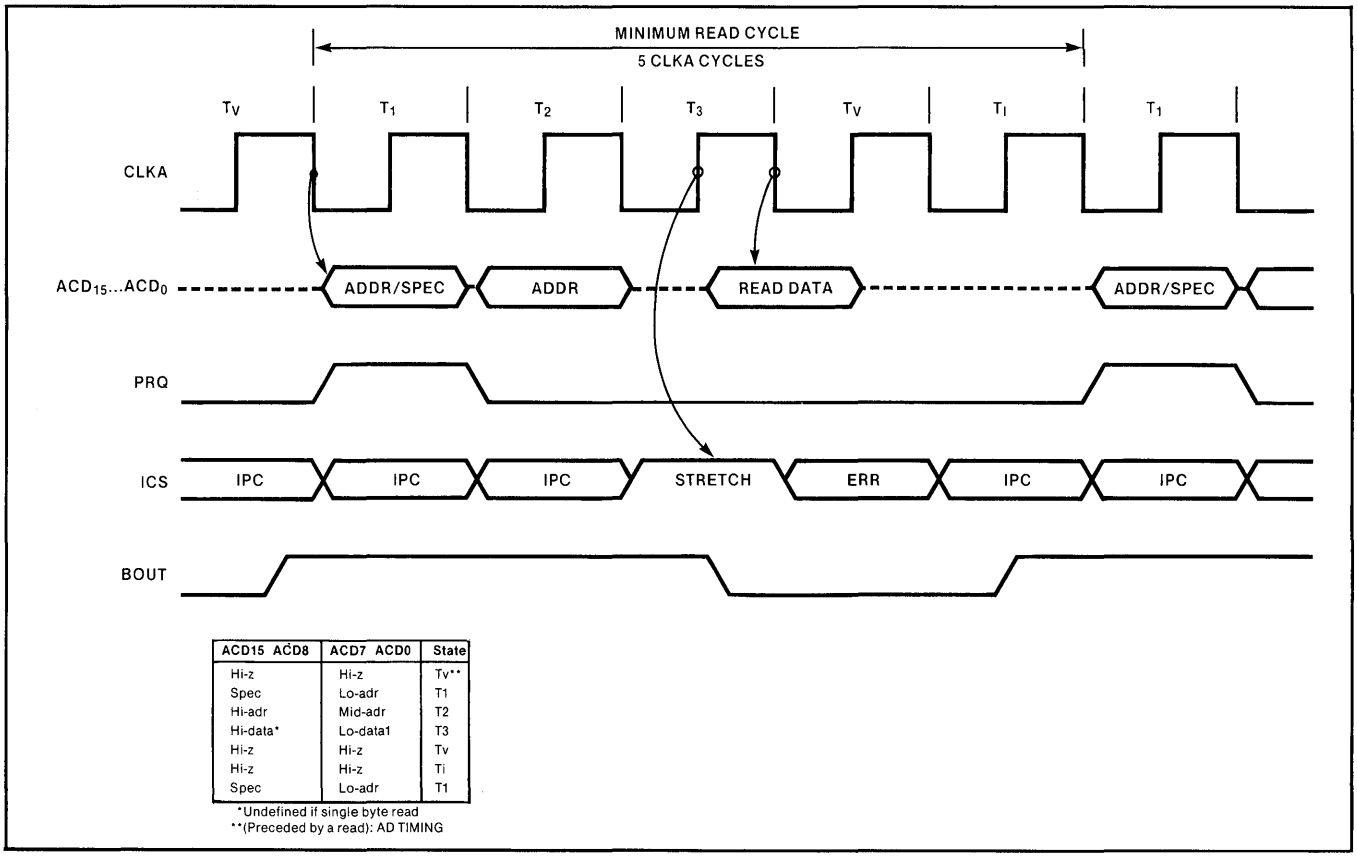
171873-16




| 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**

171873-17

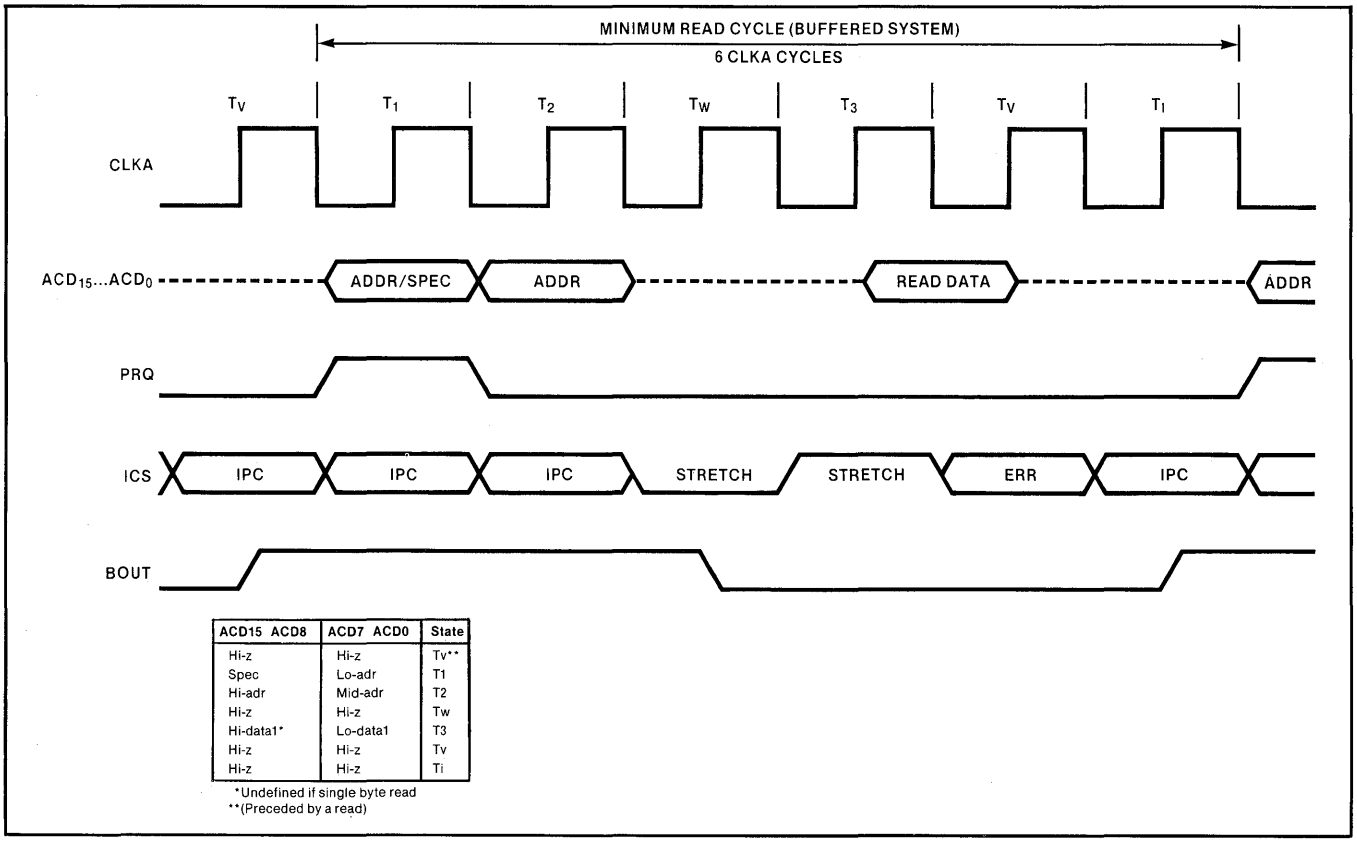



| ACD15 ACD8 | ACD7 ACD0 | State |
|---|---|---|
| Hi-z | Hi-z | Tv** |
| Spec | Lo-adr | T1 |
| Hi-adr | Mid-adr | T2 |
| Hi-data* | Lo-data1 | T3 |
| Hi-z | Hi-z | Tv |
| Hi-z | Hi-z | Ti |
| Spec | Lo-adr | T1 |

*Undefined if single byte read
**(Preceded by a read): AD TIMING

**Figure 15. Minimum Read Cycle (Not Buffered)**

171873-18

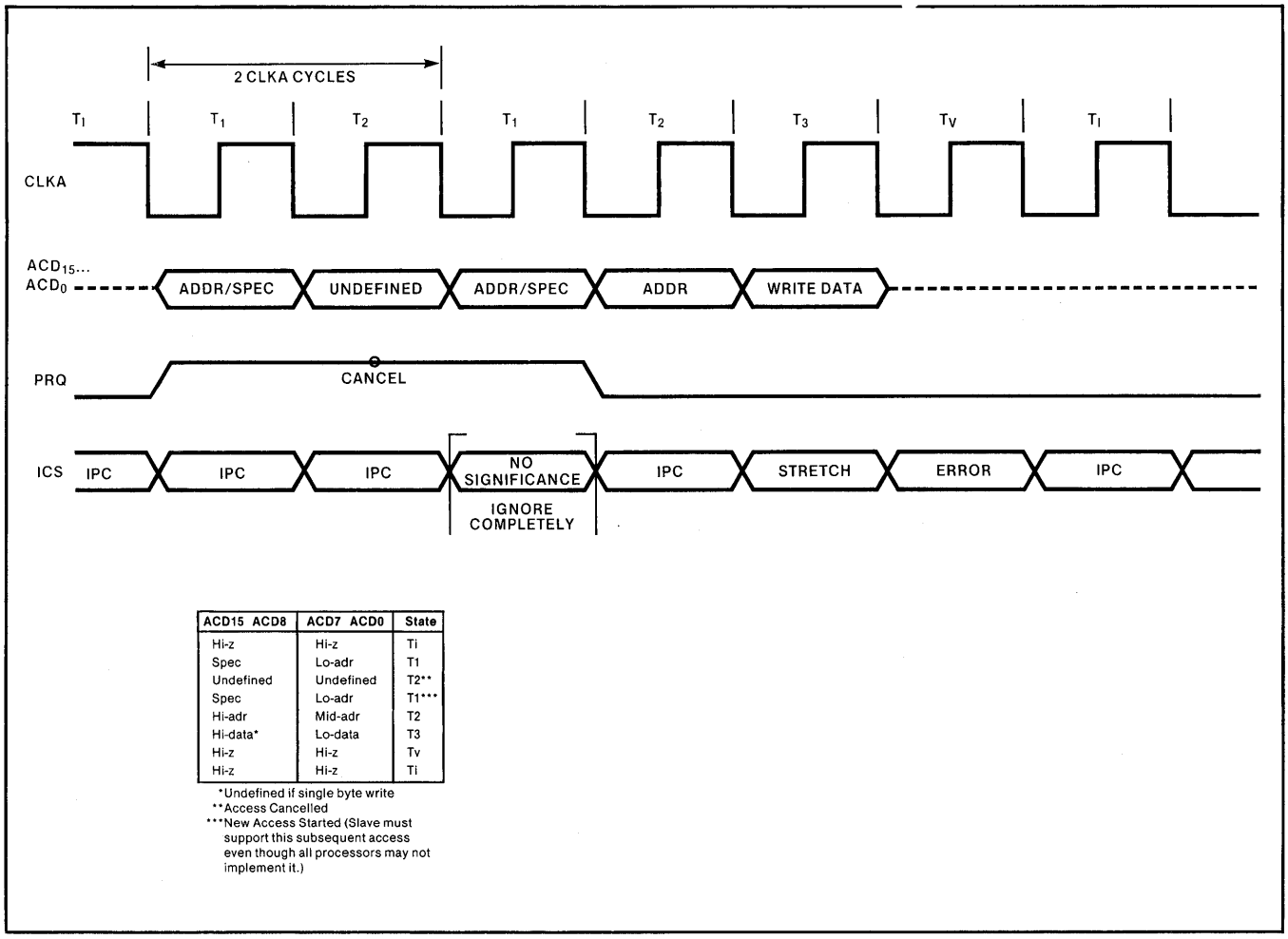



| ACD15 ACD8 | ACD7 ACD0 | State |
|---|---|---|
| Hi-z | Hi-z | Tv** |
| Spec | Lo-adr | T1 |
| Hi-adr | Mid-adr | T2 |
| Hi-z | Hi-z | Tw |
| Hi-data1* | Lo-data1 | T3 |
| Hi-z | Hi-z | Tv |
| Hi-z | Hi-z | Ti |

*Undefined if single byte read
**(Preceded by a read)

**Figure 16. Minimum Read Cycle (Buffered System)**

171873-19




| 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.)

**Figure 17. Minimum Faulted Access Cycle**

171873-20

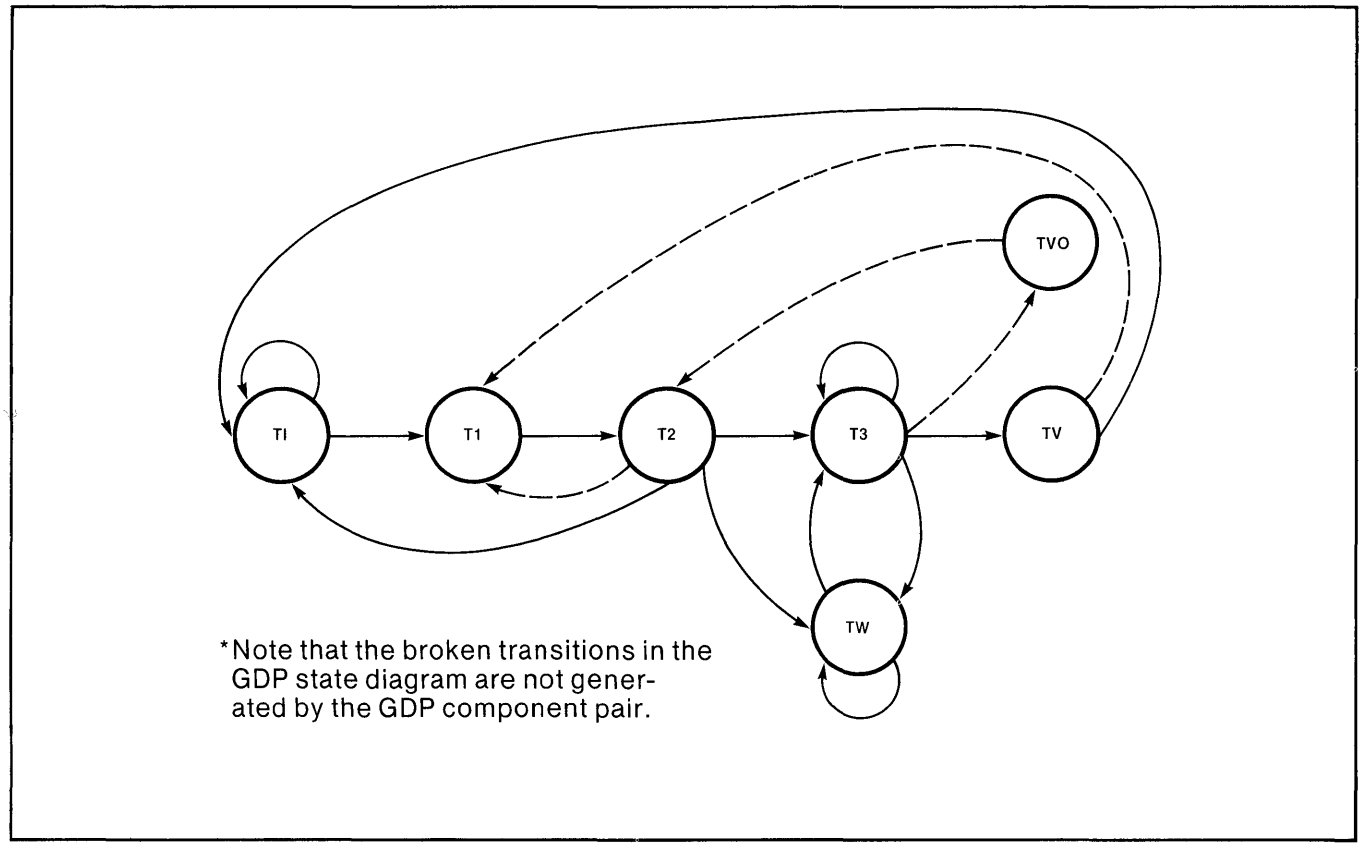



**Figure 18. GDP State Diagram** 171873-21

## 43201 PIN DESCRIPTION

### Processor Packet Bus Group

### $ACD_{15}$—$ACD_0$ (Address/Control/Data lines, Inputs, high asserted)

The Processor Packet bus Address/Control/Data lines are the basic communication path between the GDP and its environment. These lines are always inputs to the 43201 and are driven by either the 43202 or the external environment. Note that the 43201 must receive the specification byte from the 43202 during T1 of a bus transaction (Figure 11). As a result, the ACD receivers must be capable of slave timing as well as processor timing. (See Processor Packet bus timing relationships for definition of processor and slave timing).

### PRQ (Processor Packet bus Request, Input, high asserted)

The PRQ input is used to initiate a transaction between the GDP and the bus interface. PRQ is normally held low by the 43202 whenever there is no transaction. PRQ is asserted high during the first cycle of a bus transaction and returns low during the second cycle if the transaction is to be completed. The GDP may cancel a bus transaction by asserting PRQ high (instead of returning it low) during the second cycle of the transaction. The GDP will cancel a transaction if a bounds or access rights violation for the transaction has been detected. PRQ is sampled on the rising edge of CLKA.

### ICS (Interconnect Status, Input, high asserted)

The ICS input is continually monitored by the 43201 to determine the state of bus transactions. The interpretation of ICS depends on the present cycle of a bus transaction and will indicate one of the following states:

1. Interprocessor communication (IPC) message waiting.
2. Input data invalid, a stretched access.
3. Output data not taken, a stretched access.
4. Bus error in external environment.

## Intra-GDP Bus Group

### $UI_{15}...UI_0$ (Microinstruction Bus lines, Outputs, high asserted)

These lines are used to transmit microinstructions from the 43201 to the 43202. These pins are high impedance in the checker state (Refer to Hardware Error Detection Group). They are monitored by the hardware error checking logic.

### $IS_6...IS_0$ (Interchip Status lines, Inputs, high asserted)

The 43201 receives information pertaining to interchip microprogram status from the 43202 over these lines.

## System Group

### FATAL/ (Fatal, Output, low asserted)

FATAL/ is asserted by the 43201 under microcode control and is used by the GDP microcode to indicate to the system that the GDP cannot continue due to grossly incorrect information structures in memory. FATAL/ is synchronously asserted low and remains low until the processor is initialized. FATAL/ is not affected by the hardware checking logic.

### ALARM/ (Alarm signal, Input, low asserted)

The ALARM/ input signals the occurrence of an unusual system-wide condition (such as power fail). The 43201 does not respond to ALARM/ until it has completed execution of the current 432 instruction, i.e., if any instruction is currently under execution. ALARM/ is active low and is sampled on the rising edge of CLKA.

### INIT/ (Initialization, Input, low asserted)

The INIT/ pin is used to establish initialization. INIT/ must be asserted low for at least 10 CLKA cycles before the initial state is reached to allow time for the 43201 to begin execution of a microcode sequence that initializes all of the 43201 and 43202 internal registers. Once this initialization sequence has been completed, normal operation begins.

### CLR/ (Clear, Input, low asserted)

Assertion of CLR/ results in a microprogram trap which causes the GDP 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 GDP to the same state as INIT/ assertion does). The IPC will not be serviced for at least five clock cycles following CLR/ assertion.

If CLR/ is continuously asserted low for more than one clock cycle, it is ignored during alternate clock cycles (beginning with the second clock cycle) of continuous CLR/ low assertion.

## Hardware Error Detection Group

### MASTER (Master, Input, high asserted)

The MASTER pin is used to place the processor in either master or checker mode. MASTER is sampled during initialization (INIT/ asserted). If MASTER is asserted throughout initialization, the 43201 functions normally and drives the microinstruction bus. If MASTER is low throughout initialization, microinstruction bus signals $ul_{15}$—$ul_0$ go to their high-impedance state. A 43201 checker does not drive the microinstruction bus; rather, it monitors the bus and compares the data on the bus to its internally generated result, signalling disagreement on its HERR/ line. This hardware error detection capability on the 43201 is provided mainly for test purposes. MASTER should be tied to $V_{CC}$ for normal operation and tied low to enable hardware error detection and disable the bus ($ul_{15}$—$ul_0$) outputs.

### HERR/ (Hardware Error, Output, low asserted)

HERR/ is a signal produced by the 43201 to indicate disagreement between the data appearing on the micro-instruction bus ($ul_{15}$—$ul_0$) and the internally generated result of the 43201. HERR/ is asserted low when disagreement occurs and is valid during CLKA. HERR/ can drive one low power Schottky load.

## Clock Group

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

Clock A (CLKA) provides the basic timing reference for the 43201. CLKB overlaps CLKA by nominally 1/4 cycle (90 degrees phase shift). All

external signals are referenced to CLKA. Refer to the AC Electrical Characteristics for exact statement of timing relationships.

## Testing Input

### RDROM/ (Read ROM, Input, low asserted)

The RDROM/ input line is used to force a sequential read of Read-Only-Memory. If RDROM/ is low when INIT/ goes high, the 43201 goes into a special diagnostic mode. In this mode, with RDROM/ held low, the 43201 microinstruction sequencer steps through the 43201 microprogram ROM, sequentially displaying (but not executing) the 43201 microprogram on the $uI_{15}$—$uI_0$ lines. The RDROM/ feature is useful for testing. RDROM/ should be tied to $V_{CC}$ for normal operation and tied low for testing .

## Power and Ground Connections

### $V_{CC}$ (4 pins)

These pins supply +5 V±10 % referenced to GND pins.

### GND (5 pins)

These pins supply ground reference for the 43201.

### $V_{BB}$ (Internally Generated)

This pin is connected to the substrate bias voltage of the 43201. An external low leakage 1 microfarad capacitor rated at 5 volts or greater should be used to bypass $V_{BB}$. $V_{BB}$ is a negative voltage.

### N.C. (No Connection, 4 pins)

## 43202 PIN DESCRIPTION

## 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 GDP 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 GDP for an access, or
- They may transfer data (either direction).

When the 43202 is in checker mode, the ACD pins are monitored by the hardware error checking 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 GDP 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 43202 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 logic. PRQ is in a high impedance state when the 43202 is in checker mode (see MASTER description).

### ICS (Interconnect Status, Input, high asserted)

ICS is an indication to the 43202 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.

### **BOUT** (Enable Buffers for Output, Output, high asserted)

BOUT is used to control external bus transceivers to buffer the 43201, 43202 from the Processor Packet bus load. Though not required, the use of buffers may be desired in systems with heavy loading. BOUT is asserted when information is to leave the 43202 on the ACD lines. BOUT is not checked by the hardware error detection logic.

## Intra-GDP Bus Group

### $ul_{15}$—$ul_0$ (Microinstruction Bus lines, Inputs, high asserted)

The $ul_{15}$—$ul_0$ input lines provide the 43202 with microinstruction information sent from the 43201.

### $IS_6$—$IS_0$ (Interchip Status lines, Outputs, high asserted)

The $IS_6$—$IS_0$ lines drive interchip microprogram status information from the 43202 to the 43201. $IS_6$—$IS_0$ are not checked by the hardware error detection logic.

## System Group

### PCLK/ (Processor Clock, Input, low asserted)

PCLK/ is asserted to change the state of two processor timers. The affected timers are called the system timer and the service timer. Assertion of PCLK/ for one cycle causes the system timer to increment and the service timer to decrement. Assertion of PCLK/ for more than one cycle causes the system timer to be cleared and decrements the service timer. For proper operation PCLK/ must be unasserted for at least four clock cycles before being asserted. PCLK/ is synchronous with respect to CLKA, but is generally unrelated to other interface timings.

### CLR/ (Clear, Input, low asserted)

Assertion of CLR/ results in a microprogram trap which causes the GDP 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 GDP to the same state as INIT/ assertion does). The IPC will not be serviced for at least five clock cycles following CLR/ assertion.

If CLR/ is continuously asserted low for more than one clock cycle, it is ignored during alternate clock cycles (beginning with the second clock cycle) of continuous CLR/ low assertion.

## Hardware Error Detection Group

### MASTER (Master, Input, high asserted; 25k nominal pullup on-chip)

The MASTER input determines whether the 43202 is to function as a master or a checker. In master mode, the 43202 functions normally and drives all of its outputs. In checker mode, $ACD_{15}$—$ACD_0$ and PRQ enter the high impedance state and BOUT is unconditionally low. A 43202, whether master or checker, monitors the $ACD_{15}$—$ACD_0$ and PRQ lines and compares the data on them to its internally generated result, signalling disagreement on its HERR/ line. For normal operation, MASTER may be either left alone or tied high. MASTER must be tied low to disable the $ACD_{15}$—$ACD_0$ and PRQ outputs.

### HERR/ (Hardware Error, Open Drain Output, low asserted)

HERR/ is asserted low by the 43202 to indicate disagreement between the data appearing on the $ACD_{15}$—$ACD_0$ and PRQ pins and the internally generated result of the 43202. HERR/ is valid during CLKA and can normally be asserted by a 43202 every clock cycle. HERR/ is prevented from being asserted low during any clock cycle following a clock cycle in which a CLR/ low assertion is recognized by the 43202. HERR/ requires an external 2.2k ohm nominal pullup resistor.

## Clock Group

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

Clock A (CLKA) provides the basic timing reference for the 43202. Clock B (CLKB) overlaps CLKA by nominally 1/4 cycle (90 degrees phase shift). Refer to the ac electrical characteristics for exact statement of timing relationships. All external signals are referenced to CLKA.

## Power and Ground Connections

### $V_{CC}$ (Power Supply, 4 pins)

These pins supply +5 V ±10%, referenced to GND pins.

### GND (Ground, 5 pins)

These pins supply ground reference for the 43202.

### N.C. (No Connection, 7 pins)

## INSTRUCTION SET SUMMARY

Refer to Table 14 for the iAPX 432 General Data Processor operator set summary.

## 43201/43202 ELECTRICAL SPECIFICATIONS

Tables 6 through 13 and Figures 19 through 27 provide appropriate timing diagrams and tables to represent the complete electrical specifications for both the 43201 and 43202 components.

**Table 6. iAPX 43201 Electrical Specification**

| 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* | −1V to +7V |
| Power Dissipation | 2.5 Watts |
| *43201 $V_{BB}$ Pin with respect to GND | −5V to 0V |

**Table 7. iAPX 43201 Electrical Specification**

DC Characteristics

VSS = 0 Volts, VCC = 5 Volts ± 10% Ta = 0° C to 70° C

| Symbol | Description | Min | Max | Units |
|---|---|---|---|---|
| Vili | Input Low Voltage IS6...IS0 | −0.3 | +0.7 | V |
| Vihi | Input High Voltage IS6...IS0 | 3.0 | VCC+0.5 | V |
| 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 |
| Vol | Output Low Voltage (Microinstruction Lines) (Iol = −0.1 mA) | 0 | 0.35 V | |
| Voh | Output High Voltage (Microinstruction Lines) (Ioh = 0.1 mA) | 3.25 | VCC | V |
| Vol | Output Low Voltage (Iol = 2.0 mA) | — | 0.45 | V |
| Voh | Output High Voltage (Ioh = −400 uA) | 2.4 | VCC | V |
| Icc | Power Supply Current (Sum of all VCC Pins) | — | 400 | mA |
| Iil | Input Leakage Current | — | ±10 | uA |

**Table 7. iAPX 43201 Electrical Specification (Cont'd.)**

DC Characteristics

VSS = 0 Volts, VCC = 5 Volts ± 10% Ta = 0° C to 70° C

| Symbol | Description | Min | Max | Units |
|---|---|---|---|---|
| Io | Output Leakage Current | — | ±10 | uA |
| Iol | @0.45 Vol | | | |
| | HERR/ | — | .4 | mA |
| | FATAL/ | — | 4 | mA |
| | OTHER | — | 2 | mA |
| Ioh | @2.4 Voh | — | −0.1 | mA |

* For operation at 5 MHz or slower, the 43201 may be operated with Vihc minimum of 2.7 Volts.

**Table 8. iAPX 43201 AC Characteristics**

VCC = 5 ± 10% Ta = 0° C to 70° C

| Symbol | Description | 8 MHz Min | 8 MHz Max | 5 MHz Min | 5 MHz Max | Unit |
|---|---|---|---|---|---|---|
| tcy | Clock Cycle Time | 125 | 1000 | 200 | 1000 | nsec. |
| tr, tf | Clock Rise and Fall Time | 0 | 10 | 0 | 10 | nsec. |
| t1, t2, t3, t4 | Clock Pulse Widths | 26 | 250 | 45 | 250 | nsec. |
| tdc | Signal to Clock Set-up Time | 5 | — | 5 | — | nsec. |
| tcd | Clock to Signal Delay Time | — | 55 | — | 85 | nsec. |
| tis | Init to Signal Hold Time | 15 | — | 20 | — | nsec. |
| tie | Init enable Time | 10 | — | 10 | — | tcy |
| tdh | Clock to Signal Hold Time | 25 | — | 35 | — | nsec. |
| $t_{OH}$ | Clock to Signal Output Hold Time | 15 | — | 20 | — | nsec. |
| tsi | Signal to INIT/ Set-up Time | 10 | — | 10 | — | nsec. |
| tuif | Microinstruction Bus Float Time | 0 | — | 0 | — | nsec. |

The above specifications are subject to the following definitions and test conditions:

1. Note that tcy=t1 + t2 + t3 + t4 + 2*tr + 2*tf.
2. Pins under consideration were subjected to the following purely capacitive loading:
   C1 = 25 pF on HERR/
   C1 = 50 pF on ul15...ul0, IS6...IS0
   C1 = 70 pF on all remaining pins.
3. All timings are measured with respect to the switching level of 1.5 Volts. The switching point of CLKA and CLKB is referenced to the 1.8 Volt level.
4. CLKA and CLKB must be continuously applied for the 43201 to retain its state.

**Table 9. iAPX 43201 Capacitance**

| Symbol | Parameter | Typical | Unit |
|---|---|---|---|
| Cin | Input Capacitance | 6 | pF |
| Cout | Output Capacitance | 12 | pF |
| Conditions: | fc=1 MHz, Vin=0V, VCC=5V, Ta=25° C<br>Outputs in High Impedance state | | |

**Table 10. iAPX 43202 Electrical Specification**

| 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 | −1V to +7V |
| Power Dissipation | 2.5 Watts |

**Table 11. iAPX 43202 Electrical Specification**

DC Characteristics

VSS = 0 Volts, VCC = 5 Volts ± 10% Ta = 0° C to 70° C

| Symbol | 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 |
| Vili | Input Low Voltage<br>ul15...ul0 | −0.3 | +0.7 | V |
| Vihi | Input High Voltage<br>ul15...u10 | 3.0 | VCC+0.5 | V |
| Vol | Output Low Voltage<br>(Iol = 4.0 mA) ACD15...ACD0, PRQ<br>(Iol = 8.0 mA) BOUT, HERR/ | — | 0.45 | V |
| Voh | Output High Voltage<br>(Ioh = −800 uA) | 2.4 | VCC | V |
| Voli | Output Low Voltage IS6...IS0<br>Ioli = 0.1 mA | — | 0.35 | V |
| Vohi | Output High Voltage IS6...IS0<br>Iohi = 0.1 mA | 3.25 | — | V |
| Icc | Power Supply Current<br>(Sum of all VCC Pins) | — | 455 | mA |
| Ili | Input Leakage Current<br>except MASTER | — | ±10 | uA |
| Ilim | Input Leakage on MASTER | — | −400 | uA |
| Ilo | Output Leakage Current<br>Vo = 0.45V .. VCC | — | ±10 | uA |

* For operation at 5 MHz or slower, the 43202 may be operated with $V_{ihc}$ minimum of 2.7 Volts.

**Table 12. iAPX 43202 AC Characteristics**

| VCC = 5 ± 10% | | | | | | Ta = 0° C to 70° C |
|---|---|---|---|---|---|---|
| **Symbol** | **Description** | **8 MHz** | | **5 MHz** | | **Unit** |
| | | **Min** | **Max** | **Min** | **Max** | |
| tr, tf | Clock Rise and Fall Time | 0 | 10 | 0 | 10 | nsec. |
| t1, t2, t3, t4 | Clock Pulse Widths | 26 | 250 | 45 | 250 | nsec. |
| tcy | Clock Cycle Time (tcy = t1 + t2 + t3 + t4 + 2*tr + 2*tf) | 125 | 1000 | 200 | 1000 | nsec. |
| tdc | Signal to Clock Set-up 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. |

The timing characteristics given below assume the following loading on output pins. Loading is given in terms of a fixed capacitance plus a DC current load.

| **Pins** | **Loading** |
|---|---|
| HERR/ | 90 pF Iol=8 mA., Open Drain |
| BOUT | 70 pF Iol=8 mA., Ioh=−800 uA |
| PRQ | 70 pF Iol=4 mA., Ioh=−800 uA |
| IS6...IS0 | 50 pF MOS only |
| ACD15...ACD0 | 70 pF Iol=4 mA., Ioh=−800 uA |

All output delays are measured with respect to the falling edge of CLKA except for BOUT. BOUT output delays are measured with respect to the rising edge of CLKA.

All timings are measured with respect to the switching level of 1.5 Volts. The switching point of CLKA and CLKB is referenced to the 1.8V level.

The 43202 is not capable of DC operation. For continuous data and logic state retention the CLKA and CLKB signals must be present.

**Table 13. iAPX 43202 Capacitance**

| **Symbol** | **Parameter** | **Typical** | **Unit** |
|---|---|---|---|
| Cin | Input Capacitance | 6 | pF |
| Cout | Output Capacitance | 12 | pF |
| Conditions: | fc=1 MHz, Vin=0, Vout=0, VCC=5.0 V, Ta=25° C. Outputs in High Impedance state | | |

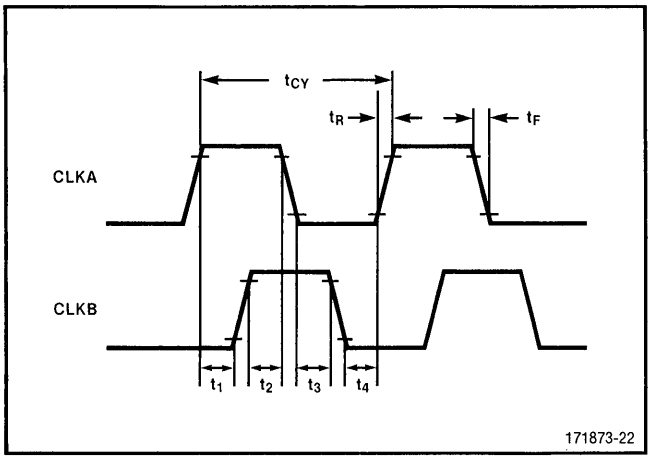



Figure 19. 43201 Clock Input Specification

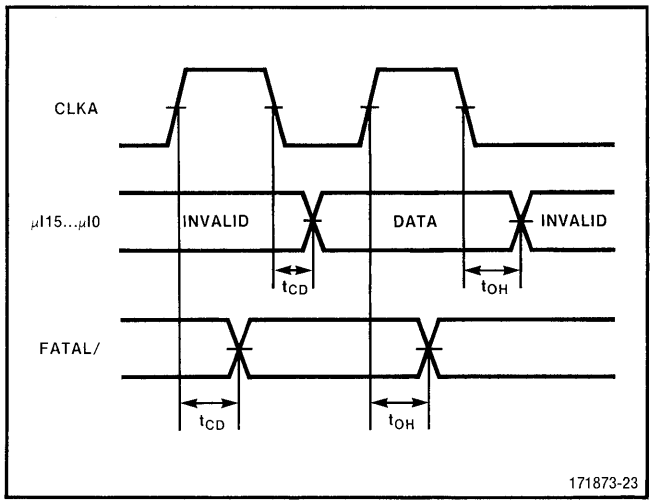



Figure 20. 43201 Output Timing Specification

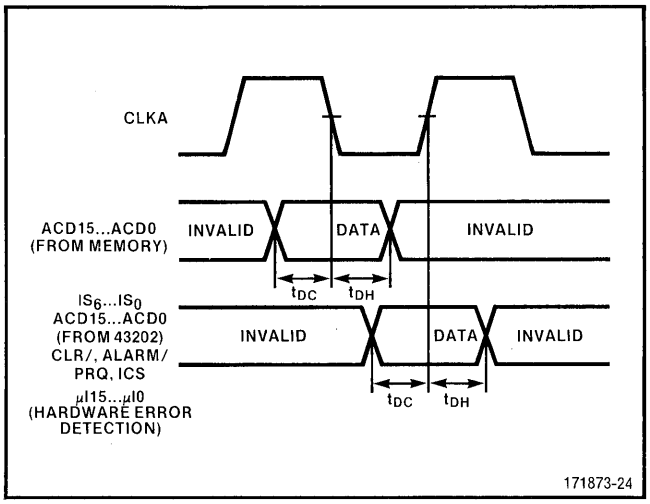



Figure 21. 43201 Input Timing Specification

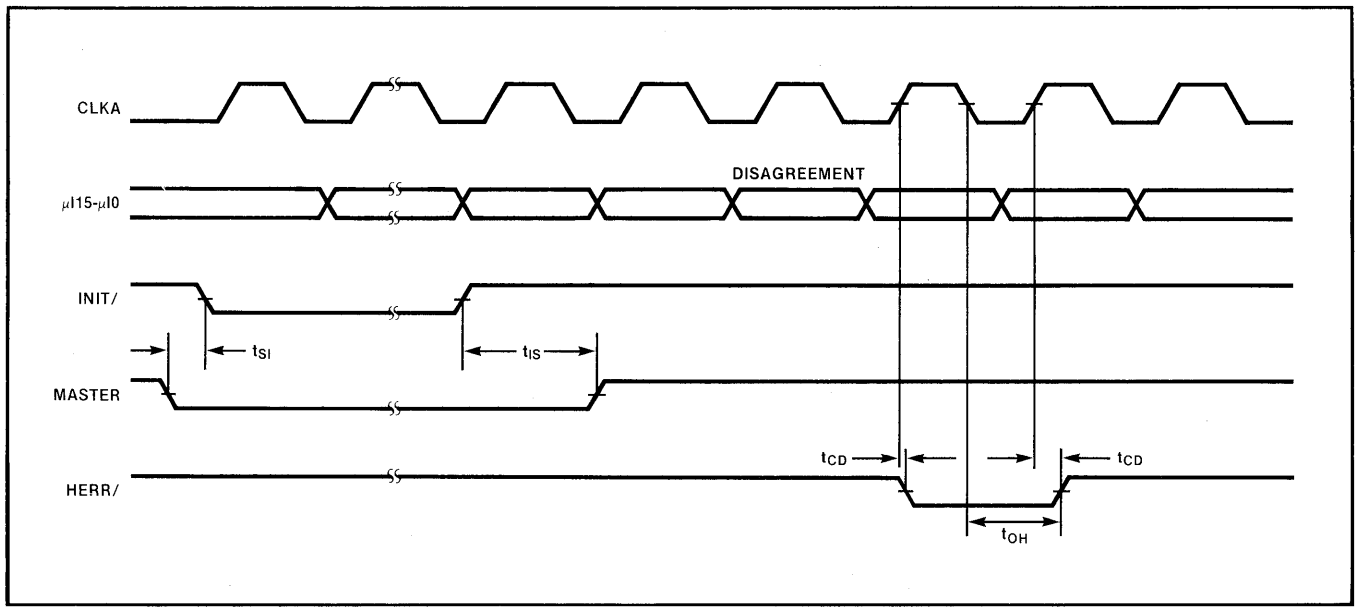



Figure 22. 43201 Hardware Error Detection Timing

171873-25

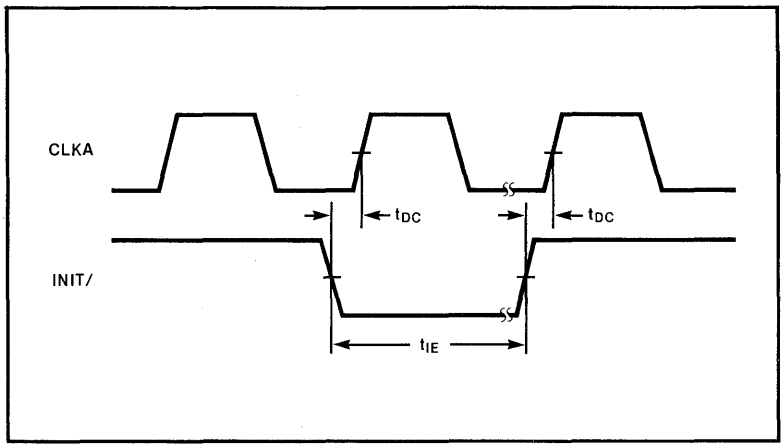



Figure 23. 43201 Initialization Timing 171873-26

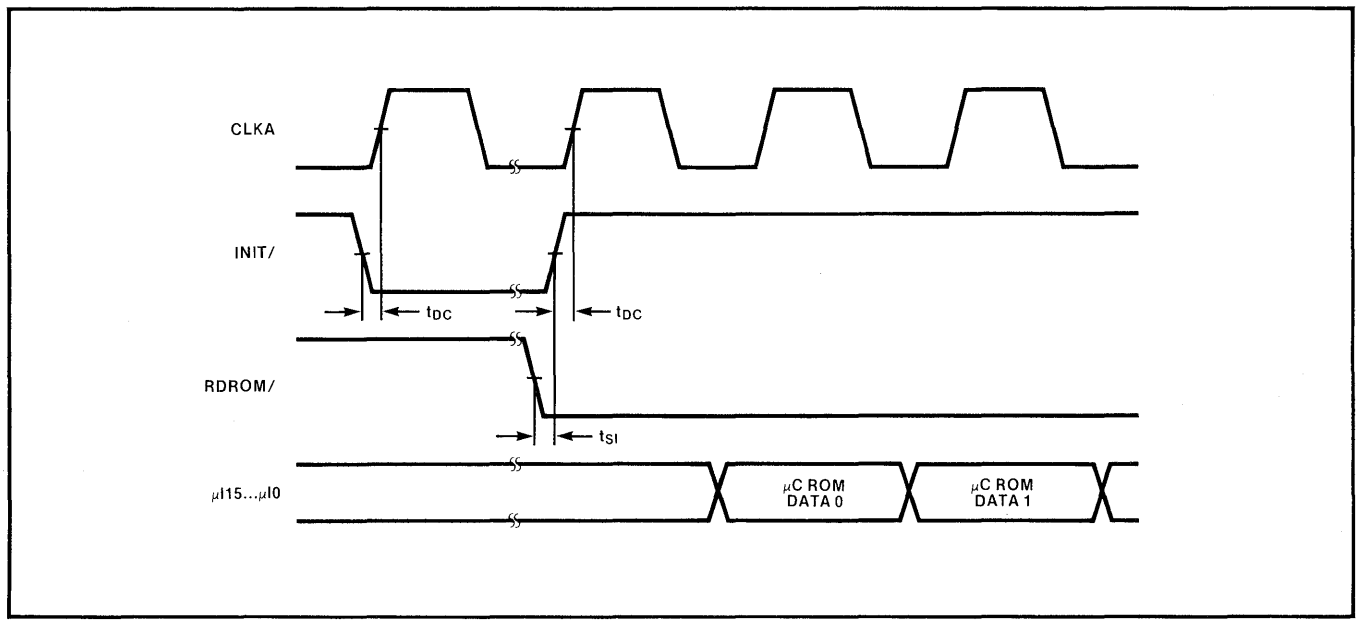



Figure 24. 43201 Microcode Interrogate Timing 171873-27

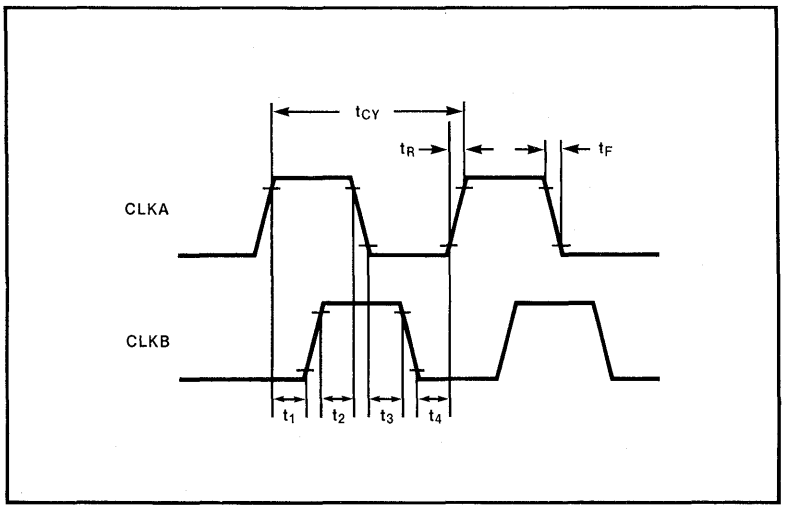



Figure 25. 43202 Clock Input Specification 171873-22

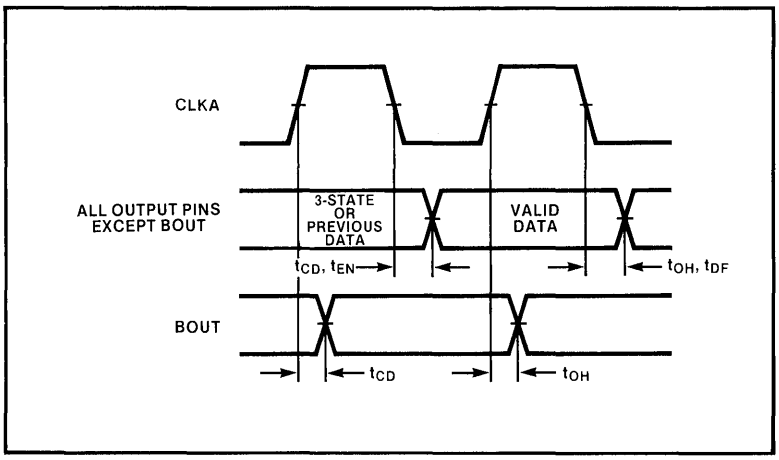



**Figure 26. 43202 Output Timing Specification** 171873-28

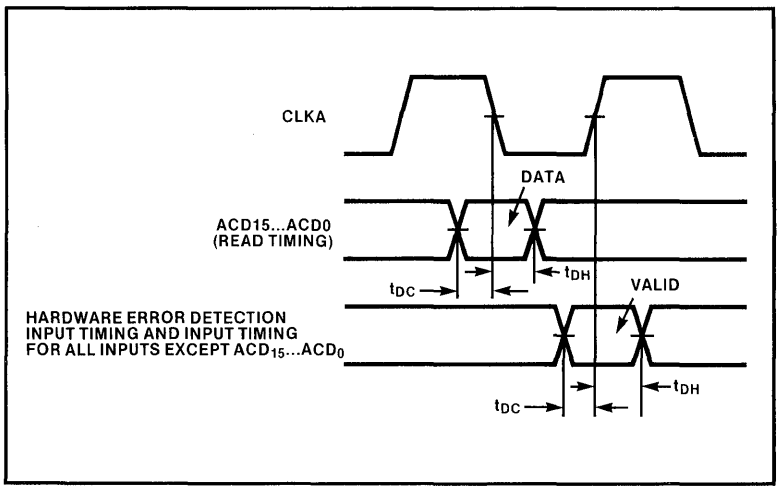



**Figure 27. 43202 Input Timing Specification** 171873-29

## Table 14. General Data Processor Operator Set Summary

| Character Operators | Short-Integer Operators | Integer Operators |
|---|---|---|
| Move Character | Move Short Integer | Move Integer |
| Zero Character | Zero Short Integer | Zero Integer |
| One Character | One Short Integer | One Integer |
| Save Character | Save Short Integer | Save Integer |
| AND Character | Add Short Integer | Add Integer |
| OR Character | Subtract Short Integer | Subtract Integer |
| XOR Character | Increment Short Integer | Increment Integer |
| XNOR Character | Decrement Short Integer | Decrement Integer |
| Complement Character | Negate Short Integer | Negate Integer |
| Add Character | Multiply Short Integer | Multiply Integer |
| Substract Character | Divide Short Integer | Divide Integer |
| Increment Character | Remainder Short Integer | Remainder Integer |
| Decrement Character | Equal Short Integer | Equal Integer |
| Equal Character | Not Equal Short Integer | Not Equal Integer |
| Not Equal Character | Equal Zero Short Integer | Equal Zero Integer |
| Equal Zero Character | Not Equal Zero Short Integer | Not Equal Zero Integer |
| Not Equal Zero Character | Greater Than Short Integer | Greater Than Integer |
| Greater Than Character | Greater Than or Equal Short Integer | Greater Than or Equal Integer |
| Greater Than or Equal Character | Positive Short Integer | Positive Integer |
| Convert Character to Short Ordinal | Negative Short Integer | Negative Integer |
| | Convert Short Integer to Integer | Convert Integer to Short Integer |
| | Convert Short Integer to Temporary Real | Convert Integer to Ordinal |
| | | Convert Integer to Temporary Real |

| Short-Ordinal Operators | Ordinal Operators | Short-Real Operators |
|---|---|---|
| Move Short Ordinal | Move Ordinal | Move Short Real |
| Zero Short Ordinal | Zero Ordinal | Zero Short Real |
| One Short Ordinal | One Ordinal | Save Short Real |
| Save Short Ordinal | Save Ordinal | Add Short Real—Short Real |
| AND Short Ordinal | AND Ordinal | Add Short Real—Temporary Real |
| OR Short Ordinal | OR Ordinal | Add Temporary Real—Short Real |
| XOR Short Ordinal | XOR Ordinal | Subtract Short Real—Short Real |
| XNOR Short Ordinal | XNOR Ordinal | Subtract Short Real—Temporary Real |
| Complement Short Ordinal | Complement Ordinal | Subtract Temporary Real—Short Real |
| Extract Short Ordinal | Extract Ordinal | Multiply Short Real—Short Real |
| Insert Short Ordinal | Insert Ordinal | Multiply Short Real—Temporary Real |
| Significant Bit Short Ordinal | Significant Bit Ordinal | Multiply Temporary Real—Short Real |
| Add Short Ordinal | Add Ordinal | Divide Short Real—Short Real |
| Subtract Short Ordinal | Subtract Ordinal | Divide Short Real—Temporary Real |
| Increment Short Ordinal | Increment Ordinal | Divide Temporary Real—Short Real |
| Decrement Short Ordinal | Decrement Ordinal | Negate Short Real |
| Multiply Short Ordinal | Multiply Ordinal | Absolute Value Short Real |
| Divide Short Ordinal | Divide Ordinal | |
| Remainder Short Ordinal | Remainder Ordinal | |
| Equal Short Ordinal | Equal Ordinal | |
| Not Equal Short Ordinal | Not Equal Ordinal | |
| Equal Zero Short Ordinal | Equal Zero Ordinal | |
| Not Equal Zero Short Ordinal | Not Equal Zero Ordinal | |
| Greater Than Short Ordinal | Greater Than Ordinal | |
| Greater Than or Equal Short Ordinal | Greater Than or Equal Ordinal | |
| Convert Short Ordinal to Character | Convert Ordinal to Short Ordinal | |
| Convert Short Ordinal to Ordinal | Convert Ordinal to Integer | |
| Convert Short Ordinal to Temporary Real | Convert Ordinal to Temporary Real | |

## Table 14. General Data Processor Operator Set Summary (Cont'd.)

| Short-Real Operators | Real Operators | Temporary-Real Operators |
|---|---|---|
| Equal Short Real<br>Equal Zero Short Real<br>Greater Than Short Real<br>Greater Than or Equal Short Real<br>Positive Short Real<br>Negative Short Real<br><br>Convert Short Real to Temporary Real | Move Real<br>Zero Real<br>Save Real<br><br>Add Real—Real<br>Add Real—Temporary Real<br>Add Temporary Real—Real<br>Subtract Real—Real<br>Subtract Real—Temporary Real<br>Subtract Temporary Real—Real<br>Multiply Real—Real<br>Multiply Real—Temporary Real<br>Multiply Temporary Real—Real<br>Divide Real—Real<br>Divide Real—Temporary Real<br>Divide Temporary Real—Real<br>Negate Real<br>Absolute Value Real<br><br>Equal Real<br>Equal Zero Real<br>Greater Than Real<br>Greater Than or Equal Real<br>Positive Real<br>Negative Real<br><br>Convert Real to Temporary Real | Move Temporary Real<br>Zero Temporary Real<br>Save Temporary Real<br><br>Add Temporary Real<br>Subtract Temporary Real<br>Multiply Temporary Real<br>Divide Temporary Real<br>Remainder Temporary Real<br>Negate Temporary Real<br>Square Root Temporary Real<br>Absolute Value Temporary Real<br><br>Equal Temporary Real<br>Equal Zero Temporary Real<br>Greater Than Temporary Real<br>Greater Than or Equal Temporary Real<br>Positive Temporary Real<br>Negative Temporary Real<br><br>Convert Temporary Real to Ordinal<br>Convert Temporary Real to Integer<br>Convert Temporary Real to Short Real<br>Convert Temporary Real to Real |
| **Access Descriptor Movement Operators** | **Rights Manipulation Operators** | **Type Definition Manipulation Operators** |
| Copy Access Descriptor<br>Null Access Descriptor | Amplify Rights<br>Restrict Rights | Create Public Type<br>Create Private Type<br>Retrieve Public Type Representation<br>Retrieve Type Representation<br>Retrieve Type Definition |
| **Refinement Operators** | **Segment Creation Operators** | **Access Path Inspection Operators** |
| Create Generic Refinement<br>Create Typed Refinement<br>Retrieve Refined Object | Create Data Segment<br>Create Access Segment<br>Create Typed Segment<br>Create Access Descriptor | Inspect Access Descriptor<br>Inspect Access |
| **Object Interlock Operators** | **Branch Operators** | **Interconnect Operators** |
| Lock Object<br>Unlock Object<br>Indivisibly Add Short Ordinal<br>Indivisibly Add Ordinal<br>Indivisibly Insert Short Ordinal<br>Indivisibly Insert Ordinal | Branch<br>Branch True<br>Branch False<br>Branch Indirect<br>Branch Intersegment<br>Branch Intersegment without Trace<br>Branch Intersegment and Link | Move to Interconnect<br>Move from Interconnect |
| **Process Communication Operators** | **Processor Communication Operators** | **Context Communication Operators** |
| Send<br>Receive<br>Conditional Send<br>Conditional Receive<br>Surrogate Send<br>Surrogate Receive<br>Delay<br>Read Process Clock | Send to Processor<br>Broadcast to Processors<br>Read Processor Status and Clock | Enter Access Segment<br>Enter Global Access Segment<br>Set Context Mode<br>Call Context<br>Call Context with Message<br>Return from Context |

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara 95051 (408) 734-8102 x598
Printed in U.S.A./Y-32/0281/50K/PS/GFH