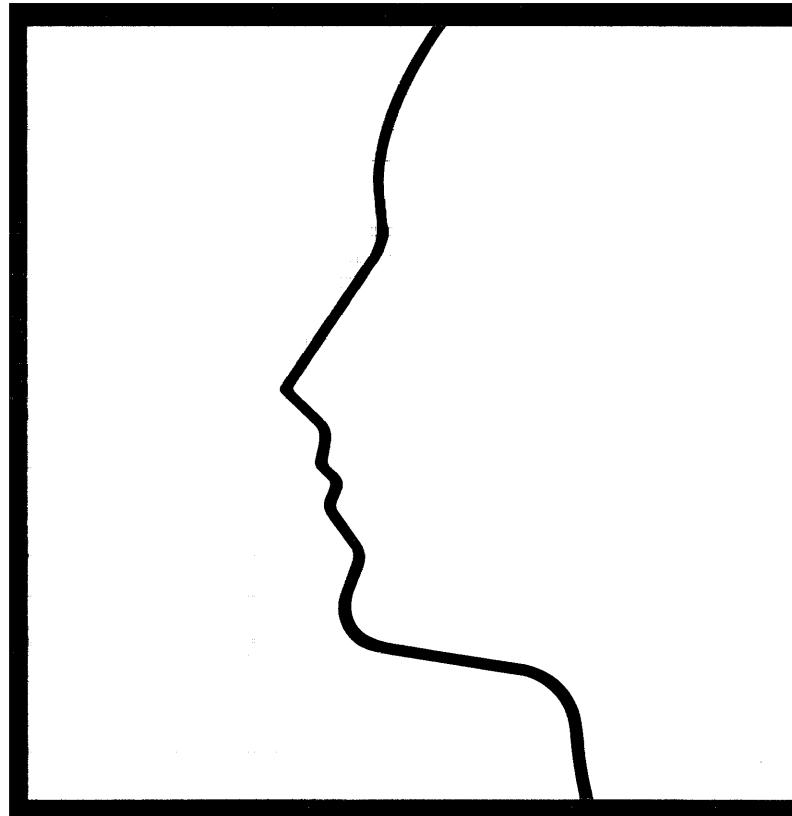# Texas Instruments

EXPLORER™

PROCESSOR

GENERAL DESCRIPTION

EXPLORER™ PROCESSOR GENERAL DESCRIPTION

MANUAL REVISION HISTORY

Explorer™ Processor General Description (2243144-0001) Rev. A

Original Issue...September 1984

Revision A...October 1985

THE EXPLORER™ SYSTEM HARDWARE MANUALS

**System Level
Hardware
Publications**

Explorer Unpacking and Inventory 2243216-0001
Explorer 7-Slot System Installation 2243140-0001
Explorer System Field Maintenance 2243141-0001

**System Enclosure
Hardware
Publications**

Explorer 7-Slot System Enclosure General Description 2243143-0001
Explorer 2-Megabyte Memory General Description 2243147-0001
Explorer Memory General Description 2533592-0001
Explorer Processor General Description 2243144-0001
Explorer Display Unit General Description 2243151-0001
Explorer System Interface General Description 2243145-0001

**Mass Storage
Hardware
Publications**

Explorer Mass Storage Enclosure General Description 2243148-0001
Explorer Winchester Disk Formatter (ADAPTEC) Supplement
    to Explorer Mass Storage Enclosure General Description 2243149-0001
Explorer Winchester Disk Drive (Maxtor) Supplement to
    Explorer Mass Storage Enclosure General Description 2243150-0001
Explorer Cartridge Tape Drive (Cipher) Supplement to Explorer
    Mass Storage Enclosure General Description 2243166-0001
Explorer Cable Interconnect Board (2236120-0001) Supplement to
    Explorer Mass Storage Enclosure General Description 2243177-0001
Explorer NuBus™ Peripheral Interface
    General Description (NUPI board) 2243146-0001

**Mass Storage
Hardware Vendor
Publications**

Series 540 Cartridge Tape Drive Product Description, Cipher Data
    Products, Inc., Bulletin Number 01-311-0284-1K (¼-inch tape drive) . . 2249997-0001
MT01 Tape Controller Technical Manual, Emulex Corporation,
    part number MT0151001 (formatter for the ¼-inch tape drive) 2243182-0001
XT-1000 Service Manual, 5¼-inch Fixed Disk Drive,
    Maxtor Corporation, part number 20005
    (5¼-inch Winchester disk drive, 112 megabytes) 2249999-0001
ACB-5500 Winchester Disk Controller User's Manual,
    Adaptec, Inc., (formatter for the 5¼-inch Winchester disk drive) 2249933-0001

**Optional Equipment
Hardware
Publications**

Explorer NuBus Ethernet® Controller General Description 2243161-0001
Model 855 Printer Operator's Manual 2225911-0001
Model 855 Printer Technical Reference Manual 2232822-0001
Model 855/856 Printer Maintenance Manual 2225914-0001

Explorer and NuBus are trademarks of Texas Instruments Incorporated.
Ethernet is a registered trademark of Xerox Corporation.

THE EXPLORER™ SYSTEM SOFTWARE MANUALS

Explorer is a trademark of Texas Instruments Incorporated.

ABOUT THIS MANUAL

Purpose

This manual describes the Texas Instruments Explorer processor. The information in this manual serves as a stand-alone descriptive document for the Explorer processor. The manual is written for original equipment manufacturers (OEMs), system designers, field maintenance people, and Texas Instruments customer representatives (CRs).

Contents of This Manual

This manual is divided into four sections as follows:

Section 1: Introduction — Provides a general description of the Explorer processor, lists its important features, and describes how the processor functions as a part of the Explorer computer system.

Section 2: Installation — Provides unpacking and installation procedures for the Explorer processor.

Section 3: Operation — Provides operating information that includes how the operator should respond to external and internal resets, and how to interpret fault indicator responses to self-tests.

Section 4: System Design and Programming Data — Provides an overall view of the operation of the Explorer processor that is keyed to block diagrams. A basic description of processor data flow, processor control and timing, and macroinstruction and microinstruction formats is also provided.

CONTENTS

List of Acronyms

Glossary

Index

Figure	Title	Page

	Table	Title	Page

INTRODUCTION

Highlights of This Section

- Bus interface

- Memory

- Board address configuration

- Features

- Characteristics

General

1.1   This section provides general information on the Explorer processor board (Figure 1-1), which serves as the processor in the Texas Instruments Explorer Computer System shown in Figure 1-2. Figure 1-3 shows the system enclosure in more detail.

NOTE: Because of heat generation within the system enclosure, the processor must always be installed in slot 6 of the enclosure.

The basic Explorer computer system block diagram, Figure 1-4, shows major circuit boards within the system enclosure and their local bus and NuBus interfaces with each other. Explorer system features that are important to understanding the operation of the Explorer processor are described in the following paragraphs.

Explorer Computer System Overview

1.2   The Explorer system features fast processing, a large memory capacity, and high-quality graphics. It also provides support for development of large-scale, complex programs and new technology research.

Figure 1-1   Explorer Processor
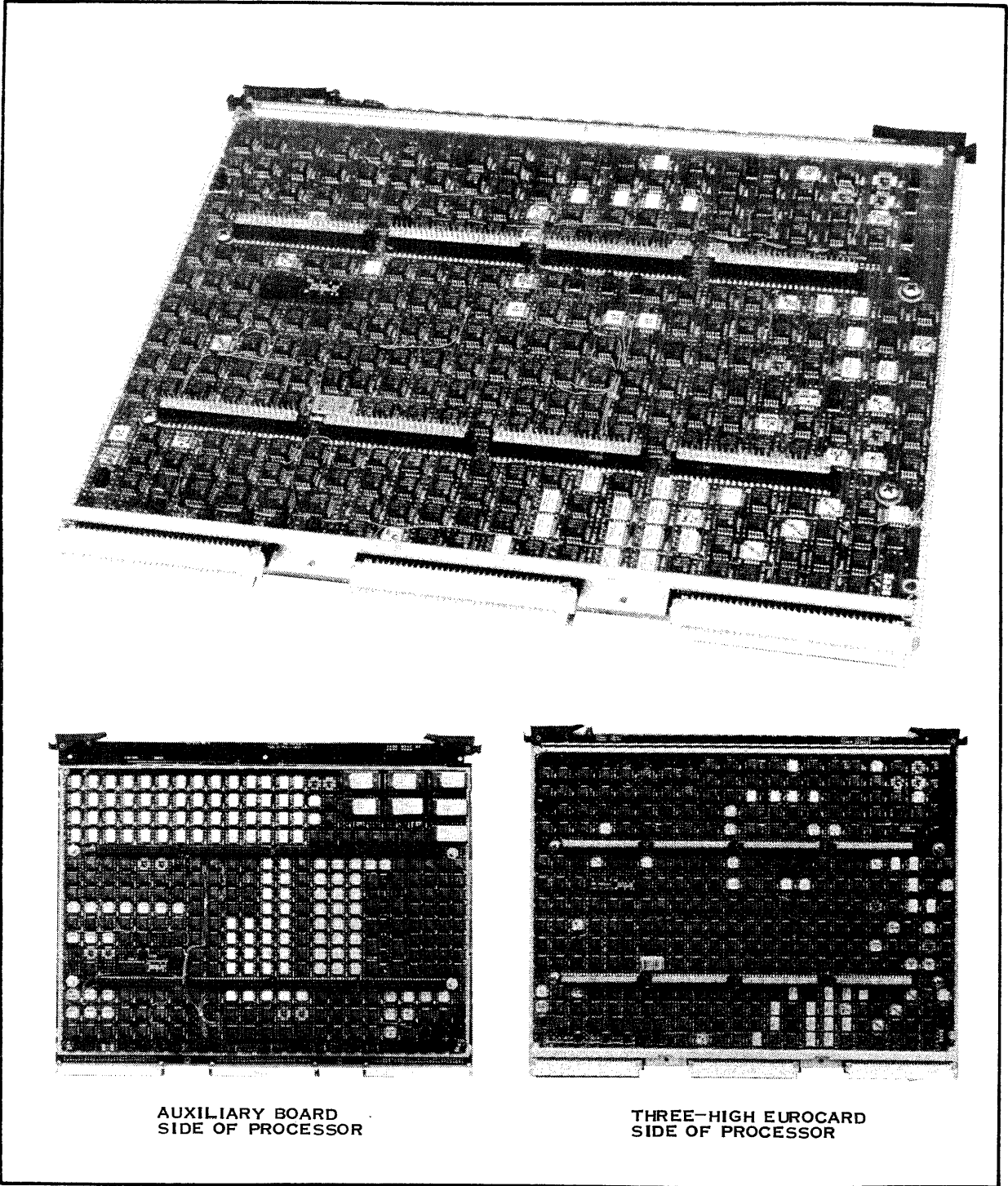
AUXILIARY BOARD
SIDE OF PROCESSOR

THREE—HIGH EUROCARD
SIDE OF PROCESSOR

Figure 1-2   Example of the Explorer System Components

display monitor

mouse on pad

mass storage enclosure

keyboard

7-slot system enclosure
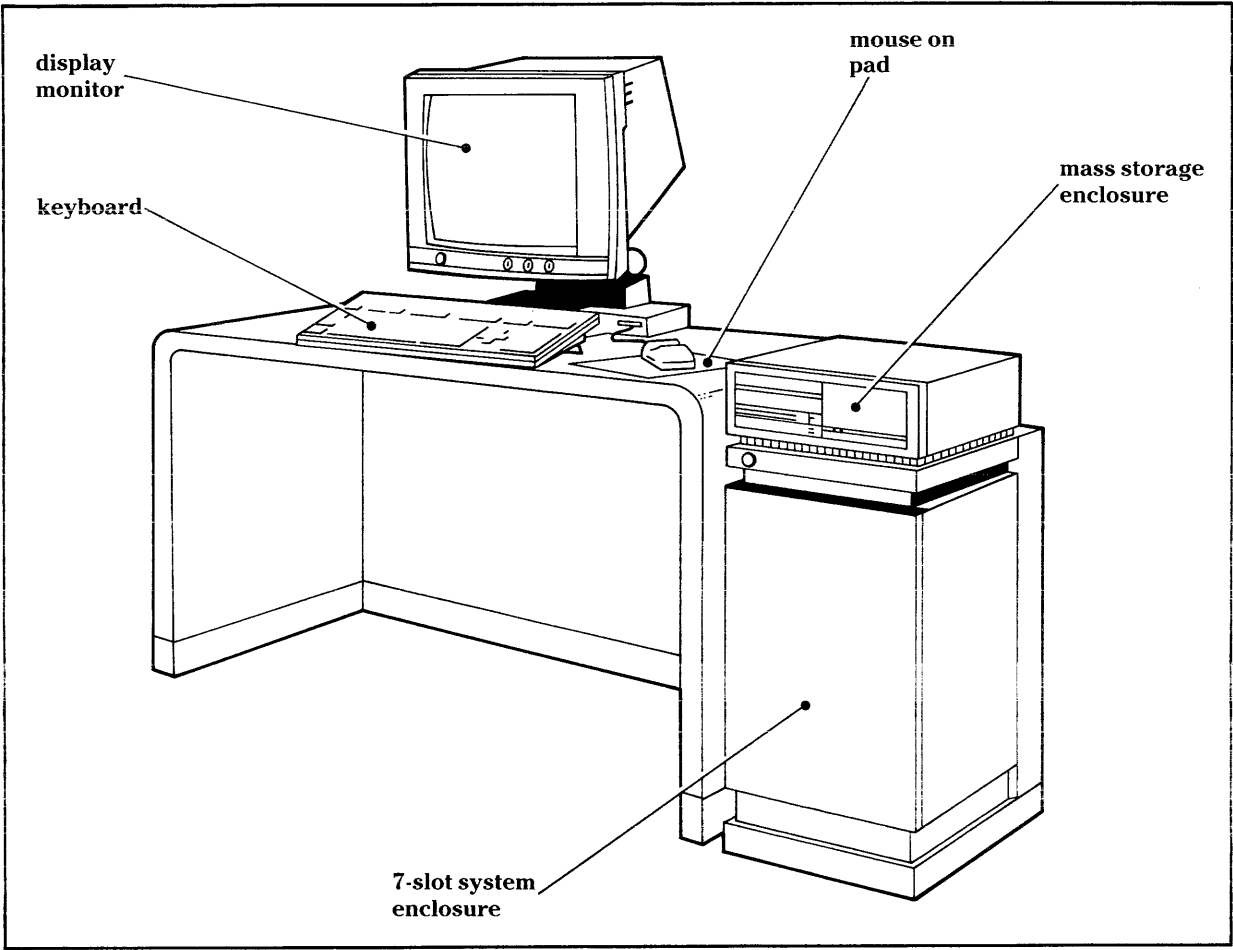
Figure 1-3 Explorer System Enclosure

inner metal door

power on/off button

metal dividers

inner door latches

6 slots for circuit boards

processor board

power supply
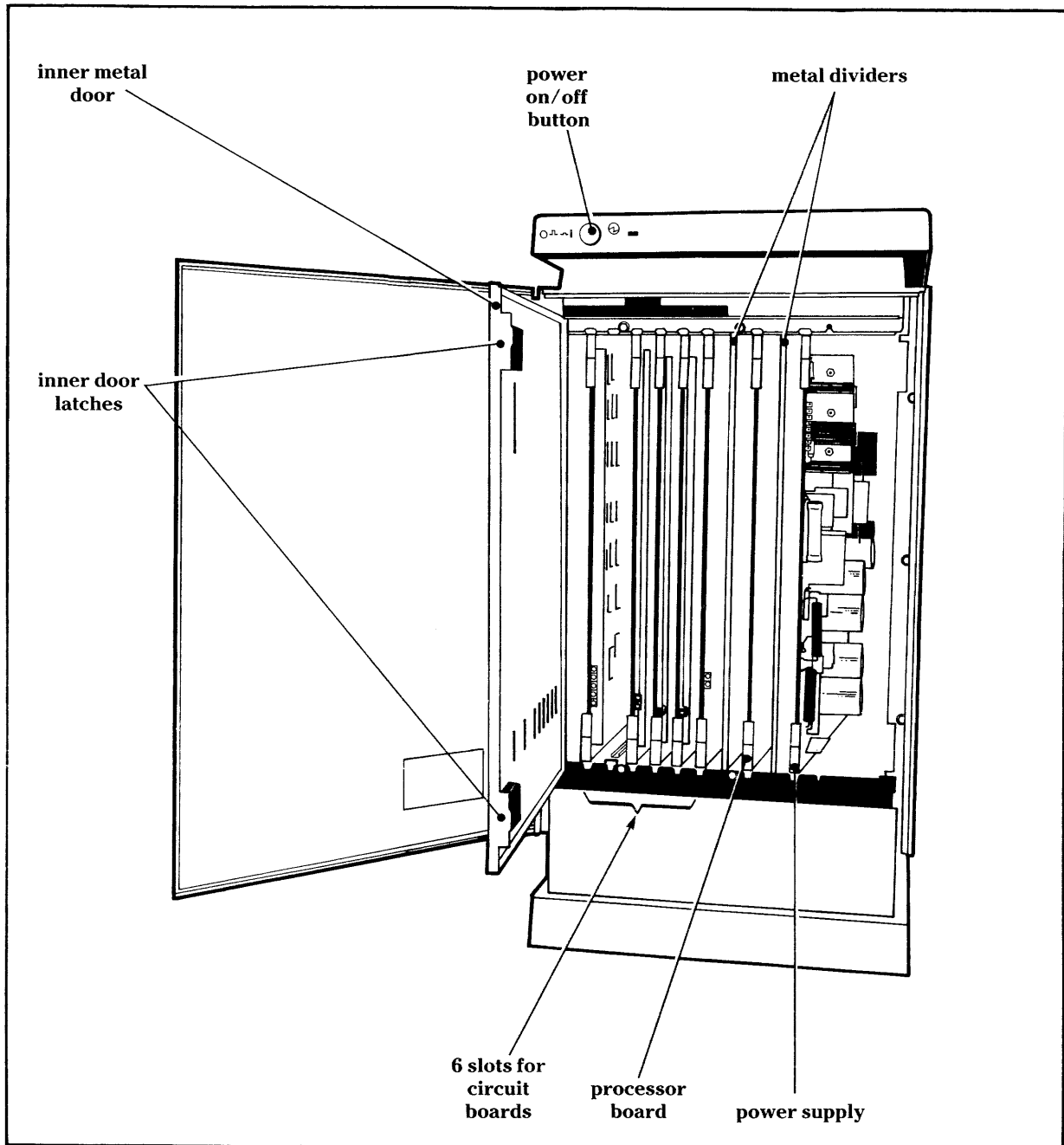
Figure 1-4 Explorer Computer System Block Diagram
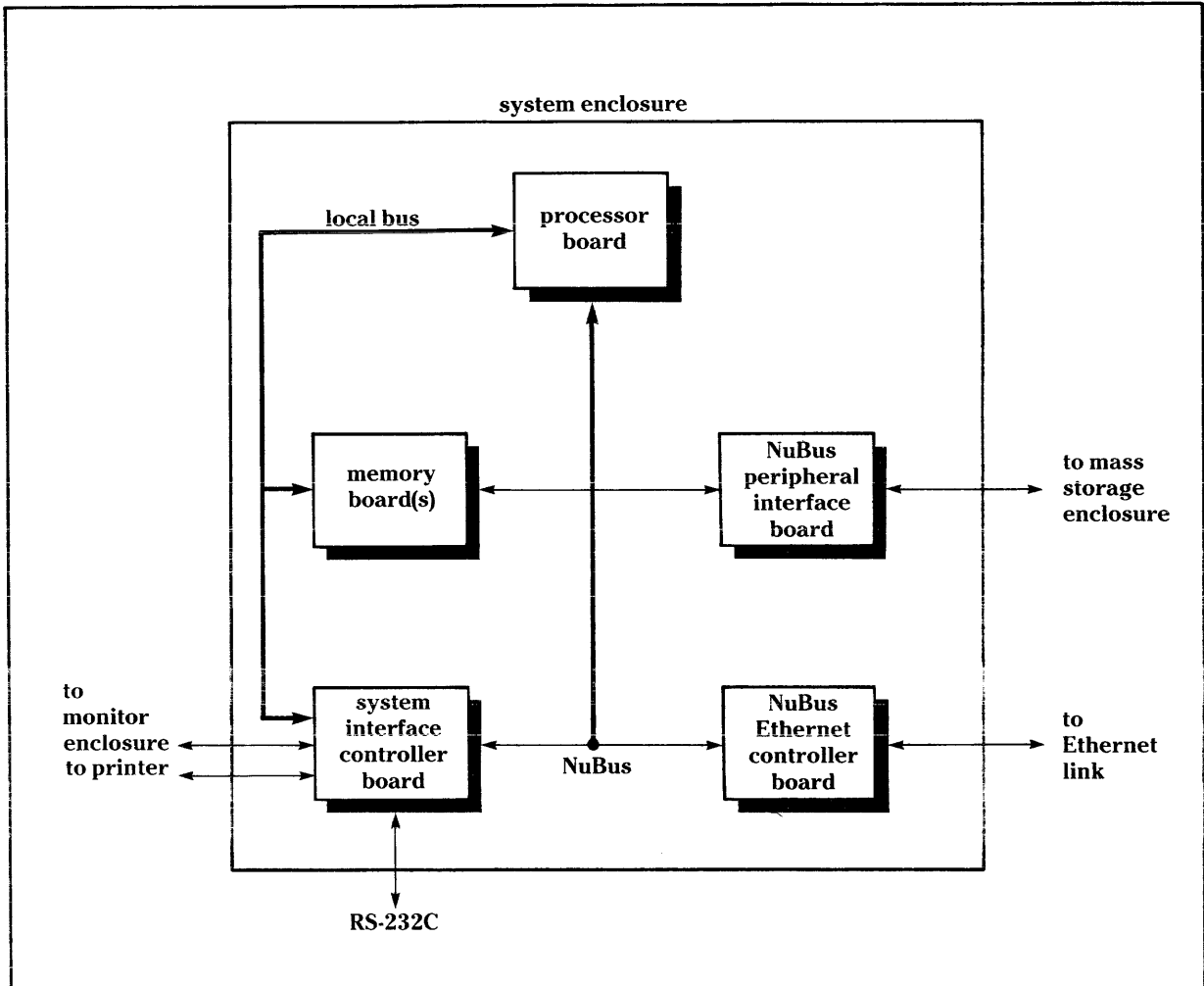
Bus Interfaces **1.2.1** The NuBus is a 32-bit, synchronous bus with addresses and data multiplexed on the same lines. A simple communications protocol allows circuit boards on the NuBus to be either master or slave devices. All control, data, address, and power lines are brought to one 96-pin DIN connector. On the system enclosure backplane, connector P1 provides connection to the NuBus for all system boards. Refer to the *Explorer 7-Slot System Enclosure General Description* manual for additional NuBus information.

The local bus is also a high-speed, synchronous bus, but it provides separate address and data lines. All control, data, address, and power lines for the local bus are also brought to one 96-pin DIN connector. The Explorer processor is the master device on the local bus; all other circuit boards on the local bus are slave devices.

Local Memory **1.2.2** The local memory for the Explorer system consists of either one or two memory boards. The processor accesses this memory via the local bus.

The memory board has a capacity of 2 megabytes when using 64K-bit dynamic random-access memory (DRAM) chips, and a capacity of 8 megabytes when using 256K-bit DRAM chips. The system enclosure provides 2 slots (slots 3 and 4) into which memory boards can be installed, thus increasing the total possible memory capacity to 16 megabytes.

Explorer system protocol mandates that, in either case (1 memory board or 2 memory boards), a memory board must occupy slot 4. This slot, when enabled by a local bus signal from the processor, becomes a NuBus master to act as the local-bus-to-NuBus transfer port for the processor. The memory board uses both the NuBus and the local bus to communicate with other system components.

Board Address Configuration **1.2.3** System enclosure slots 3, 4, 5, and 6 are reserved for the memory boards, the system interface board, and the processor board, respectively. In general, all other circuit boards can be inserted into any of the remaining slots without requiring jumpers or switches to link them to specific locations. Hard-wired identification (ID) codes (> 0 through > F) on the backplane identify the slots. Once a board is inserted into a particular slot, the ID code for that slot becomes a part of the board address. The board address is in the form of > FSXXXXXX where S is the slot ID code and XXXXXX represents numbers in the range of > 000000 through > FFFFFF.

NOTE: A value preceded by a right angle bracket (>) indicates a hexadecimal value.

Each circuit board in the Explorer system contains a configuration read-only memory (ROM) that provides a unique identity for the board. During power-up, the processor reads the contents of each configuration ROM to determine the addresses of all circuit boards. All succeeding operations use these addresses to communicate with the boards.

Each backplane slot is allocated 16 megabytes of address space within the range of > FS000000 through > FSFFFFFF. The total address space reserved and mapped for the 16 possible slots on the NuBus is 256 megabytes. This is only one sixteenth of the total 4-gigabyte address space capability of the 32-bit NuBus and local bus. The unused address space, from > 00000000 to > F0000000, is uncommitted and used as required.

Explorer Processor Features

1.3 The Explorer processor is a microprogrammed processor specially designed for high-speed symbolic processing. The microcode provides optimum performance for the list processing (Lisp) computer language. The hardware features that support Lisp programming include the following:

- Bit-field hardware to manipulate complex data structures

- Tagged architecture for typed data

- Multiway branching to facilitate the complex control operations associated with symbolic processing

- Lisp instruction set decoding

- Large virtual address space with demand-paged memory mapping

- Full 32-bit data paths and a 16K by 56-bit writable control store for microinstructions

- Special memory reclaimer (garbage collection) hardware to facilitate dynamic reallocation of memory

- Large stack cache

- Self-test routines and a microcoded diagnostic engine for testing other boards in the system

- A 142-nanosecond microinstruction cycle

Explorer Processor Specifications

1.4   Table 1-1 lists general specifications for the Explorer processor.

Table 1-1

Explorer Processor Specifications

Item	Characteristic
Word size	32 bits (4 bytes, 8 bits each)
Clock rate	Approximately 7 MHz
Address bus	32 bits
Data bus	32 bits
Power	160 watts (5 volts at 32 amperes)
Temperature:	
Operating	10 to 35 degrees C (50 to 95 degrees F)
Nonoperating	− 40 to 65 degrees C (− 40 to 149 degrees F)
Relative humidity:	
Operating	15 to 80 percent, noncondensing
Nonoperating	5 to 95 percent, noncondensing
Board size:	
Width	366.7 mm (14.437 in)
Depth	280 mm (11.024 in)
Electromagnetic emission	The Explorer processor is a component of an Explorer computer system and is tested to meet applicable FCC and VDE electromagnetic interference requirements at the system level.

INSTALLATION ■ *2*

**Highlights of
This Section:**

■ Removing a processor board

■ Unpacking a new processor

■ Installing a processor

Introduction

2.1   This section provides the instructions for removing the processor board from an Explorer system enclosure, unpacking a new processor, and installing a processor in the system enclosure.

CAUTION: The processor board contains static-sensitive electronic components. To prevent damage to these components, make sure that you are properly grounded before handling the processor board.

The recommended grounding method is to use a static-control system composed of a static-control floor or table mat and a static-control wrist strap (these are commercially available). If you do not have a static-control system, you can discharge any static charge by touching a properly grounded object prior to handling the processor board. As an additional safety measure, put the processor board on a grounded work surface after removing it from the system enclosure or its static-protective package.

Before storing or transporting the processor board, return it to its static-protective package or the system enclosure.

Removal

2.2   To remove the processor from the Explorer system enclosure, perform the following steps:

1. If the system power is on, press the system enclosure power switch and release it to its off (out) position. Figure 1-3 shows the power switch location.

2. Open the front external door of the system enclosure.

3. Open the inner metal door by sliding the two latches (Figure 1-3) to the left.

4. Pull out the two ejector/injector levers located at the top and bottom of the edge of the processor board to unseat the board from the backplane connectors.

5. Grasp the edge of the board with both hands (near the top and bottom) and pull straight out until the board is free of the enclosure.

6. Temporarily place the board on a grounded work surface. Later, after unpacking the new board, put the old board in the static-protective bag.

Unpacking Procedure

2.3 If any unpacking instructions are attached to the exterior of the packing container, read and follow those instructions. If not, perform the following steps:

1. Open the packing container (Figure 2-1) and carefully remove the packing material; then remove the processor board with its static-protective bag in place.

2. Remove the static-protective bag (Figure 2-2) from the processor board. Be sure to follow the static-caution recommendations when handling the processor board without the static-protective bag.

3. Check the processor board for scratches, broken parts, marred finish, or other damage that may have occurred during shipment. Follow your local procedures to report any damage.

Figure 2-1   Processor Board in Packing Container
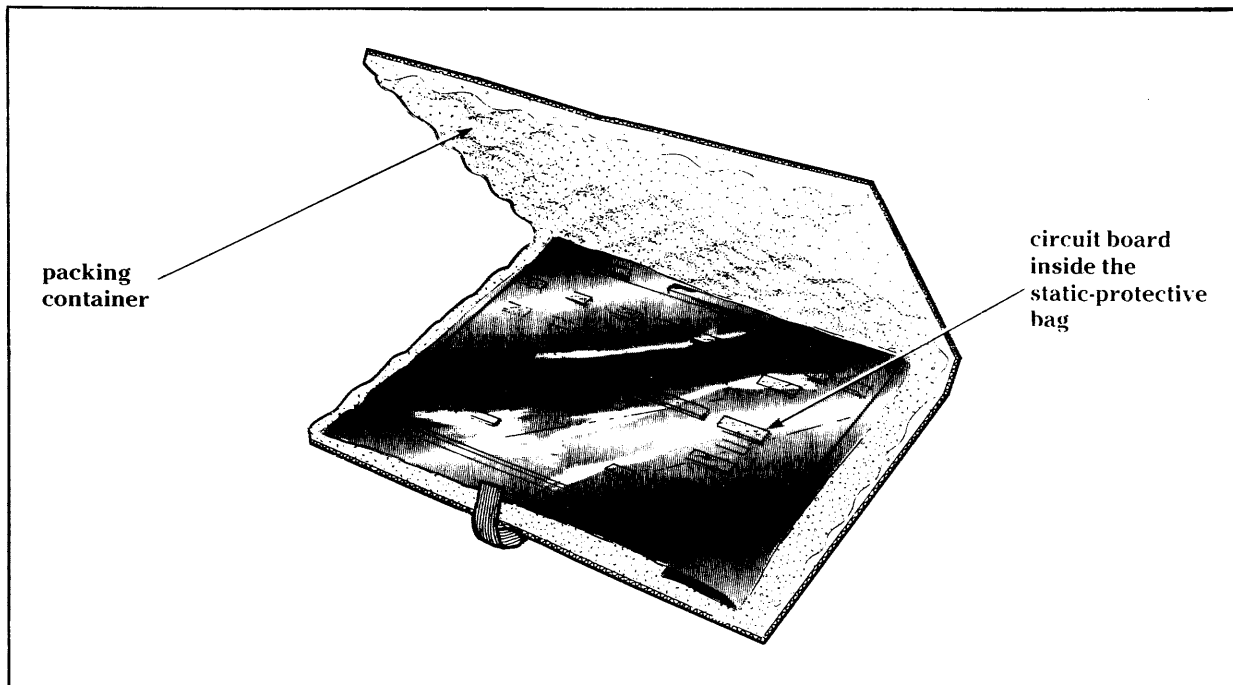
Figure 2-2   Removal of Static-Protective Bag

circuit board

static-protective bag
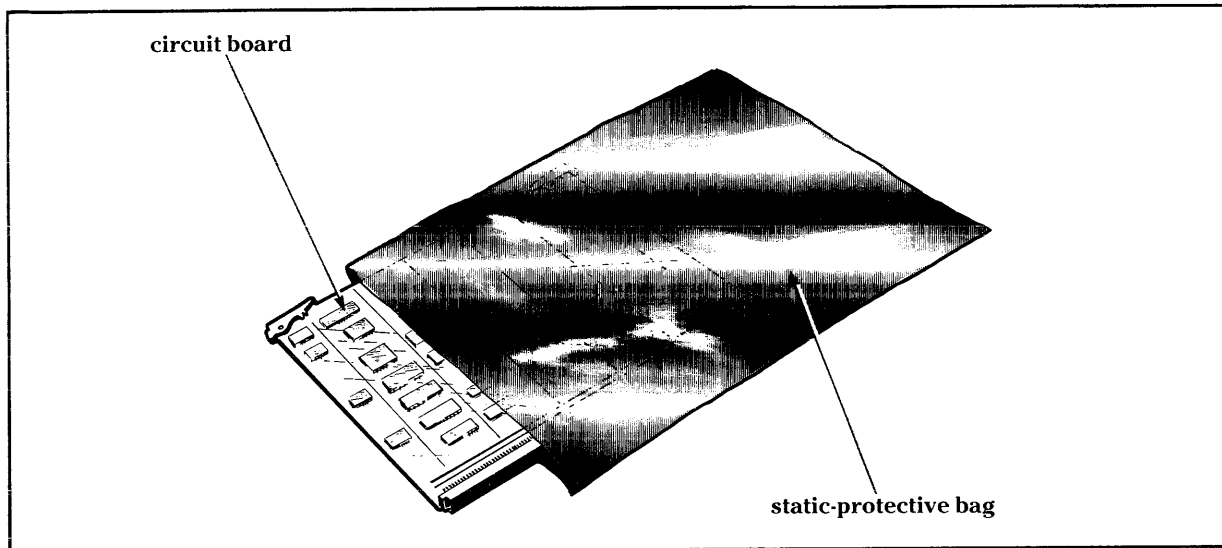
CAUTION: The chassis slot into which the processor is to be installed requires special high-airflow ducting for cooling air. Proper airflow occurs only with the doors and panels in place during operation. If the system enclosure cooling system is disturbed, the resultant heat buildup can damage the processor board.

Installation

2.4 To install a processor board in the Explorer system enclosure, perform the following steps:

NOTE: Refer to Figure 1-3.

1. Set the system enclosure power switch to its off (out) position.

2. Open the front external door of the system enclosure.

3. Open the inner metal door by sliding the two latches to the left.

4. Position the processor with connector P1 at the top, then carefully slide the processor into the slot track.

5. Push the processor in until its connectors make contact with the backplane connectors.

6. Press the two ejector/injector levers located at the top and bottom of the processor board to seat and lock the processor into the backplane connectors.

7. Close and latch both front doors.

8. Perform the self-test procedures described in Section 3 of this manual.

OPERATION 3

Highlights of This Section
- General information
- Self-test procedure

System Start-Up Procedures

3.1 The *Explorer 7-Slot System General Description* provides complete system start-up instructions. The start-up procedure in this section is intended primarily to allow installation of a new processor board.

WARNING: Potentially lethal voltages are present in the system enclosure. Interlocks on the enclosure rear door and the front inner door disable power to the enclosure when the doors are open. Do not bypass or otherwise tamper with the interlocks.

Processor Board Self-Tests

3.2 Computer system self-tests occur automatically at power-up. However, self-tests can be initiated after power-up either by cycling system power off and then on, or by entering a command from the keyboard. The following procedure describes both methods.

1. Open the front door of the system enclosure so the fault indicators can be observed during self-test. Do not open the inner metal door.

2. To begin the self-test with power initially off, set the enclosure power switch to its on (in) position. To initiate a self-test after initial power-up, perform one of the following steps:

   ■ Set the enclosure power switch off and then on.

   ■ Simultaneously press both META keys, both CTRL keys, and the ABORT key on the Explorer keyboard.

3. Check to see that the red fault LED on each circuit board in the enclosure is on. This indicates that the self-test for that board is in progress. Refer to Figure 3-1 for fault LED locations.

4. If the processor board is good, its red fault LED will go out during the self-test period. The processor then runs the self-test microcode on other boards in the enclosure.

NOTE: All red fault LEDs should go out within two minutes after self-test initiation.

---

5. If the processor fault LED does not go out, try another processor board.

NOTE: Failure of the slot-4 memory board or any board in the system that locks the NuBus can also cause the processor board to fail.

---

6. When the self-tests are completed, close the outer front door for normal operation.

Figure 3-1 **Fault LED Locations**

outer door opened

inner metal door

viewing slots

board-dependent LEDs

fault LED
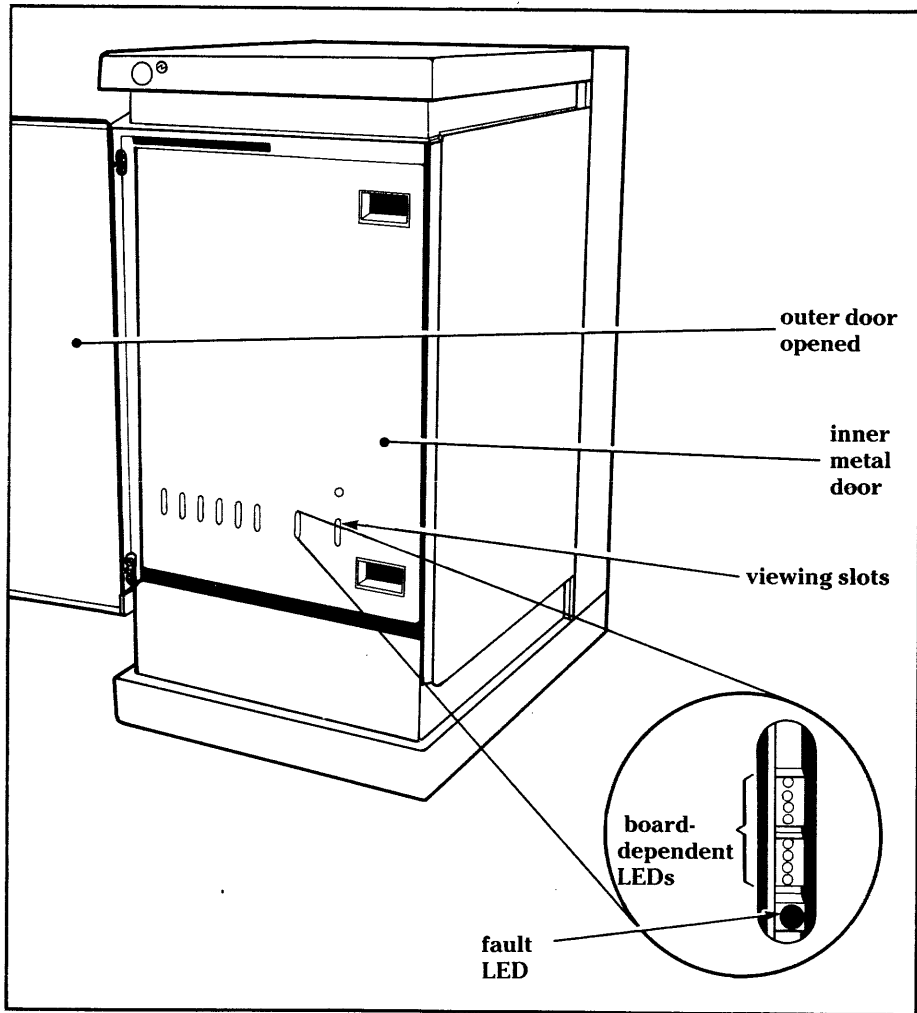
SYSTEM DESIGN AND
PROGRAMMING DATA

**Highlights of
This Section:**

- Operation overview

- Functional description

- Memory map and interface

- NuBus interface

- Macroinstructions

- Microinstructions

- Formats

General

4.1 All numbers used to describe bit positions and field widths in this section are decimal. Memory sizes are decimal unless a suffix of K (where K equals 1024) or M (where M equals 1 048 576) is attached. Memory capacity is assumed to be the number of 8-bit bytes unless otherwise specified. Bits are always described within a field so that the least significant bit (LSB) has the lowest value descriptor. Bits are numbered sequentially from least to most significant, and bit position zero is always the LSB unless otherwise specified.

The Explorer processor can operate on words, halfwords, and bytes. The bit organization for words, halfwords, and bytes is shown in Figure 4-1. A minus sign (−) following a signal is used to denote active low. This manual denotes concatenation (linking numbers together) by the use of two colons, for example (31::30).

The processor includes features that aid in the execution of a Lisp-oriented virtual machine. Lisp data is stored in main memory in 32-bit quantities called storage quantums (Qs) (Figure 4-2). A Q is divided into three fields:

■ Q(31:30) — CDR code field. The information in this field reduces the amount of storage required for list data.

■ Q(29:25) — Q's data type or tag field.

■ Q(24:00) — Q-pointer field. This field contains either a pointer (the address of another Q) or immediate data, depending on the Q's data type.

Figure 4-1   **Organization of Words, Halfwords, and Bytes**

bit 31   bit 0

32-bit word			
halfword 1		halfword 0	
byte 3	byte 2	byte 1	byte 0
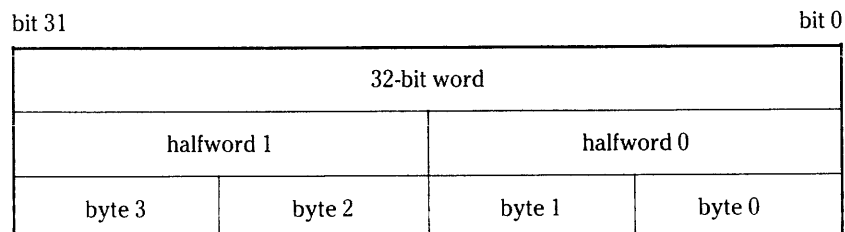
Figure 4-2   Storage Quantum Format

31 30	29 25	24 0

- Pointer field
- Data type field
- CDR code
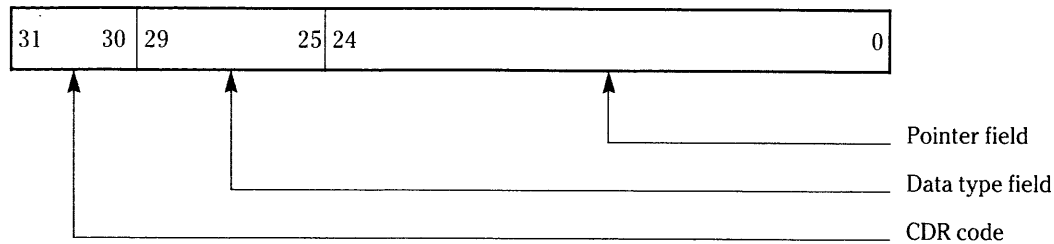
Processor Operational Overview

4.2   The processor is a general purpose, 32-bit microprogrammed processor designed to support high-speed, symbolic processing. The microcode is designed for optimum Lisp performance. A typical configuration of an Explorer computer system includes the following:

- A 2-megabyte memory array (8-megabyte array when 256K-bit dynamic random-access memory chips are used)

- A console interface with a bit-mapped graphics display and a keyboard interface

- A mass storage system

- A local area network (LAN) interface/controller

The processor accesses local memory via the local bus. This allows high-bandwidth memory transfers without the need to gain access to the NuBus. The processor can access other memory, post events to other processors, and perform I/O operations through the NuBus. Local memory and the processor are both accessible from the NuBus. The processor provides locations where other processors can post events. It also provides control registers and a configuration ROM that conforms to system conventions.

Processor Functional Description

4.3   The following paragraphs provide an overall functional view of the processor.

Processor Functional Features

4.3.1   Nine arithmetic logic units (ALUs) and one look-ahead carry generator perform 16 high-speed, binary arithmetic operations on two 32-bit words. The ALU can be used as a comparator, performing 16 logic functions. Important functional features of the processor and their interactions are shown in Figure 4-3.

Figure 4-3   Explorer Processor Block Diagram

R bus (7)

μPCS
64 by 20

PC
source
select

IR

IR
decode

to
processor
control
logic

O bus(32)

dispatch
memory
4K by 17

trap
logic

+1

control store
2K by 56 PROM
16K by 56 RAM

A memory
1K by 32

ALU control
(from
IR decode)

ALU

A bus
bus

O bus
multi-
plexer

masker

M bus(32)

R bus

memory
data
register

virtual
memory
address
register

M memory
64 by 32

data
local
bus

address

memory
map

macro
location
counter

PDL
memory
1K by 32

barrel
shifter

local
bus
clock

clock

on-board
clock

machine
control
register

PDL
pointer

PDL
index

MF-to-M
bus
buffer

NuBus

NuBus
interface

MF bus(32)

Q
register

The following features are provided by the processor:

- Control store programmable, read-only memory (PROM) — 2K by 56-bit PROM used for load and self-test programs. Refer to Section 3 for self-test information.

- Writable control store, static, random-access memory (RAM) — 16K by 56-bit static RAM used to store microinstructions. The control store PROM and writable control store RAM are connected in parallel.

- A memory — 1K by 32-bit scratchpad memory used for microcode variable storage and as a storage for constants. The A memory is the source for the 32-bit A bus input to the ALU.

- M memory — 64 by 32-bit scratchpad memory that is one source for the 32-bit M bus. A copy of data written to the M memory is kept in the first 64 locations of the A memory.

- Push-down list (PDL) memory — 1K by 32-bit PDL memory used as a special-purpose cache to contain the top portion of the processor push-down stack. The PDL memory output is the M bus. The PDL memory can be indexed by either the PDL pointer or the PDL index. The contents of the PDL pointer or the PDL index can be read via the MF bus.

- MF bus — The MF bus has the following sources:

  - Dispatch constant register

  - PDL index register

  - PDL pointer

  - Microprogram counter stack (μPCS) pointer

  - Location counter

  - Q shift register

  - Virtual memory mapping RAM

- Barrel shifter — Provides a shift function to its M bus source by rotating the input from 0 to 31 positions. The barrel shifter output is to the R bus.

- R bus — The R bus is 32 bits wide and feeds the output bus (O bus) multiplexer, feeds a single bit to the condition logic, and feeds seven bits to the bit selector logic for the dispatch memory.

- O bus multiplexer and masker — Selects from one of seven sources of data (not all shown on simplified block diagram). The O bus multiplexer output is the O bus. The masker that is part of the O bus multiplexer selects between the R bus and the A bus. Bits under the mask are from the R bus. Those not under the mask are from the A bus.

- Microprogram counter stack (μPCS) — 64 by 20-bit memory that functions primarily as a microcode subroutine call/return stack. The μPCS can also be used to temporarily store data.

- Program counter (PC) source select — Selects a 14-bit field from various input sources. The 14-bit field is called the micro PC and is used to address the control store.

- Dispatch memory — Provides 4K by 17-bit storage of 14-bit micro PCs, each with its associated 3-bit transfer-type field.

- Instruction register (IR) and IR decode logic — 56-bit register that holds the 56-bit microinstruction from which all control signals originate. The IR decode logic decodes microinstructions to provide the appropriate processor control signals.

- Memory data (MD) register — 32-bit I/O data interface to the local bus.

- Q register — 32-bit shift register connected to the output of the ALU that supports multiply and divide algorithms in the microcode.

- Machine control register (MCR) — Controls and tests various machine signals such as the highest interrupt pending. Other hardware functions are controlled by storing data in the MCR.

- Macro location counter — Holds the macroinstruction PC.

- Local bus — The processor uses the local bus as the address, data, and control interface from the processor to external devices.

- Clock — The processor generates a crystal-controlled clock used for synchronous control of the processor and as the clock for the local bus.

- NuBus interface — A NuBus slave-only interface that provides board configuration data to the bus and supports reset operations and interrupts.

Microinstruction Formats

4.3.2 Explorer processor instructions are sets of microinstructions accessed from either the control store PROM or the writable control store RAM. There are four different microinstruction formats: ALU, byte, jump, and dispatch. Some fields in the 56-bit microinstruction are common to two or more of the four formats, but other fields have functions applicable to only one of the four formats. The microinstruction format is described in detail later in this section. The four microinstruction formats and their functions are as follows:

■ ALU — Performs an arithmetic or logical function on data from the memories or registers and stores the result in a memory or register

■ Byte — Performs bit-field manipulation on the data and stores the result

■ Jump — Conditionally alters control flow by jumping, calling, or returning if the condition tested is true

■ Dispatch — Performs a multiway branch or call, based on a field of the source data

Processor Data Paths

4.3.3 The 32-bit data paths for the processor are shown in Figure 4-4. There are two main functional units, each of which can be used by a microinstruction to manipulate data. The ALU unit performs functions such as left shift, right shift, and sign extend. The barrel shifter and masker can be used to extract and deposit bit fields of any size or position from a 32-bit word.

The A bus and M bus provide inputs to the ALU. The A bus is fed from the A memory. The M bus is fed from either the M memory, the PDL memory, the memory data register, or the MF bus. The MD register receives and holds data from the local bus. The MF bus connects to various registers in the processor and is used for fast sources such as registers.

A major advantage of the barrel shifter is that the completely shifted output is available within one microcycle. The shifted data from the barrel shifter is applied to the R bus. The R bus feeds the masker that is part of the O bus multiplexer. The O bus multiplexer is used by the ALU to provide shifting and sign-extend functions. The masker selects between the R bus and the A bus. Bits under the mask come from the rotated M bus source, and bits not under the mask come from the A bus.

The O bus multiplexer applies the result from the ALU or the shifter/masker to the O bus. This result on the O bus can be stored in the M memory and also in the A memory, the PDL memory, or another register. The O bus is the heart of the processsor and is involved in almost every data motion.

Figure 4-4   Data Path Block Diagram
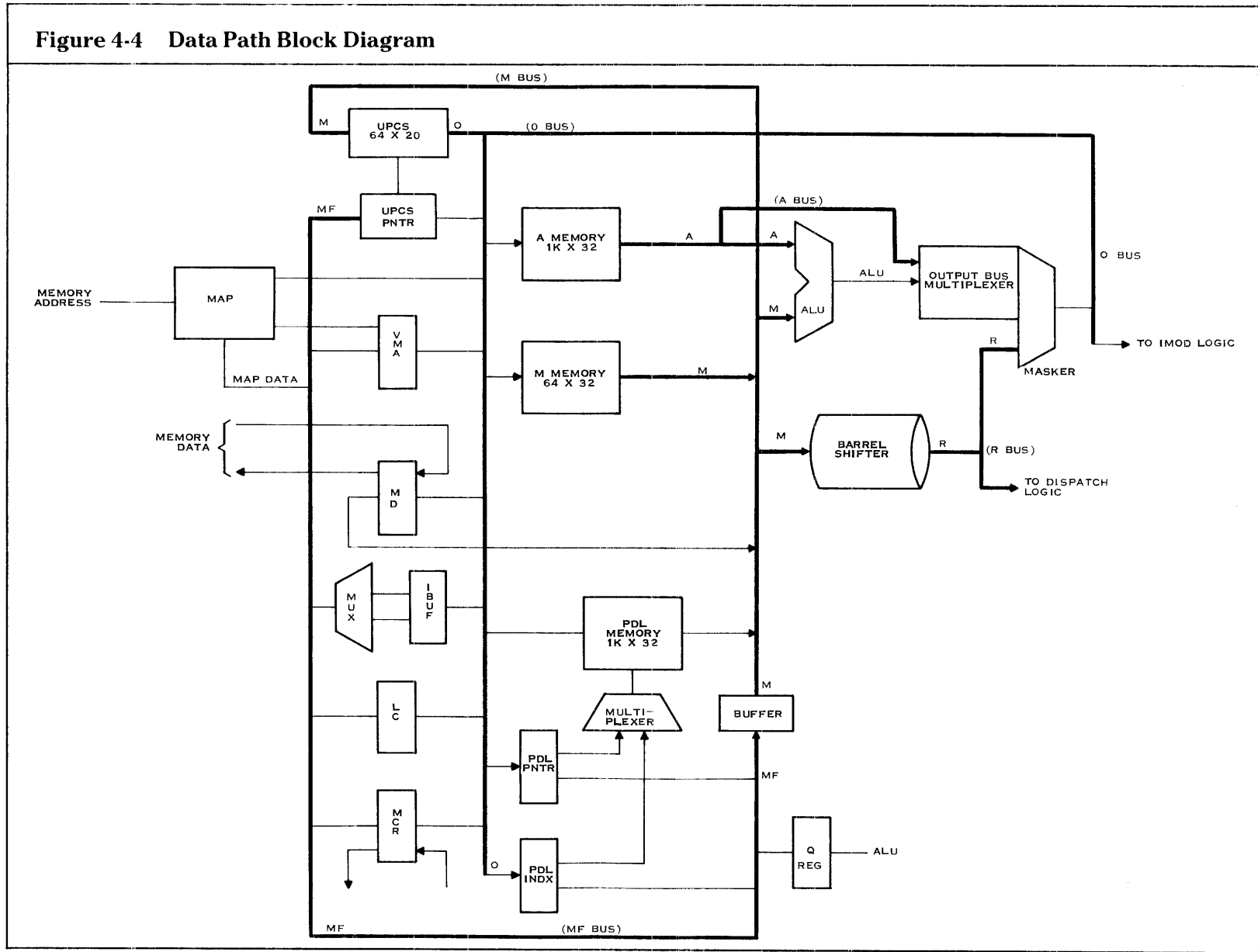
PDL memory is addressed by either the PDL pointer or the PDL index. The PDL pointer is the stack pointer and points to the top element on the stack. The PDL index is an auxiliary stack pointer which is used for accessing other regions of the stack. This is the only means of addressing the PDL memory. The PDL pointer and index can be read via the MF bus as M sources and can be written to by the O bus.

The Q register assists in multiplication and division. It is used as a 32-bit bidirectional shift register with right-shift and left-shift serial inputs. The Q register can be parallel-loaded directly from the ALU output and is used by the ALU function logic to control the multiply-step and divide-step ALU operations.

The macroinstruction program count is kept in the macro location counter. The location counter has a special path to the virtual memory address (VMA) register (not shown on the data path block diagram) for starting a macroinstruction fetch. The count is offset by one bit to account for the 16-bit instructions that are stored two per word. Logic associated with the counter and the macroinstruction buffer (IBUF) register reduce fetching by keeping the second instruction of any 32-bit instruction word and using it without a fetch if possible.

The MCR provides for control and testing of various machine signals. The highest priority pending NuBus interrupt can be read from this register. The enable signals for interrupts and other hardware functions are controlled by storing the signals in the MCR.

The MD register and the VMA register provide the interface to memory. A read operation can be started by loading the VMA register with an address. A write operation can be started by loading the VMA register with an address and the MD register with data. Memory interface operations are discussed later in this section.

The μPCS is the microprogram call/return stack addressed by the μPCS pointer. The μPCS can also be used for temporarily storing data. Loading and storing μPCS from the data part of the machine also helps in context switching where the μPCS must be saved and restored. The μPCS pointer can be read and written into to facilitate initialization.

A microinstruction can be modified by the previous microinstruction. This feature is called instruction modify (IMOD). IMOD logically ORs the result of a microinstruction with part of the next microinstruction. The result can modify either the high-order 24 bits or the low-order 32 bits of the next microinstruction. The microinstruction fields are arranged so that no field crosses the boundary between IMOD-low and IMOD-high. One use of IMOD is to set the rotation count and/or field length of the byte microinstruction that follows. IMOD logic is discussed in paragraph 4.3.4, Processor Control Paths.

All registers and memories can be read from and written to. This aids diagnostics and simplifies microprograms that must access all or most of the machine state.

Processor Control Paths

4.3.4 The control paths (Figure 4-5) can be divided into two parts. One part deals with instructions, and one part deals with instruction addresses.

The currently executing microinstruction is in the IR from which all control signals originate. The instruction is fetched from either the writable control store RAM (called the I memory) or the self-test/loader PROM. The instruction goes through the IMOD logic to the IR. A flag in the MCR determines whether the instruction is fetched from the writable control store RAM or the self-test/loader PROM. The PROM is enabled automatically on power-up and also when a boot load is requested.

The PC is a register that contains the address of the instruction currently being fetched from either the writable control store RAM or the self-test/loader PROM. On each machine cycle, the PC is loaded from one of several sources based on the control operation in that cycle. The normal control operation is to increment the PC. Other control operations are jump, call, return, abort, and dispatch. There is a source for the PC corresponding to each of these. The jump or call address comes from the jump address field of the IR. The return address comes from the μPCS top-of-stack (popped). The abort target address is generated by the abort (trap) address generation logic. The dispatch address comes from the dispatch memory.

The dispatch memory is addressed by logically ORing the selected dispatch source field with the dispatch address field of the IR. The selected dispatch source field is normally from zero to seven low-order bits of the R bus. The selected dispatch source field can also be taken from the data-type field of the MF bus or from selected fields of IBUF. When taken from selected fields of IBUF, this field is used to decode macroinstructions.

A call control operation is a jump operation with a pushing of the return address on the μPCS. A return address select multiplexer selects a return address of my-own-address, next-instruction-address, or instruction-after-next-address. These are needed to get the proper return address with various pipelining options as discussed in the processor timing description that follows.

Figure 4-5 Control Path Block Diagram
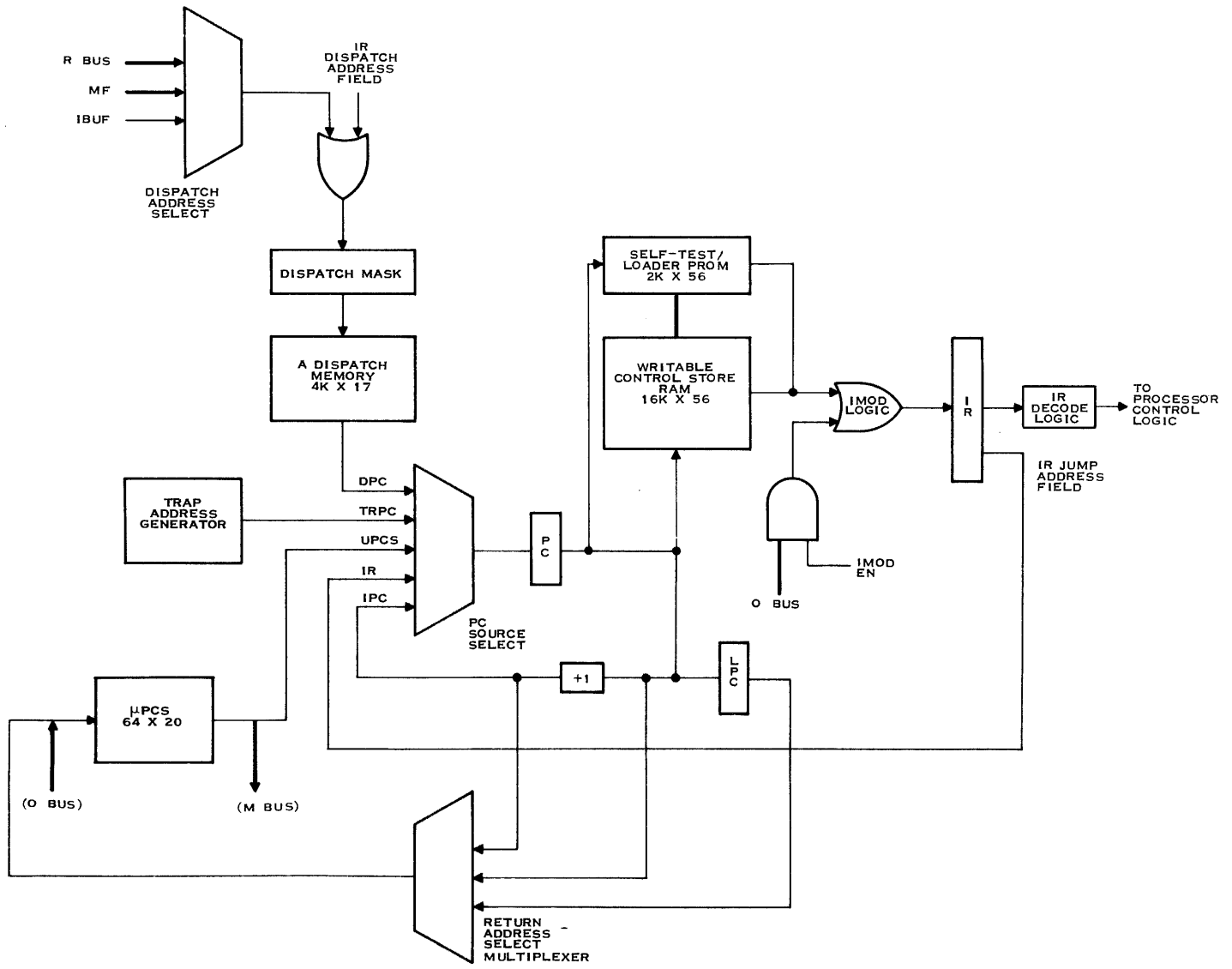
Processor Timing   **4.3.5**   The processor has one level of programmer-visible control pipelining, although some data pipelining is largely invisible to the programmer. Data pipeline timing is illustrated in Figure 4-6; control pipeline timing is shown in Figure 4-7.

Data Pipeline   **4.3.5.1**   Note in Figure 4-6 that each instruction consists of two machine cycles. Each machine cycle contains two minor cycles. Thus, each instruction contains four minor cycles:

■ Source (SRC)

■ Execute (EXEC)

■ Dead (DEAD)

■ Destination (DEST)

Figure 4-6   **Data Pipeline Timing**

DATA PIPELINE TIMING

NOTE.
t =ONE MACHINE CYCLE
I$_t$ =ONE INSTRUCTION CYCLE
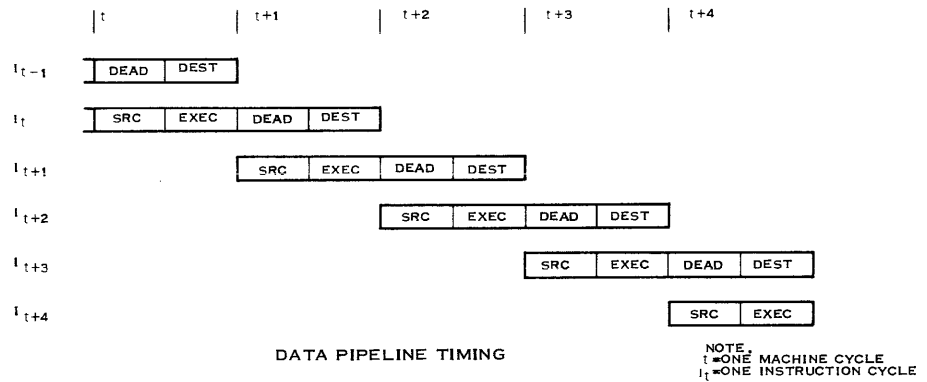
Figure 4-7 **Control Pipeline Timing**

	t	$t+1$	$t+2$	$t+3$	$t+4$

I_{t-1} ⌐ PENDING DEST ⌐

I_t [DECODE & EXEC | PENDING DEST ⌐]

I_{t+1} [FETCH | DECODE & EXEC | PENDING DEST ⌐]

I_{t+2} [PC | FETCH | DECODE & EXEC | PENDING DEST ⌐]

I_{t+3} [PC | FETCH | DECODE & EXEC | PENDING DEST ⌐]

I_{t+4} [PC | FETCH | DECODE & EXEC]

CONTROL PIPELINE TIMING

NOTE
t = ONE MACHINE CYCLE
I_t = ONE INSTRUCTION CYCLE
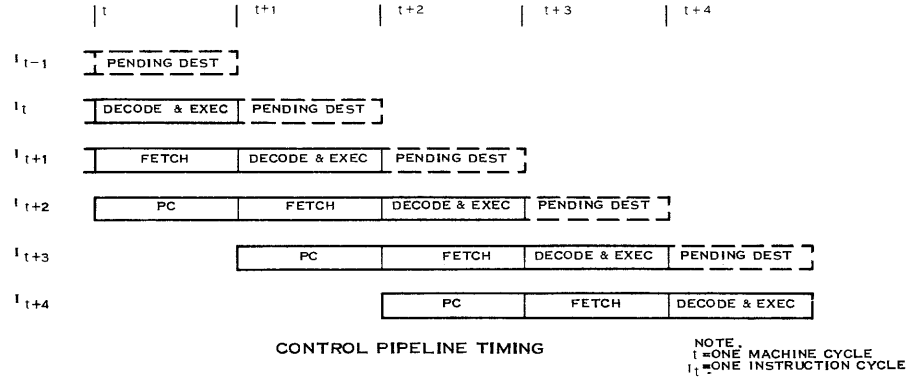
Instruction execution is overlapped so that as soon as two minor cycles of one instruction are completed, a new instruction begins. That is, the source cycle of the current instruction occurs during the dead cycle of the previous instruction. Similarly, the current execute cycle occurs during the previous destination cycle. Minor cycle operations are as follows:

■ Source cycle — Fetches the sources for execution.

■ Execute cycle — Operates upon the sources and produces the result. Stores the results for fast destinations at the end of this cycle.

■ Dead cycle — Holds the result of the instruction in the last-result register (L) pending use in the destination cycle. AL, ML, and μPCSL data registers are examples of L registers.

■ Destination cycle — Stores the results for slow destinations from the previous instruction.

Any destination with short set-up time (a register, for example) is considered a fast destination. Any destination with a long set-up time (such as a memory) is considered a slow destination.

This pipelining of data can cause problems since the result of the previous instruction has not yet been stored during a source cycle. To compensate for this, pass logic has been provided on the A memory, M memory, PDL memory, and μPCS. Pass logic notices that the source needed is the same as the destination pending and uses the L register as the source instead.

Control Pipeline **4.3.5.2** Normally the control pipeline (Figure 4-7) consists of three stages:

■ PC

■ Fetch

■ Decode and Execute

Each stage requires one machine cycle to complete. A fourth stage is added for execution of the call microinstruction — pending destination. A complete microinstruction consists of the following:

■ PC stage — Generates PC value to be used in the next stage.

■ Fetch stage — Fetches the instruction from memory and stores it in IR.

■ Decode and execute stage — The instruction is decoded and executed.

Microinstructions overlap in the following manner: while a decode and execute stage is being carried out in the current instruction (I_i), a fetch stage is being done for the next instruction (I_{i+1}), and a PC stage for instruction I_{i+2}. The decode and execute stage of I_i determine the PC value for I_{i+2}. However, it has no effect on I_{i+1} since this instruction is already in the pipeline at the time a transfer of control decision is made.

To program such a machine, jump instructions would have to be moved one instruction before the point at which the transfer should occur, and many jump instructions would be followed by no-operation (NOP) instructions.

To prevent the waste of expensive control store memory on NOP instructions, an N bit is provided on jump instructions. The N bit, when set, inhibits the execution of the next instruction. If the jump condition is true and the N bit is set, the instruction in the pipeline is inhibited (forced to a NOP). If the microprogrammer can arrange for the instruction following the jump to be useful whether the branch is taken or not, the instruction in the pipeline can be executed by specifying an N bit of 0, thereby gaining a speed advantage.

Since calling-type transfers must save the return address on the stack, the proper return address must be chosen by examining the N bit. If the N bit is set, the return address is the instruction after the call. That address is already in the PC (since it is in the pipeline). If the N bit is not set, the return address is the second instruction after the call. That address is PC + 1 (or incremented PC).

Memory Map and Memory Interface

4.3.6 The processor local bus interface to the system memory and peripherals is via the VMA and MD registers. Each of these registers is assigned several O bus destinations, and all destinations load the register. Only one destination loads the register, and one starts an unmapped (physical address) read operation. Another destination starts a mapped (virtual address) read operation, and the others start mapped and unmapped write operations.

Figure 4-8 is a block diagram of the memory map logic showing the interface of the local bus to the local memory array/NuBus interface. Note that the processor contains a NuBus slave-only interface to support reset operations, interrupts, the configuration ROM, the configuration register, and the flag register.

On an unmapped read or write operation, the content of the VMA is used directly as the local bus address. On a mapped read or write operation, the VMA is translated by the virtual memory map, and the results of the translation are used as the local bus address. This address is placed on the local bus where it is used by a memory board. If the address falls within the memory's address space, the local memory responds without starting a NuBus access. Otherwise, the system NuBus is acquired and a NuBus read or write operation is initiated. The system interface board is connected to the local bus and can also respond to the processor's memory requests.

The memory map is divided into two stages and operates on 25-bit addresses to produce 32-bit NuBus addresses. The low-order 25-bits of the VMA are the Lisp-field Q-pointer. Only the Q-pointer field of the VMA is examined by the map; the other bits of the VMA are ignored. This pointer is the virtual address to be mapped.

Figure 4-9 is a diagram of the memory mapping process. The virtual address is divided into the virtual page number and the page offset. The page offset bypasses the map and is not changed. The page offset is 8 bits, setting the page size at 256 words (1K byte). The virtual page number has 17 bits, allowing 128K virtual pages in the system.

The virtual page number is further divided into a virtual page block number field consisting of 12 bits and a virtual page block offset field consisting of 5 bits. The virtual page block number is used to address the 4K by 12-bit level 1 (LVL1) memory map to produce a level 2 (LVL2) block number. The 7-bit LVL2 block number is combined with the 5-bit virtual page block offset to address the LVL2 map. The 22 bits from the LVL2 map are combined with the page offset to produce a local bus address. Virtual address mapping always produces a word address since logic in the processor keeps the two LSBs of the 32-bit address set to zeros.

The map also stores other information, some of which is available for software use. The Lisp system uses this information to aid garbage collection and to provide several other features. The garbage collector (GC) performs dynamic storage reallocation by periodically searching memory to locate data structures that are no longer needed. This memory is then made available for new data structures. Some map information is examined by hardware and can signal a fault condition. The microcode must poll for map faults and should do so after starting each mapped memory access. A fault is signaled if the map entry is invalid, if a write operation is attempted to a read-only page, or if software has marked this page as special. In addition, two map outputs are available for use in dispatching. One output is an information bit from the LVL1 map. The other output is computed from the GC volatility fault logic.

Figure 4-8   Map Logic Block Diagram
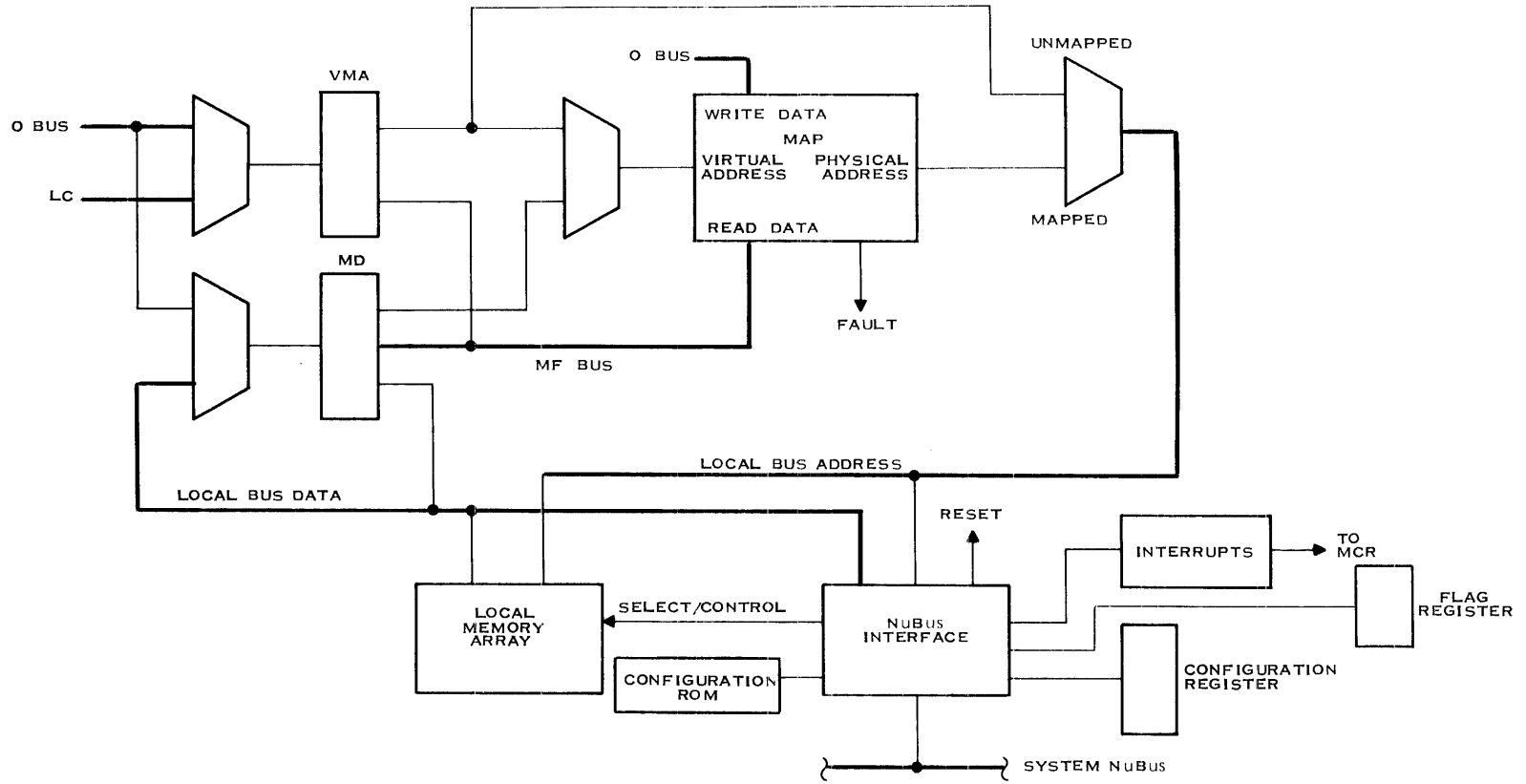
Figure 4-9   Memory Mapping Process
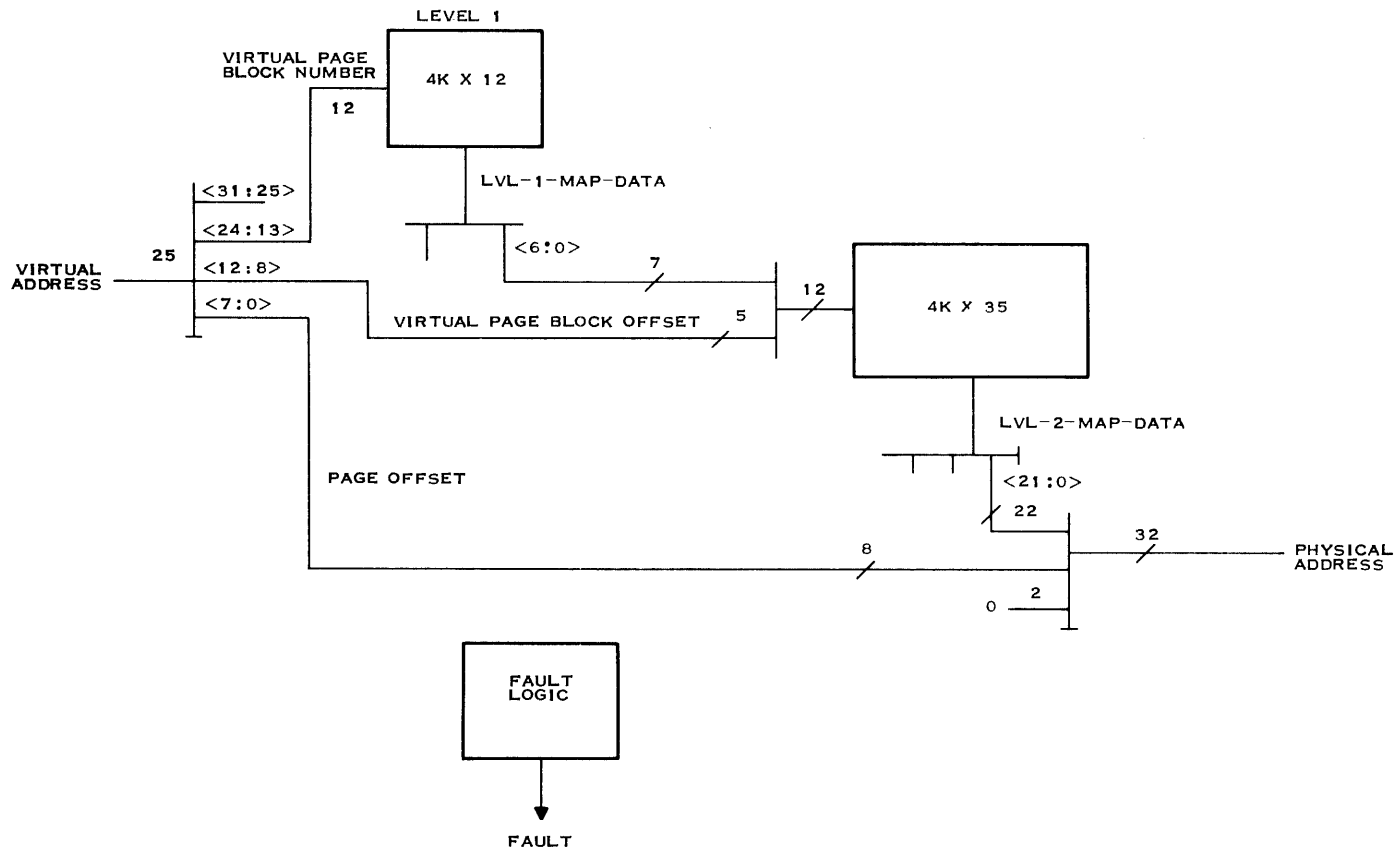
Normally, the map translates the 17-bit virtual page number portion of the Q-pointer field of the MD register contents; the VMA register is used during a mapped cycle. The map can be read and written into at the entry selected by the contents of the MD register through MF bus sources and O bus destinations.

Bus Structure　**4.3.7**　Figure 4-10 shows the bus structure of the processor board. There are numerous buses throughout the processor. Each bus is discussed separately. Numbers within the blocks on the diagram are logic-device identifying numbers.

A Bus　**4.3.7.1**　The A bus feeds the B-input of the ALU, (hereafter referred to as the A side of the ALU), the O bus multiplexer, the type-field comparator, the dispatch memory transceivers, and the control store data transceivers. The A bus is 32 bits wide. Sources for the A bus are as follows:

- The A scratchpad memory and the A memory (through the A memory register)

- The AL data register (through the A memory register)

- The dispatch memory transceivers

- The control store data transceivers

NOTE:　When reading either the 24-bit control store memory output or the 17-bit dispatch memory to the A bus (through their respective data transceivers), the high-order bits of the A bus are not zero-filled. The state of these high-order bits is not determined.

M Bus　**4.3.7.2**　The M bus feeds the A-input of the ALU, the barrel shifter, the type-field comparator, the type-classifier RAM, and the control store data transceivers.

The M bus has a number of sources, one of which is the MF bus. These buses are 32 bits wide. When accessing sources that are less than 32 bits wide, the zero-generation logic forces the unused bit positions on the M and MF buses to zero. The M and MF bus sources are as follows:

- M bus sources:

  - M memory

  - PDL memory

  - MD register

  - Control store data transceivers

  - μPCS — 20 bits

- MF bus sources:

  - Dispatch constant register — 10 bits

  - PDL index register — 10 bits

  - PDL pointer — 10 bits

  - MCR

  - μPCS pointer — 6 bits

  - Location counter — 26 bits

  - Q register

  - Virtual memory mapping RAM with map data registers:

    - Memory map data LVL1 — 16 bits

    - Memory map data LVL2 control — 16 bits

    - Memory map data LVL2 page frame number — 22 bits

  - VMA register

  - IBUF register — 16 bits (This contains the current instruction.)

  - The argument-offset field of the IBUF register's current instruction — 6 bits

  - The branch-offset field of the IBUF register — 9 bits

  - Zero-generation logic

Figure 4-10   Processor Detailed Block Diagram

Figure 4-10   Processor Detailed Block Diagram
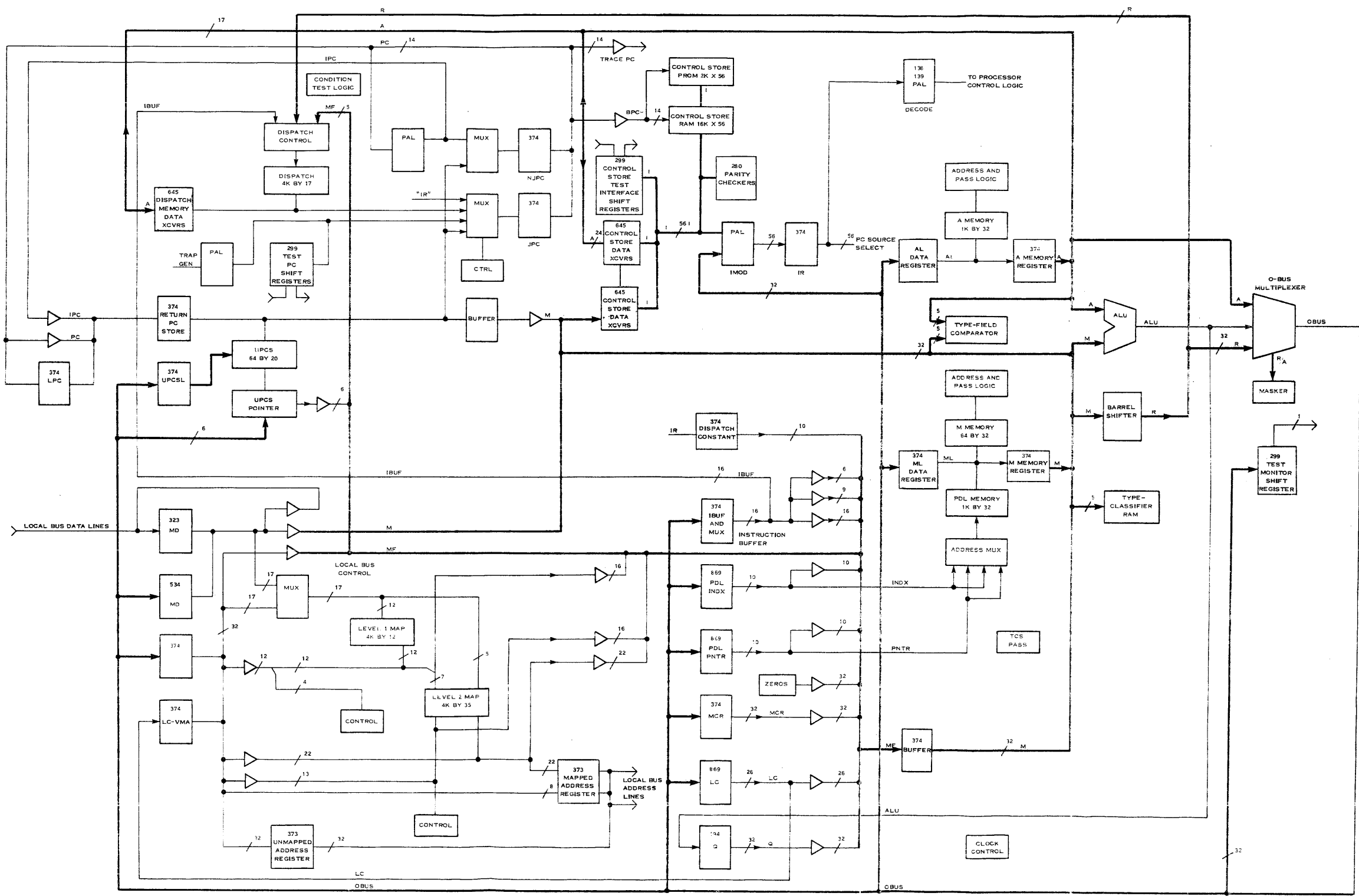
R Bus   **4.3.7.3**   The R bus is the output of the barrel shifter. The 32-bit R bus is applied to the O bus multiplexer and masker. A single bit of the R bus is applied to the transfer of control condition test logic (not shown on the block diagram), and seven bits are applied to the dispatch address selector in the dispatch control logic.

I Bus   **4.3.7.4**   The I bus is the microstore data bus. The bus is 56 bits wide and connects both control store memories (PROM and RAM) to the control store data transceivers, the control store test interface shift registers, the next-instruction-modifier logic (IMOD), and the parity checkers.

O Bus   **4.3.7.5**   The source for the O bus is a multiplexer with the following inputs:

- A bus

- ALU bus

- ALU bus shifted right one bit

- ALU bus shifted left one bit

- R bus

- ALU output in mirror image

- ALU bus with the output sign extended from ALU(24)

Through masking and selection, the O bus multiplexer provides an output consisting of the pointer field of the ALU output. However, the CDR code and data-type fields of the ALU output are replaced with the corresponding fields of the A bus. The O bus feeds the following:

- PDL index register (10 bits)

- PDL pointer (10 bits)

- Location counter (26 bits)

- VMA register

- MD register

- AL and ML data registers (A, M, and PDL memories)

- Next microinstruction modify logic (IMOD)

- μPCS pointer (6 bits)

- μPCS data (through μPCSL register, 20 bits) — μPCSL equals microprogram counter stack last result.

■ O bus test monitor shift register

■ IBUF register

■ MCR

There are both true and complement versions of the O bus. The complement of the O bus feeds the MD register, the AL and ML data registers, and the O bus test-monitor shift register. All other O bus destinations are connected to the true bus. This division of outputs is not shown on the simplified block diagram. The O bus output is split because of the limited drive capability of the O bus test-monitor shift register.

Local Bus and NuBus

4.3.7.6   The local bus is the interface between the processor and high-speed memory. The NuBus is the interface from the processor to the mass storage controller(s) and the NuBus Ethernet controller(s). Logic on the memory board provides an interface between the local bus and the NuBus for communications with these devices. The bit-mapped graphics display is accessed on the local bus. The NuBus interface is described in detail in later paragraphs in this section.

Processor Clock

4.3.8   The processor crystal-controlled master clock is a 28-megahertz clock from which a 14-megahertz minor-cycle clock and a 7-megahertz microinstruction clock are generated.

The operation of the microinstruction clock is described here to clarify explanations of the operation of various parts of the processor. During the source phase (clock high), data sources from internal memories and registers are accessed. During the low portion of the clock, data from memories internal to the processor is operated upon and the result of the previous microinstruction is stored in the internal memories. During the operation/ destination phase, the result of the current microinstruction is stored in internal registers. The 14-megahertz minor-cycle clock is used to control synchronous logic that must be clocked at twice the rate of the microinstruction clock.

Refer to Figure 4-10, the block diagram of the processor. During the time the clock is high, data is enabled from the source register or internal memory in preparation to being clocked into the registers that are sources for the A bus and the M bus. The A memory feeds the A bus through the A memory register that is clocked on the falling edge of the clock. On the falling edge of the clock, data from the A memory is captured and the data on the A bus remains stable while the result of the previous microinstruction is being stored in the A memory. Data transfers from M bus sources to the M bus occur in a manner similar to that described for the A bus.

During the low portion of the clock, the ALU or barrel shifter completes the operation on the data and places the result on the O bus. On the next rising edge of the clock, the data on the O bus is stored in the destination registers unless the destination is the A, M, PDL, μPCS, or map memory. If the destination is A, M, or PDL memory, the result is held in the AL and ML data registers until the second half of the microcycle that has just been started. At that time, the data from the AL and/or ML data registers is stored into the A, M, or PDL memory. The AL and ML data registers always contain the same data. (Two sets of registers performing the same function are required for electrical reasons.) The μPCSL register holds data at the end of a microcycle if the destination is μPCS. The PDL memory has the same timing as the M memory. Data for the map memories is held in the VMA register. The nominal clock period for the processor is 143 nanoseconds. The clock logic is synchronous and contains interface logic to the test interface that permits an external clock to replace the on-board clock to facilitate speed-margining of the processor. The clock logic is connected to the test interface. The test interface can control the clock to allow it to be single-stepped.

The clock logic also contains a long clock function that allows the clock to be held in the low state for (multiple) additional quarter-clock period(s). This means that multiples of 35.71 nanoseconds of additional low time can be provided to complete execution of the current operation. This feature allows the clock to operate at its fastest possible rate for most operations and to operate in a long clock mode when a long logic path is selected. The hardware supports only the provision of 35.71 or 71.43 nanoseconds of additional time in response to long clock requests. Long clock requests can only be invoked by the hardware.

The minor-cycle clock is also affected by the long clock feature. The minor-cycle clock is held low in the operation/destination phase of the microinstruction clock.

The memory controller that is part of the local memory board runs off the 28-megahertz master clock of the processor and is not affected by the long clock feature. The memory controller has a local bus clock controller associated with it that generates the local bus clock and synchronizes the local bus clock with the processor clock before the start of any memory cycle. If the VMA register or the MD register is accessed by microcode while a memory cycle is in progress, the processor clock will stop. The local bus clock controller is capable of restarting the processor clock within 35.71 nanoseconds.

NuBus Interface   **4.3.9**   The following paragraphs describe the NuBus interface requirements. The NuBus interface responds to the following:

■ Full-word accesses

■ Least significant halfword accesses

■ Least significant byte accesses

All other accesses result in an error acknowledge.

NuBus Slave   **4.3.9.1**   By serving as a slave to the NuBus, the processor provides access to its configuration PROM, configuration register, and flag register; it also allows another NuBus master to post events to the processor. The NuBus slave controller does not support block transfers.

NuBus Address   **4.3.9.2**   The starting address of the processor is determined by the ID
Assignments   code for the slot the processor occupies. There are no address switches or jumpers on the processor. The configuration PROM occupies 256 bytes of NuBus memory space from > FSFFFC00 through > FSFFFFFF. The configuration register is located at > FSD00000. The flag register is located at > FSC00000. Table 4-1 lists the addresses used for posting events to the processor.

Table 4-1   **Processor Event-Posting Addresses**

NuBus Address (Hexadecimal)	Interrupt Priority Level (Decimal)	
> FSE003C	15	(Lowest)
> FSE0038	14	
> FSE0034	13	
> FSE0030	12	
> FSE002C	11	
> FSE0028	10	
> FSE0024	9	
> FSE0020	8	
> FSE001C	7	
> FSE0018	6	
> FSE0014	5	
> FSE0010	4	
> FSE000C	3	
> FSE0008	2	(Highest)
> FSE0004	1	Warm Boot
> FSE0000	0	Power Fail

Configuration Register **4.3.9.3** The configuration register contains two bits that can be written to. Bit two of the register is used to force the fault LED on when set to one. However, setting this bit to zero may not turn off the LED, because the processor may have failed self-test, which causes the fault LED to be lit. Bit zero, when set to 1, causes the board to reset. This bit must be set to zero before another reset can be issued.

Flag Register **4.3.9.4** The flag register contains three bits that can be read by other NuBus devices. Bit two indicates subsystem pass/fail. The processor always indicates pass. Bit one indicates board-level self-test pass/fail (1 equals failed). Bit zero indicates self-test status (0 equals completion). The processor always reports the appropriate value.

Configuration ROM **4.3.9.5** A configuration ROM on the processor is accessible from the NuBus. Table 4-2 provides a listing of information contained in the configuration ROM.

Table 4-2 Configuration ROM Contents

Address Range	Data Value	Definition
> FSFFFECC	> 04	Restart event offset. Roughly equivalent to warm-boot chord in Lisp.
> FSFFFED0	> 00	
> FSFFFED4	> E0	
> FSFFFED8	> 08	System test and boot master (STBM) event offset. STBM interrupt for communication during system booting.
> FSFFFEDC	> 00	
> FSFFFEE0	> E0	
> FSFFFEE4	> 00	Event offset. Starting location of memory mapped interrupt space. First location is interrupt 0; second is interrupt 1, and so on (32-bit word addressing). The processor has a total of 16 interrupts.
> FSFFFEE8	> 00	
> FSFFFEEC	> E0	
> FSFFFEF0	> 00	Nonvolatile RAM (NVRAM). Does not apply to the processor.
> FSFFFEF4	> FF	
> FSFFFEF8	> FF	
> FSFFFEFC	> FF	
> FSFFFF00	> 10	Resource type. This is a one-byte binary field that allows the initial program loader to determine the resources required to perform the initial program load (IPL).
> FSFFFF04	> C3	Configuration ROM ID.
> FSFFFF08	> 02	Self-test time (log2 seconds).
> FSFFFF0C	> 03	ROM layout. The value > 03 indicates that the current processor ROM layout conforms to the Explorer system specification. Each revision of the document that affects the ROM will increment this value.

Table 4-2   Configuration ROM Contents (Continued)

Address Range	Data Value	Definition
> FSFFFF10	> 25	ROM flags. This one-byte field reports certain system information inherent to the processor board.
> FSFFFF14	> 00	Flag register offset. This three-byte field indicates the offset between the start of the processor control space and the processor flag register address.
> FSFFFF18	> 00	
> FSFFFF1C	> C0	
> FSFFFF20	> FF	Diagnostic offset. (None)
> FSFFFF24	> FF	
> FSFFFF28	> FF	
> FSFFFF2C	> FF	Device driver offset. (None)
> FSFFFF30	> FF	
> FSFFFF34	> FF	
> FSFFFF38	> 00	Configuration register offset. This three-byte field indicates the offset between the start of the processor control space and the lowest address of the processor configuration register.
> FSFFFF3C	> 00	
> FSFFFF40	> D0	
> FSFFFF44	> 30	This is the ASCII code for the complete processor board, TI part number 2243881-0001.
> FSFFFF48	> 30	
> FSFFFF4C	> 30	
> FSFFFF50	> 30	
> FSFFFF54	> 32	
> FSFFFF58	> 32	
> FSFFFF5C	> 33	
> FSFFFF60	> 38	
> FSFFFF64	> 30	
> FSFFFF68	> 34	
> FSFFFF6C	> 30	
> FSFFFF70	> 2D	
> FSFFFF74	> 30	
> FSFFFF78	> 30	
> FSFFFF7C	> 30	
> FSFFFF80	> 31	
> FSFFFF84	> 43	Board type. The first three addresses contain the ASCII code for CPU, indicating that this is a processor board. The remaining five bytes indicate the memory size, cache size (log2 in K bytes), zero, and CPU type.
> FSFFFF88	> 50	
> FSFFFF8C	> 55	
> FSFFFF90	> 00	
> FSFFFF94	> 00	
> FSFFFF98	> 00	
> FSFFFF9C	> 00	
> FSFFFFA0	> 00	
> FSFFFFA4	> 54	Vendor identification. These four addresses contain the ASCII code for TIAU, indicating that the board was manufactured by Texas Instruments Data Systems Group, Austin. This code can vary, depending on the manufacturing site.
> FSFFFFA8	> 49	
> FSFFFFAC	> 41	
> FSFFFFB0	> 55	

Table 4-2   Configuration ROM Contents (Continued)

Address Range	Data Value	Definition
> FSFFFFB4	> 08	ROM size. (log2 in bytes)
> FSFFFFB8 > FSFFFFBC	> FF > FF	Cyclic redundancy check (CRC) signature. This signature is the result of a cyclic redundancy check of the ROM contents. The method is transparent but should always yield the same signature, indicating that the ROM contents are correct.
> FSFFFFC0 > FSFFFFC4 > FSFFFFC8 > FSFFFFCC > FSFFFFD0 > FSFFFFD4 > FSFFFFD8		Revision level. This is the ASCII code that indicates the current revision level (*) of the board. The asterisk indicates an unrevised board. The next revision level will be A and each subsequent revision level will increment this value.
> FSFFFFDC > FSFFFFE0 > FSFFFFE4 > FSFFFFE8 > FSFFFFEC > FSFFFFF0 > FSFFFFF4 > FSFFFFF8 > FSFFFFFC		Serial number. These addresses contain codes that indicate the week and year of manufacture, the manufacturing site, and a serial number.

NOTES:

1. A weighted log2 value is a coding scheme in which the most significant digit is a hexadecimal mantissa (M) and the least significant digit is an exponent (E) to the base 2. The value is $M*(2**E)$.

2. The ROM is accessed by byte only, not words. Valid header addresses > FSFFFF00, -04, -08, . . . , $-F4$, F8, FC. ROM data is valid only on the least significant byte of NuBus.

3. The binary fields are stored such that the logically highest NuBus address of each field contains the most significant byte, while the lowest address contains the least significant byte.

4. ASCII fields are stored as strings, with the first (most significant) character at the lowest address. Characters are stored one per word in byte 0, with contiguous word addresses.

5. Any field containing > FF indicates that the field is invalid, with the exception of the CRC (> 00 and > FF are valid signatures).

6. Any unused field should be left unblown.

Interface Connectors and Signal Assignments

4.3.10 Each of the three 96-pin DIN interface connectors on the processor has three columns of pins; each column consists of 32 pins. On the connectors, the columns are marked rows A, B, and C. The pins for each column are marked 1 through 32. On the processor, however, the signal pins are identified as numbers 01 through 96. Numbers in parentheses in the connector pin assignment tables, (01) through (96), indicate processor pin number designations.

NOTE: DIN is an acronym for Deutsches Institut fuer Normung, which translates to the German Institute of Standards.

NuBus Connector Signals

4.3.10.1 Functions of the NuBus signals and processor pin numbers are shown in Table 4-3. Connector pin assignments (by column) for the NuBus connector P1 are shown in Table 4-4.

Table 4-3   NuBus Signals and Pin Assignments

Signature	Pin	Definition
ACK–	28	Transfer acknowledge. This signal is driven for only one clock period by the processor when it is addressed as a slave device on the NuBus. ACK– is asserted to indicate the completion of a transaction.
AD31–	21	These NuBus address/data signals are multiplexed to carry a 32–bit address at the
AD30–	85	start of a cycle and 32 bits of data during the remainder of the cycle. These signals
AD29–	20	can be generated by any master on the NuBus. AD0– and AD1– carry address
AD28–	84	information during the start cycle of a byte transaction and control information
AD27–	19	during the start cycle of a nonbyte transfer operation.
AD26–	83	
AD25–	18	
AD24–	82	
AD23–	17	
AD22–	81	
AD21–	16	
AD20–	80	
AD19–	15	
AD18–	79	
AD17–	14	
AD16–	78	
AD15–	13	
AD14–	77	
AD13–	12	
AD12–	76	
AD11–	11	
AD10–	75	
AD9–	10	
AD8–	74	

Table 4-3   NuBus Signals and Pin Assignments (Continued)

Signature	Pin	Definition
AD7–	9	
AD6–	73	
AD5–	8	
AD4–	72	
AD3–	7	
AD2–	71	
AD1–	6	
AD0–	70	
ARB3–	25	These signals form an arbitration binary code and are given by contenders for the bus. The distributed arbitration logic samples the signals to establish which of the vying devices becomes the next master on the NuBus. These signals are not implemented on the processor.
ARB2–	89	
ARB1–	24	
ARB0–	88	
CLK–	96	NuBus clock. CLK– synchronizes bus arbitration and data transfers between system modules on the NuBus. The clock signal has a nominal frequency of 10 megahertz with a 75 percent duty cycle. In general, signals are changed at the rising edge of CLK– and tested at the falling edge.
EARTH	87	
GND	2	Ground.
	22	
	23	
	31	
	34	
	35	
	44	
	45	
	46	
	47	
	48	
	49	
	50	
	51	
	52	
	53	
	54	
	55	
ID3–	27	These binary–coded slot identification signals are hard-wired into each slot so that each installed board can be linked to its physical (slot) location.
ID2–	91	
ID1–	26	
ID0–	90	
RESET–	65	Reset. Assertion of this signal returns all boards on the NuBus to their initial power–up state.
RQST–	30	Bus request. This signal can be asserted by any NuBus master that wants control of the bus. The signal is not implemented on the processor.

Table 4-3   NuBus Signals and Pin Assignments (Continued)

Signature	Pin	Definition
SP– SPV–	4 3	System parity (SP–) transmits parity information between NuBus devices that implement NuBus parity checking. SPV– is a system parity valid signal to indicate that SP– is being used. These signals are both wired high on the processor and are not used.
START–	90	Transfer start. This signal is driven for only one clock period by the current NuBus master at the beginning of a transaction. START– indicates to the slaves on the NuBus that the address/data signal lines are carrying a valid address.
TM1– TM0–	5 69	Transfer mode signals. These signals are asserted by the current NuBus master during start cycles to indicate the type of bus operation being initiated. The signals are also driven by NuBus slaves during acknowledge cycles to indicate the type of acknowledgement as shown below:

TM1–	TM0–	Type of Acknowledgement
L	L	Bus transfer complete
L	H	Error
H	L	Bus time-out error
H	H	Try again later

Signature	Pin	Definition
Vcc	29 36 37 38 39 60 61 67 68 93 94	
–5 Vdc	40 41 42 43 56 57 58 59	Not used by the processor.
–12 Vdc	1 33	
+12 Vdc	32 64	

Table 4-4 **Processor NuBus Connector Pin Assignments**

Connector Pin No.	Signals		
	Column A	Column B	Column C
1	(01) − 12	(33) − 12	(65) RESET−
2	(02) GND	(34) GND	(66) GND
3	(03) SPV−	(35) GND	(67) Vcc
4	(04) SP−	(36) Vcc	(68) Vcc
5	(05) TM1−	(37) Vcc	(69) TM0−
6	(06) AD1−	(38) Vcc	(70) AD0−
7	(07) AD3−	(39) Vcc	(71) AD2−
8	(08) AD5−	(40) − 5	(72) AD4−
9	(09) AD7−	(41) − 5	(73) AD6−
10	(10) AD9−	(42) − 5	(74) AD8−
11	(11) AD11−	(43) − 5	(75) AD10−
12	(12) AD13−	(44) GND	(76) AD12−
13	(13) AD15−	(45) GND	(77) AD14−
14	(14) AD17−	(46) GND	(78) AD16−
15	(15) AD19−	(47) GND	(79) AD18−
16	(16) AD21−	(48) GND	(80) AD20−
17	(17) AD23−	(49) GND	(81) AD22−
18	(18) AD25−	(50) GND	(82) AD24−
19	(19) AD27−	(51) GND	(83) AD26−
20	(20) AD29−	(52) GND	(84) AD28−
21	(21) AD31−	(53) GND	(85) AD30−
22	(22) GND	(54) GND	(86) GND
23	(23) GND	(55) GND	(87) EARTH
24	(24) ARB1−	(56) − 5	(88) ARB0−
25	(25) ARB3−	(57) − 5	(89) ARB2−
26	(26) ID1−	(58) − 5	(90) ID0−
27	(27) ID3−	(59) − 5	(91) ID2−
28	(28) ACK−	(60) Vcc	(92) START−
29	(29) Vcc	(61) Vcc	(93) Vcc
30	(30) RQST−	(62) GND	(94) Vcc
31	(31) GND	(63) GND	(95) GND
32	(32) + 12	(64) + 12	(96) CLK−

NOTE:

Vcc = + 5 V.

Local Bus **4.3.10.2**   One 96-pin male DIN connector is used to connect the proces-
Connector Signals sor to the local bus on the backplane. Functions of the local bus signals and
processor pin numbers are shown in Table 4-5. Connector pin assignments
(by column) for the local bus connector P2 are shown in Table 4-6.

Table 4-5   Local Bus Signals and Pin Assignments

Signature	Pin	Definition
AD31	32	These are the 32-bit local bus address signals generated by the processor.
AD30	31	
AD29	30	
AD28	29	
AD27	28	
AD26	27	
AD25	26	
AD24	25	
AD23	24	
AD22	23	
AD21	22	
AD20	21	
AD19	20	
AD18	19	
AD17	18	
AD16	17	
AD15	16	
AD14	15	
AD13	14	
AD12	13	
AD11	12	
AD10	11	
AD09	10	
AD08	9	
AD07	8	
AD06	7	
AD05	6	
AD04	5	
AD03	4	
AD02	3	
AD01	2	
AD00	1	
BCLK–	53	BCLK– is the local bus clock and is generated by the processor. All local bus control and data transfers are synchronized to BCLK–.
BERR–	54	BERR– is a bus error signal asserted by the memory board to the processor to indicate that an error has been detected on a data transfer.
BS0–	45	BS0– is a local bus select signal generated by the processor. (This signal is not being used at present and therefore is grounded.)
DAT31–	96	These are the 32-bit data signals used by the processor for read and write transfers on the local bus.
DAT30–	95	
DAT29–	94	
DAT28	93	
DAT27	92	
DAT26	91	
DAT25	90	
DAT24	89	

Table 4-5   Local Bus Signals and Pin Assignments (Continued)

Signature	Pin	Definition
DAT23	88	
DAT22	87	
DAT21	86	
DAT20	85	
DAT19	84	
DAT18	83	
DAT17	82	
DAT16	81	
DAT15	80	
DAT14	79	
DAT13	78	
DAT12	77	
DAT11	76	
DAT10	75	
DAT09	74	
DAT08	73	
DAT07	72	
DAT06	71	
DAT05	70	
DAT04	69	
DAT03	68	
DAT02	67	
DAT01	66	
DAT00	65	
FAST–	56	This signal is not used at present and therefore is grounded.
GND	34	
	35	
	36	
	44	
	48	
	49	
	50	
	51	
	55	
	61	
	62	
	63	
LOCK–	42	LOCK– is generated by the processor. When LOCK– is asserted, the resources of the memory board currently being accessed by the processor are locked and cannot be accessed by the NuBus. When locked, the memory board stays locked until LOCK– returns to inactive.

Table 4-5 Local Bus Signals and Pin Assignments (Continued)

Signature	Pin	Definition
LTM0– LTM1–	33	These are the local bus transfer mode signals that are generated by the processor. LTM0– and LTM1– are used with AD00– and AD01– to form a four-bit binary code that informs the memory board of the type of transfer mode the processor is requesting. Command encoding of transfer modes of read and write operations of words, halfwords, or bytes is as follows:

LTM0–	LTM1–	AD1–	AD0–	Transfer Mode
L	L	L	L	Write byte 3
L	L	L	H	Write byte 2
L	L	H	L	Write byte 1
L	L	H	H	Write byte 0
L	H	L	L	Write halfword 1
L	H	L	H	Reserved
L	H	H	L	Write halfword 0
L	H	H	H	Write word
H	L	L	L	Read byte 3
H	L	L	H	Read byte 2
H	L	H	L	Read byte 1
H	L	H	H	Read byte 0
H	H	L	L	Read halfword 1
H	H	L	H	Reserved
H	H	H	L	Read halfword 0
H	H	H	H	Read word

Signature	Pin	Definition
MEMACK–	46	MEMACK– is a memory acknowledge signal asserted by the memory board to the processor to indicate that the current memory cycle will be completed on the next falling edge of the local bus clock.
MEMREQ–	41	MEMREQ– is a memory request signal generated by the processor. MEMREQ– is asserted by the processor when a memory cycle is required.
OPEN	43 47 52 57 59	
Reserved	58	
Spare	40	
Vcc	37 38 39 60	

NOTE:

There is no hardware on either processor or memory board to support block moves on the local bus.

Table 4-6   **Processor Local Bus Connector Pin Assignments**

Connector Pin No.	Signals		
	Column A	Column B	Column C
1	(01) AD00	(33) LTM0–	(65) DAT00
2	(02) AD01	(34) GND	(66) DAT01
3	(03) AD02	(35) GND	(67) DAT02
4	(04) AD03	(36) GND	(68) DAT03
5	(05) AD04	(37) Vcc	(69) DAT04
6	(06) AD05	(38) Vcc	(70) DAT05
7	(07) AD06	(39) Vcc	(71) DAT06
8	(08) AD07	(40) SPARE1	(72) DAT07
9	(09) AD08	(41) MEMREQ–	(73) DAT08
10	(10) AD09	(42) LOCK–	(74) DAT09
11	(11) AD10	(43) OPEN	(75) DAT10
12	(12) AD11	(44) GND	(76) DAT11
13	(13) AD12	(45) BS0–	(77) DAT12
14	(14) AD13	(46) MEMACK–	(78) DAT13
15	(15) AD14	(47) OPEN	(79) DAT14
16	(16) AD15	(48) GND	(80) DAT15
17	(17) AD16	(49) GND	(81) DAT16
18	(18) AD17	(50) GND	(82) DAT17
19	(19) AD18	(51) GND	(83) DAT18
20	(20) AD19	(52) OPEN	(84) DAT19
21	(21) AD20	(53) BCLK–	(85) DAT20
22	(22) AD21	(54) BERR–	(86) DAT21
23	(23) AD22	(55) GND	(87) DAT22
24	(24) AD23	(56) FAST–	(88) DAT23
25	(25) AD24	(57) OPEN	(89) DAT24
26	(26) AD25	(58) RESERVED	(90) DAT25
27	(27) AD26	(59) OPEN	(91) DAT26
28	(28) AD27	(60) Vcc	(92) DAT27
29	(29) AD28	(61) GND	(93) DAT28
30	(30) AD29	(62) GND	(94) DAT29
31	(31) AD30	(63) GND	(95) DAT30
32	(32) AD31	(64) LTM1–	(96) DAT31

NOTE:

Vcc = +5 V

Test Connector Signals   **4.3.10.3**   The third 96-pin connector at the bottom of the board is used as an interface to a factory/laboratory tester. These pins are not connected to any bus in a normal chassis configuration. Connections can be made to the test interface connector at the chassis bulkhead without removing the processor from the chassis. This allows testing of the processor in an undisturbed environment. Functions of the test interface signals and connector P3 pin assignments are included in the processor part of the *Explorer System Field Maintenance* manual, TI part number 2243141-0001.

Processor Macro-instructions

4.4 Macroinstructions are 16 bits in length and are stored two per 32-bit data word. The first macroinstruction is stored in the low order bits of the word; the other macroinstruction is stored in the high order bits of the word. There are five macroinstruction formats as follows:

■ Main instruction (class I)

■ Nondestination instruction (class II)

■ Branch instruction (class III)

■ Miscellaneous instruction (class IV)

■ Array reference immediate (AREFI) instruction (class V)

A brief description of each of the formats is provided in paragraphs that follow.

NOTE: The macroinstruction information may change with subsequent microcode and load band changes.

Main Instruction, Class I, Format

4.4.1 The basic macroinstruction format is the main instruction, class I, format shown in Figure 4-11. Certain values contained in the opcode field of the class I macroinstruction can indicate that the instruction is one of the other four classes of macroinstructions. The mapping of the opcode field into instruction class is shown in Table 4-7. Since at least one field of the main instruction format is shared with each of the other classes, the fields of the main instruction format are described first. The format for the other instruction classes is described in separate paragraphs, but descriptions of shared fields are not repeated.

Figure 4-11   Main Instruction, Class I, Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Offset field

Register field

Operation code field

Destination field
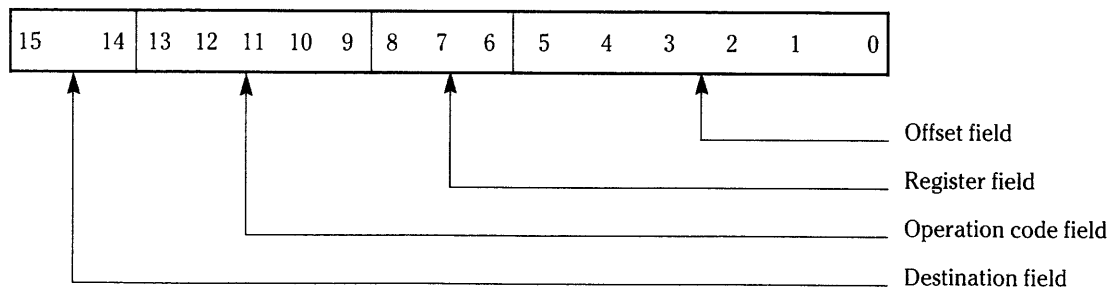
Table 4-7	Opcode-Field to Instruction-Class Map

Opcode	Class
0	I
1	I
2	I
3	I
4	I
5	I
6	I
7	I
8	I
9	II
10	II
11	II
12	III
13	IV
14	II
15	Unused
16	V
17	Unused
18	Unused
19	Unused
20	Unused
21	Unused
22	Unused
23	Unused
24	Unused
25	II
26	II
27	II
28	III
29	IV
30	II
31	Unused

Destination Field **4.4.1.1** The destination field consists of two bits that specify the destination of the result of the instruction. The destination field is common to classes I, IV, and V macroinstruction formats. Instructions in class II format have an implied destination (usually the PDL memory), and class III format instructions do not have a result. Coding of the destination field is shown in Table 4-8.

Table 4-8   **Destination Field Coding**

Code	Destination
0	Ignore
1	PDL
2	Return
3	Last

Ignore means that the result of the instruction is not stored but the indicators are still set. (See paragraph 4.4.1.3, entitled Indicators.) PDL indicates that the result of the instruction is pushed on to the PDL memory and that the indicators are set. Return means that the result of the current function is returned to the caller according to the destination of the call instruction. Last indicates that the result is to be pushed on to the PDL memory (as in destination PDL) as the last argument to a pending call. Destination last completes the calling of the function and transfers control to this destination.

Register and Offset Fields   **4.4.1.2**   The register and offset fields together specify the effective address of the operand. These fields are used in class I and class II instructions. The effective address is formed by using the register specified in the register field as the base register and adding the offset to it. The register field is coded as shown in Table 4-9.

Table 4-9   **Register Field Coding**

Code	Base Register
0	FEF
1	FEF + 64
2	FEF + 128
3	FEF + 192
4	Constants
5	Locals
6	ARGS
7	PDL/IVAR

NOTES:

FEF	=	functions entry frame
ARGS	=	arguments
IVAR	=	instance variable

The FEF and FEF+ registers are for addressing symbols and constants used in the function. The FEF+64, FEF+128, and FEF+192 forms increase the range of the FEF that can be used for symbols and constants. The constants register addresses a table that contains commonly used constants. The constants table is shown in Table 4-10.

NOTE: The effective address of an IVAR slot is the slot number plus one with reference to the self variable.

The locals register provides access to the locals block on the PDL of the currently active function. The ARGS register provides access to the argument block on the PDL of the currently active function. Effective addresses are determined by offset field values as shown in Table 4-11.

Indicators **4.4.1.3** The instruction set uses two indicators: nil and atom. The indicators can be tested in branch instructions (class III). Instructions that produce a result set the indicators (even if the destination is ignore). The nil indicator is set only if the result is the symbol nil. The atom indicator is set only if the result is not a list.

Nondestination Instruction, Class II, Format **4.4.2** Nondestination instructions have an effective address but no destination field. Most instructions produce a result that is stored in an implicit destination (usually the PDL). Nondestination instructions are most useful for stack operations such as arithmetic and a pop from the top of the stack.

The format of a nondestination instruction is shown in Figure 4-12. As shown, the three high-order bits of a nondestination instruction select one of eight nondestination operations (NDOP) within one of four nondestination groups as selected by the group field. Coding for nondestination group selection is shown in Table 4-12. The low nine bits form the register and offset fields as they did in class I instructions.

Table 4-10 **Constants Table**

Offset	Object Type	Object
0	Symbol	NIL
1	Symbol	T
2	Fixnum	0
3	Fixnum	1
4	Fixnum	2

Table 4-11   **Offset Field Value Effective Addresses**

Offset Field Value	Effective Address
63	The top of the PDL memory
56	An IVAR
0 to 31	The unmapped IVAR at the indicated slot number
32 to 55	The IVAR obtained by reading the selected slot of the self-mapping table

The eight NDOPs within each of the four groups are defined in Table 4-13.

Branch Instruction, Class III, Format   **4.4.3**   Branch (BR) instructions alter program flow within a function but cannot transfer outside the currently executing FEF. The unconditional BR instruction always transfers to the target. Conditional BR instructions test one of the indicators and transfer if the test is true. Two more complex instructions are provided for looping support.

BR instructions have no effective address nor destination field. Instead, the effective address field is used as a nine-bit branch offset. The branch offset is a signed two's complement quantity to be added to the location counter after it has been incremented to point to the instruction following the BR instruction. This sum is the location counter of the branch target.

If the branch offset is -1, then a long branch is indicated. The next 16 bits of the instruction stream are read and used as a signed two's complement quantity to be added to the location counter after it has been incremented to point to the instruction following the BR instruction. If the branch is not taken, execution continues with the instruction following the BR instruction. Note that even if the branch is not taken, the branch offset must be checked for a long branch and, if it is a long branch, the next halfword (the long branch offset) must be skipped.

Figure 4-12   Nondestination Instruction, Class II, Format
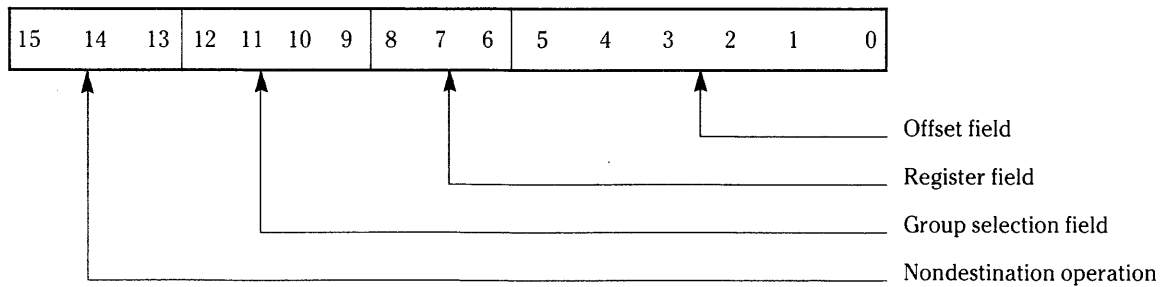
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Offset field

Register field

Group selection field

Nondestination operation

Table 4-12

Nondestination Group Selection

Code	Group
9	1
10	2
11	3
14	4

Table 4-13

Nondestination Decoding

Group	NDOP	Instruction
1	0	Illegal
	1	+ (plus)
	2	− (difference)
	3	* (times)
	4	/ (quotient)
	5	and
	6	XOR
	7	IOR
2	0	= (equal)
	1	> (greater)
	2	< (less)
	3	EQ
	4	SCDR
	5	SCDDR
	6	SETE−1+
	7	SETE−1−
3	0	bind
	1	bindnil
	2	bindpop
	3	setnil
	4	setzero
	5	push
	6	move
	7	pop
4	0	stack-closure-disconnect
	1	stack-closure-unshare
	2	make-stack-closure
	3	push-number
	4	stack-closure-disconnect-first
	5	push-CDR-if-CAR-equal
	6	push-CDR-store-CAR-if-CONS
	7	Illegal

The format of a BR instruction is shown in Figure 4-13. The high-order three bits of a BR instruction select the specific branch operation. This field occupies the same bit positions as the destination field and the high bit position of the class I opcode field. The remainder of the class I opcode field is 12 to select class III instructions. The remaining nine bits (the class I register and offset fields) are the branch offset. The coding of the selection of the operation field is shown in Table 4-14.

Miscellaneous Instruction, Class IV, Format

4.4.4 Miscellaneous instructions have a destination but no effective address field; most take ARGS on the stack. These instructions are used mainly for Lisp functions that are microcoded for speed and also for special purpose instructions.

The format of miscellaneous instructions is shown in Figure 4-14. The high-order two bits select the destination just as in the class I instruction format. Bits 9 through 13 (corresponding to the class I opcode field) must be either 13 or 29 to select a class IV instruction. If this field is 13, a group 0 miscellaneous instruction is indicated; if it is 29, a group 1 miscellaneous instruction is indicated. The remaining nine-bit field is the miscellaneous operation (MISCOP) field. This field selects an instruction within either group 0 or group 1.

Array Reference Immediate Instruction, Class V, Format

4.4.5 Array reference immediate (AREFI) instructions support abbreviated sequences for accessing one-dimensional arrays (also array leaders and instances) with a small constant index. The format of this class V instruction is shown in Figure 4-15. The kind of object to be referenced is indicated by the reference kind (REFKIND) field. Decoding of the REFKIND field is shown in Table 4-15. The index field is a six-bit index into the structure. The structure to be accessed is an argument on the stack.

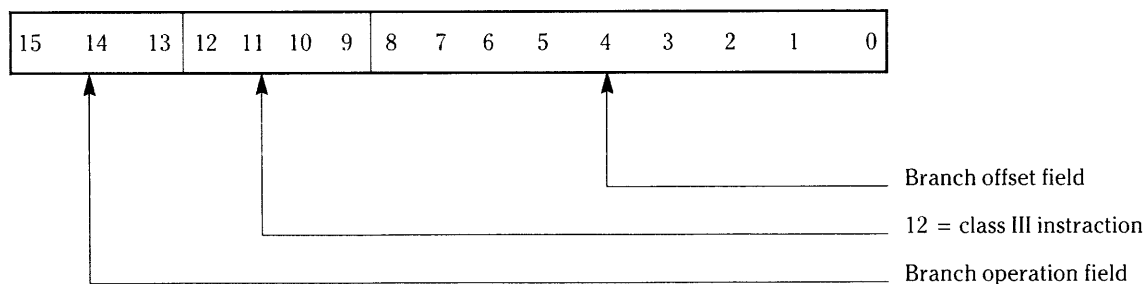## Figure 4-13   Branch Instruction, Class III, Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Branch offset field

12 = class III instraction

Branch operation field

Table 4-14   **Branch Operation Decoding**

Branch Operation	Instruction
0	branch
1	branch-nil
2	branch-not-nil
3	branch-nil-else-pop
4	branch-not-nil-else-pop
5	branch-atom
6	branch-not-atom
7	Illegal

Figure 4-14   **Miscellaneous Instruction, Class IV, Format**

Miscellaneous operation field

Class I opcode field

Destination field
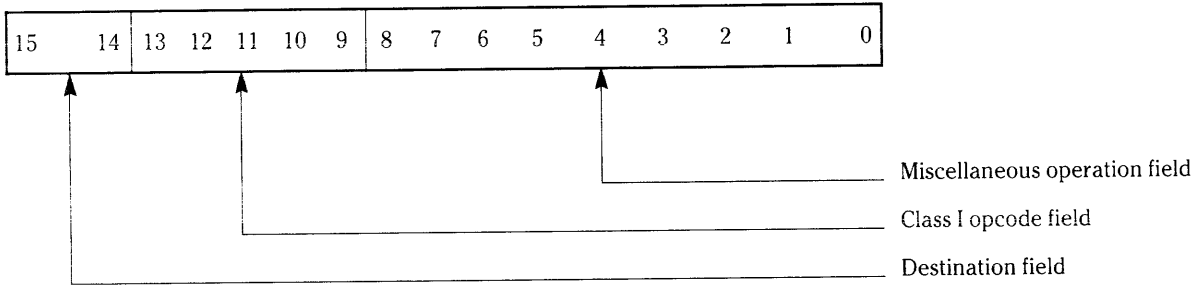
Figure 4-15   **Array Reference Immediate Instruction, Class V, Format**

Index field

Reference kind field

Accessed structure

Destination field
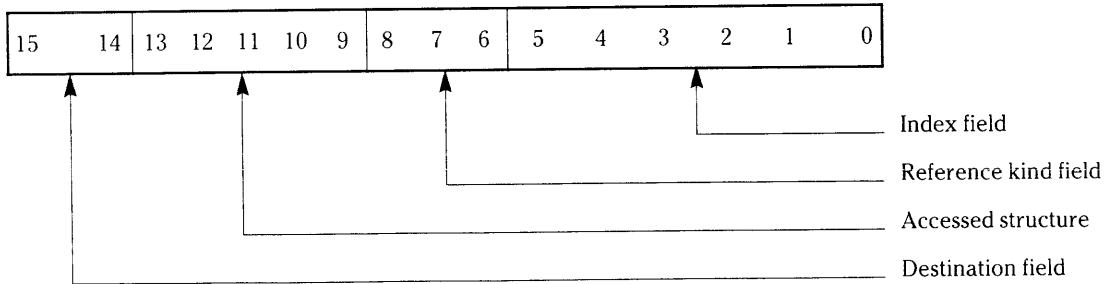
Table 4-15   **Reference Kind Decoding**

REFKIND	Instruction
0	AREFI-array-new
1	AREFI-array-leader-new
2	AREFI-inmstance-new
3	AREFI-array-new-common-lisp
4	AREFI-set-array-new
5	AREFI-set-array-leader-new
6	AREFI-set-instance-new
7	Illegal

Processor Microinstructions

4.5   The microinstructions in the processor are 56 bits wide, including the parity bit, and at the output of the microinstruction register are called IR(55:00). As described earlier, there are four basic microinstruction formats: the ALU, byte, jump, and dispatch formats. Each of these microinstruction formats are shown in Figure 4-16 and are described in the paragraphs that follow. As shown, the microinstructions are organized into functional fields. Some of the fields are common to all formats, while others are unique to one format.

Common Field Definitions

4.5.1   Fields that are common to all microinstruction formats are described in the following paragraphs.

Operation Field, IR(55:54)

4.5.1.1   Functions for operation field values (IR(55:54)) are shown as follows:

Field Value	Function
00	ALU operation
01	Byte operation
10	Jump operation
11	Dispatch operation

Figure 4-16   Explorer Processor Microinstruction Format
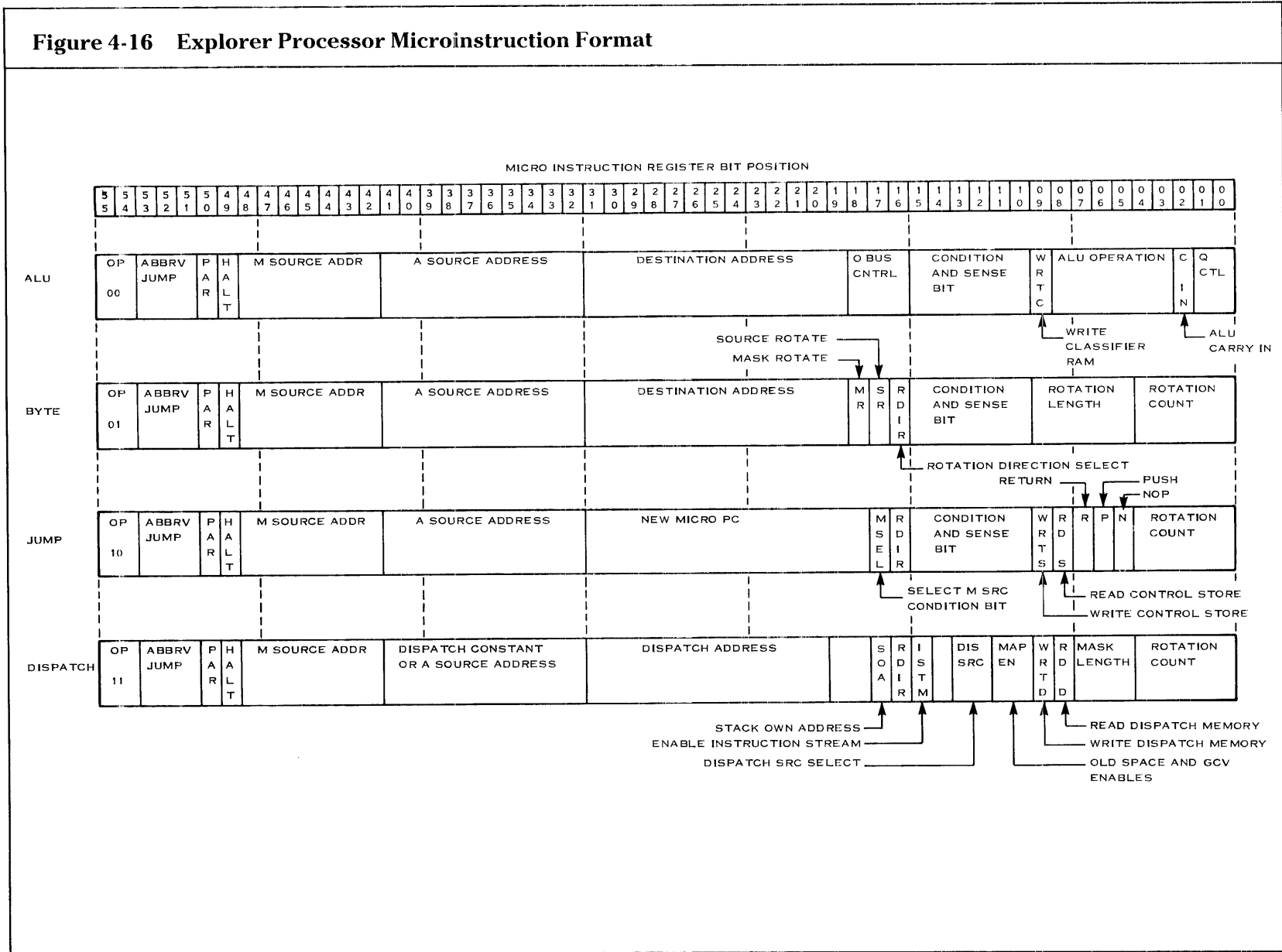
Abbreviated Jump Field, IR(53:51)　**4.5.1.2**　Functions for abbreviated jump field values (IR(53:51)) are shown as follows:

Field Value	Function
000	nil (unconditional)
001	Skip (conditional)
010	Call-to-ILLOP (conditional)
011	Call-to-trap (conditional)
100	Call-to-BUSERR (conditional)
101	Call—Currently unused (conditional)
110	POPJ (conditional return)
111	POPJ-XCT-next (conditional return)

The abbreviated jump operations allow a change of control in ALU and byte microinstructions only, having no effect in jump or dispatch microinstructions. In ALU and byte microinstructions, if the condition tested is true, skip the next microinstruction, or perform the call to ILLOP, trap, BUSERR, or unused micro-PC addresses.

POPJ causes a return from a microsubroutine, which inhibits execution of the next microinstruction. POPJ should not be set when the destination of a microinstruction is the μPCS. POPJ can be used in all microinstruction formats. POPJ is interpreted as POPJ-XCT-Next in jump and dispatch microinstructions.

POPJ-XCT-Next causes a return from a microsubroutine after executing the next instruction. This should not be set when the destination of a microinstruction is the μPCS. POPJ-XCT-Next can be used in all microinstruction formats.

Parity Field, IR(50)　**4.5.1.3**　IR(50) is an odd parity bit and is calculated over the control word. IR(50) can be disabled from causing an error PC trap by MCR(12).

Halt Field, IR(49)　**4.5.1.4**　The HALT bit, when set, causes the machine to halt in the next cycle. The instruction that has this bit set executes to completion, including all pipelined writes. The machine then stops execution.

M-Source Address Field, IR(48:42)　**4.5.1.5**　The first 64 addresses (IR(48) = 0) of the M-address field reference the M memory itself. The last 64 addresses (IR(48) = 1) reference sources on the M and MF buses (functional sources). (See paragraph 4.5.2, M Bus Sources.)

A-Source Address Field, IR(41:32)　**4.5.1.6**　The A-source operand is at the address specified in the A-source field.

Destination Address Field, IR(31:19)

4.5.1.7 The destination address field is common to the ALU and byte microinstruction formats. The destination addresses (Figure 4-17) are broken into parts as follows:

- The first half of the addresses (when IR(31) = 0) is used to specify both an M-memory location and one of 32 possible internal machine registers.

- The second half of the addresses (when IR(31) = 1) is used to specify an A memory destination.

- Bits IR(30:25) specify internal machine registers.

- Bits IR(24:19) specify the M scratchpad write addresses. Whenever the 64-location M memory is written to, the corresponding address in the first 64 locations of the A memory is also written to. Writing to one of the first 64 A memory locations does not cause a write operation to the M memory.

M Bus Sources

4.5.2 Table 4-16 lists the M bus sources, their addresses, and the data supplied to the M bus (some via the MF bus). The M source address field is common to all of the microinstruction formats.
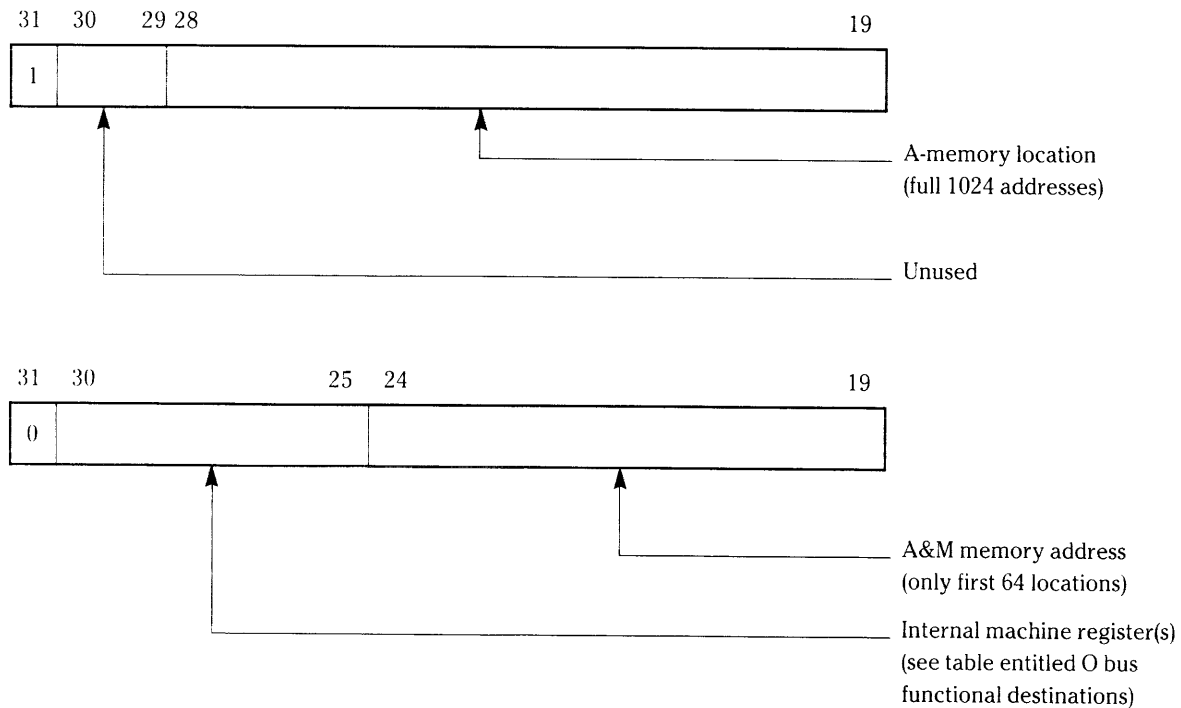
Figure 4-17 Destination Address Field

Table 4-16   **M Bus Sources, IR(48) = 1**

Bits IR(47:42)	Source
000000	VMA register M(31:00) = VMA(31:00)
000001	Q register M(31:00) = Q(31:00)
000010	IBUF argument offset field zero extended M(31:06) = 0 M(05:00) = IBUF(05:00) of current macroinstruction
000011	Stacked micro-PC pointer M(31:06) = 0 M(05:00) = μPCS pointer(05:00)
000100	MCR M(31:28) = ID(03:00) NuBus slot ID– (low active) M(27) = NuBus flag register self-test busy indicator (0 = incomplete or fail, 1 = complete) M(26) = Macroinstruction chaining enable M(25) = Enable MISCOP group 1 if group 0 is enabled M(24) = Enable MISCOP group 0 M(23) = Loop on self-test M(22) = Need fetch M(21) = NuBus reset M(20) = Power fail warning and warm-boot-event enable M(19) = Interrupt level (03) M(18) = Interrupt level (02) M(17) = Interrupt level (01) M(16) = Interrupt level (00) M(15) = Interrupt enable M(14) = Sequence break (tested by page fault, interrupt, or sequence break condition) M(13) = Abort on bus-error enable M(12) = Internal parity error halt enable M(11) = Microcode PROM disable M(10) = Interlocked memory cycle control M(09) = Forced access request M(08) = Memory cycle enable M(07) = (0 = failed, 1 = passed. Not valid if MCR(27 = 0) M(06) = NuBus flag register self-test fail indicator (0 = failed, 1 = passed. Also the fault LED control.) M(05) = LED5– M(04) = LED4– M(03) = LED3–

Table 4-16   **M Bus Sources, IR(48) = 1 (Continued)**

Bits IR(47:42)	Source

	M(02)	=	LED2–
	M(01)	=	LED1–
	M(00)	=	LED0–

000101   (Location counter)
M(31:26)   =   0
M(25:00)   =   location counter(25:00)

000110   Memory map data LVL2, address bits
M(31:22)   =   0
M(21:00)   =   Physical page number (page frame number)

000111   Dispatch constant register
M(31:10)   =   0
M(09:00)   =   Dispatch constant (09:00)

001000   Memory map data LVL1
M(31:16)   =   0
M(15)   =   Unmapped cycle
M(14)   =   Not(forced cycle)
M(13)   =   Privilege fault — write
M(12)   =   Privilege fault — access
M(11)   =   Map entry valid
M(10)   =   Reserved
M(09:07)   =   GC region volatility
               (M(09) is the GC valid bit.)
M(06:00)   =   LVL1 map index into LVL2 map

001001   Memory map data LVL2 — control bits
M(31:16)   =   0
M(15)   =   Last TM0
M(14)   =   Last TM1
M(13)   =   Last locked
M(12:11)   =   LVL2 GC most volatile point
M(10)   =   Forced access bit
M(09:08)   =   Read/write access rights
M(07:06)   =   Map status
M(05:00)   =   Meta bits (software controlled memory
               management)

001010   IBUF register
M(31:16)   =   0
M(15:00)   =   IBUF(15:00) of current macroinstruction

001011   IBUF branch offset field zero extended
M(31:09)   =   0
M(08:00)   =   IBUF(08:00) of current macroinstruction

Table 4-16   **M Bus Sources, IR(48) = 1 (Continued)**

Bits IR(47:42)	Source
001100 through 001111	Reserved
010000	Stacked micro-PC M(31:20) = 0 M(19:00) = μPCS(19:00)
010001	Stacked micro-PC data, pop stack. M(31:20) = 0 M(19:00) = μPCS(19:00)
010010	MD register M(31:00) = MD(31:00)
010011 through 011111	Reserved
100000	PDL buffer, addressed by pointer M(31:00) = PDL(31:00)
100001	PDL buffer, addressed by index M(31:00) = PDL(31:00)
100010	Unused, reserved for future expansion
100011	Unused, reserved for future expansion
100100	PDL buffer, addressed by pointer, pop M(31:00) = PDL(31:00)
100101	PDL buffer, addressed by index, decrement M(31:00) = PDL(31:00)
100110	Unused, reserved for future expansion
100111	Unused, reserved for future expansion
101000	PDL pointer(09:00) M(31:10) = 0
101001	PDL index(09:00) M(31:10) = 0
101010	Unused, reserved for future expansion
101011	Unused, reserved for future expansion

Table 4-16   **M Bus Sources, IR(48) = 1 (Continued)**

Bits IR(47:42)	Source
101100	PDL pointer, pop M(31:10) = 00 M(09:00) = PDL pointer(09:00)
101101	PDL index, decrement M(31:10) = 0 M(09:00) = PDL index(09:00)
101110	Unused, reserved for future expansion
101111	Unused, reserved for future expansion
110000 through 111111	Unused

Selectable O Bus Destinations

4.5.3   Table 4-17 lists the selectable destinations available from the O bus. The table includes the address and the data bit assignments of the destination.

Rotation Count and Direction Fields

4.5.4   The rotation count and direction fields are common to byte, jump, and dispatch microinstruction formats. The field is described in Table 4-18.

Condition and Sense Bit Field

4.5.5   The condition and sense bit field (IR(15:10)) is common to ALU, byte, and jump microinstruction formats. Bit IR(15) is the jump condition sense bit. When IR(15) = 1, invert the jump condition sense. Functions of condition and sense bits IR(14:10) are shown in Table 4-19. In ALU and byte instructions, the conditions that can be tested are those conditions accessible with the condition specifier. In the jump microinstruction, the M source select (MSEL) (IR(17)) bit allows any bit of any M source to be tested in place of any of the conditions accessible with the condition specifier.

ALU Format

4.5.6   The ALU instruction is used to perform arithmetic and logical operations in the processor. The instruction performs operations on the two source variables and writes the result at the destination address. The carry-in bit to the ALU is directly under microcode control. The ALU microinstruction format also allows for control of the O bus multiplexer and control of the operation of the Q register. Table 4-20 shows the ALU microinstruction format.

Table 4-17   **O Bus Functional Destinations, IR(31) = 0**

Bits IR(30:25)	*Destination*
000000	None
000001	Location counter(25:00) = O bus(25:00)

000010	MCR
	  Unused (O bus(31:28))
	  NuBus flag register self-test busy indicator
	    (O bus(27), 0 = incomplete or fail, 1 = complete)
	  Macroinstruction chaining enable (O bus(26))
	  Enable MISCOP group 1 if group 0 is enabled (O bus(25))
	  Enable MISCOP group 0 (O bus(24))
	    Unused (O bus(23:22))
	  NuBus reset — active high (O bus(21))
	  Unused (O bus(20))
	  Unused (O bus(19:16))
	  Interrupt enable (O bus(15))
	  Sequence break request (O bus(14))
	  Trap on bus-error enable (O bus(13))
	  Internal parity-error-abort enable (O bus(12))
	  Microcode PROM disable (O bus(11))
	  Interlocked memory cycle control (O bus(10))
	  Forced access request (O bus(09))
	  Memory cycle enable (O bus(08))
	  NuBus flag register subsystem fail indicator
	    (O bus(07), 0 = failed, 1 = passed)
	  NuBus flag register self-test fail indicator
	    (O bus(06), 0 = failed, 1 = passed. Also fault LED control.)
	  Fault LEDs (O bus(05:00))
	    (0 = turn on LED)

000011	μPCS pointer μPCS pointer(05:00) = O bus(05:00)
000100	μPCS data μPCS(19:00) = O bus(19:00)
000101	μPCS data, push μPCS(19:00) = μPCSL(19:00)

000110	Next IR modifier — IMOD low
	  (IMOD register (31:00))
	  NXTIR(31:00) = O bus(31:00) OR I(31:00)

000111	Next IR modifier — IMOD high
	  (IMOD register (55:32))
	  NXTIR(55:32) = O bus(23:00) OR I(55:32)

001000	IBUF register IBUF(31:00) = O bus(31:00)

Table 4-17   **O Bus Functional Destinations, IR(31) = 0 (Continued)**

Bits IR(30:25)	Destination
001001 through 001110	Unused
001111	Test sync signal destination
010000	VMA register VMA(31:00) = O bus(31:00)
010001	VMA register, write map LVL1 — The map is addressed from the MD register and written from the VMA register (11:00).
010010	VMA register, write map LVL2 control — The map is addressed from MD and LVL1 and written from VMA(12:00).
010011	VMA register, write map LVL2 address — The map is addressed from MD and LVL1 and written from VMA(21:00).
010100	VMA register, start memory read VMA(31:00) = O bus(31:00)
010101	VMA register, start memory write VMA(31:00) = O bus(31:00)
010110	VMA register, start memory read unmapped VMA(31:00) = O bus(31:00)
010111	VMA register, start memory write unmapped VMA(31:00) = O bus(31:00)
011000	MD register MD(31:00) = O bus(31:00)
011001	MD register, write map LVL1 — See 010001
011010	MD register, write map LVL2 control — See 010010
011011	MD register, write map LVL2 address — See 010011
011100	MD register, start memory read MD(31:00) = O bus(31:00)
011101	MD register, start memory write MD(31:00) = O bus(31:00)
011110	MD register, start memory read unmapped MD(31:00) = O bus(31:00)

Table 4-17

O Bus Functional Destinations, IR(31) = 0 (Continued)

Bits IR(30:25)	Destination
011111	MD register, start memory write unmapped MD(31:00) = O bus(31:00)
100000	PDL buffer, addressed by pointer PDL(31:00) = O bus(31:00)
100001	PDL buffer, addressed by index PDL(31:00) = O bus(31:00)
100010	Reserved
100011	Reserved
100100	PDL buffer, addressed by pointer, push PDL(31:00) = O bus(31:00)
100101	PDL buffer, addressed by index, increment PDL(31:00) = O bus(31:00)
100110	Reserved
100111	Reserved
101000	PDL pointer(09:00) = O bus(09:00)
101001	PDL index(09:00) = O bus(09:00)
101010	Reserved
101011	Reserved
101100	Reserved
101101	Reserved
101110	Reserved
101111	Reserved
110000 through 110101	Unused
110110	VMA register, start memory read unmapped VMA(31:00) = O bus(31:00) (If NuBus access, then TM0 = 1.)

Table 4-17	**O Bus Functional Destinations, IR(31) = 0 (Continued)**

Bits *IR(30:25)*	*Destination*
110111	VMA register, start memory write unmapped VMA(31:00) = O bus(31:00) (If NuBus access, then TM0 = 1.)
111000	Reserved
111101	Reserved
111110	MD register, start memory read unmapped MD(31:00) = O bus(31:00) (If NuBus access, then TM0 = 1.)
111111	MD register, start memory write unmapped MD(31:00) = O bus(31:00) (If NuBus access, then TM0 = 1.)

NOTE:

For the unmapped memory accesses specified in this table, the two LSBs in the address are supplied by the VMA. Hardware associated with the mapper detects a bit combination that can request a NuBus block transfer and override A(01) to force the request to a single-word transfer.

Table 4-18	**Rotation Count and Direction Fields**

Bits	*Function*
IR(16)	Rotation direction select If IR(16) is zero, the rotation direction is left. If IR(16) is one, the rotation direction is right.
IR(04:00)	Rotation count to rotate mask and/or M source

Table 4-19   **Condition and Sense Bit Field**

Bits IR(14:10)	*Function*
00000	LSB of shifter output R(00) (normally used only for M sources)
00001	M source less than A source (ALU negative)
00010	ALU(32)
00011	Not (M source equals A source)
00100	Page fault
00101	Page fault or interrupt pending
00110	Page fault or interrupt pending or sequence break
00111	Unconditionally true
01000	Not (A-TYPE equals M-TYPE)
01001	Not (memory busy)
01010	Q(0)
01011	Bus error on last transfer attempt
01100	Typed-data overflow
01101	Boxed sign bit (ALU(24))
01110	Not (interrupt pending)
01111 through 10000	Reserved, do not use
11111	M-type classifier RAM

Table 4-20   **ALU Microinstruction Format**

Bits	Function
IR(55:54)	Opcode = 00 = ALU instruction
IR(53:51)	Abbreviated jump field
IR(50)	Parity
IR(49)	Halt
IR(48:42)	M source
IR(41:32)	A source
IR(31:19)	Destination
IR(18:16)	O bus control

O bus control:

```
000 — A bus
001 — R bus
010 — A bus
011 — ALU output
100 — ALU output left shift
        O bus(31:01) = ALU(30:00)
        O bus(00) = Q(31)
101 — ALU output right shift
        O bus(31) = sign
        O bus(30:00) = ALU(31:01)
110 — ALU pointer field sign extended
        O bus(31:24) = ALU(24), O bus(23:00)
111 — ALU mirror
```

Bits	Function
IR(15:10)	Jump condition specifier, sense bit
IR(09)	Write M-type classifier RAM
IR(08:03)	ALU operation (Fixnum overflow is valid in any ALU add or subtract operation whether IR(08) is set or not.)
IR(02)	Carry into ALU
IR(01:00)	Q-register control

Q-register control:

```
00 — Do nothing
01 — Shift left, shift in the inverse of the sign of the ALU
        output
        Q(31:01) = Q(30:00)
        Q(00) = ALU(31)–
10 — Shift right, shift in ALU(00)
        Q(30:00) = Q(31:01)
        Q(31) = ALU(00)
11 — Load Q from ALU output
        Q(31:00) = ALU(31:00)
```

The most significant bit (MSB) of the ALU operation field, IR(08), is used to select typed-data ALU operations. In typed-data ALU operations, bits ALU(24:00) are passed to the O bus, and bits A(31:25) are substituted for ALU(31:25). The typed-data select bit affects only the result of the operation and not the data into the ALU. This means that the ALU flags are based on the 32-bit result of the operation and not just the 25-bit typed-data field. All of the ALU logic operations and all of the normal ALU arithmetic operations can be used with the typed-data bit selected. ALU operations can also be used with the typed-data bit set. The multiply and divide ALU operations operate correctly only with the typed-data bit reset.

Selecting the typed-data bit requires that the O bus control field of the ALU microinstruction be set to ALU (011 binary). If the typed-data bit is set, only the ALU and the A input to the O bus multiplexer will participate in generating the data to be placed on the O bus.

The following information applies during ALU instructions with IR(08) equal 1.

When the O bus control field has the LSB set to zero, the selected input to the O bus multiplexer is passed unaltered to the O bus. Selecting the O bus control field to be 001 causes the contents of the R bus to be placed in the 25 LSBs of the O bus with the 7 MSBs being supplied by the A bus. Thus, O bus(31:00) equals A(31:25)::R(24:00).

Selecting the O bus control field to be 7 (111) causes the seven MSBs of the sign-extended ALU result to be placed on the O bus along with the 25 LSBs of the mirrored ALU output. Thus, O bus(31:00) equals ALU(24)(in the seven MSBs)::ALU(08:31).

Therefore if IR(08) equals 1, then IR(16) must also equal 1.

Byte Format **4.5.7** The byte instruction allows a variable-length field from the M source to be inserted into an equal length field of the A source. The byte instruction forces the ALU operation to a subtract mode so that the ALU-related condition flags are meaningful. Table 4-21 describes the byte microinstruction format.

Table 4-21   **Byte Microinstruction Format**

Bits	Function
IR(55:54)	Opcode = 01 = byte instruction
IR(53:51)	Abbreviated jump field
IR(50)	Parity
IR(49)	Halt
IR(48:42)	M source
IR(41:32)	A source
IR(31:19)	Destination
IR(18)	Mask rotate 0 — Rotate mask 0 bits 1 — Rotate mask left by number of bits specified in the rotate field
IR(17)	Source rotate 0 — Rotate M source by 0 bits 1 — Rotate M source left by number of bits specified in the rotate field
IR(16)	Rotation direction, 1 = right
IR(15:10)	Jump condition, sense
IR(09:05)	Length of field from M source
IR(04:00)	Rotation count

The following list explains the results of using all of the combinations of IR(18:17) (the mask and source rotate enables).

■ 00 — This is a subset of the other operations and is not considered useful. Only the LSB of the M bus is selected.

■ 01 — A field of arbitrary position from the M source is right justified in the output. Figure 4-18 shows the load byte operation.

■ 10 — A masked field from the M source is used to replace the same length field and position field in a word from the A source. Figure 4-19 shows the selective deposit byte operation.

■ 11 — A right-justified field from the M source is used to replace a field of arbitrary position in the word from the A source. Figure 4-20 shows the deposit byte operation.

Figure 4-18   **Load Byte Microinstruction**

A-SOURCE WITH M-SOURCE FIELD INSERTED
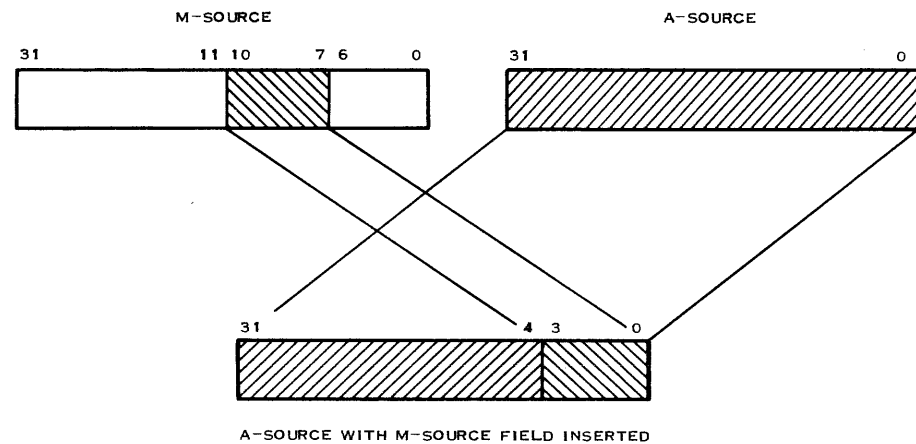
Figure 4-19   **Selective Deposit Byte Microinstruction**

A-SOURCE WITH M-SOURCE FIELD INSERTED
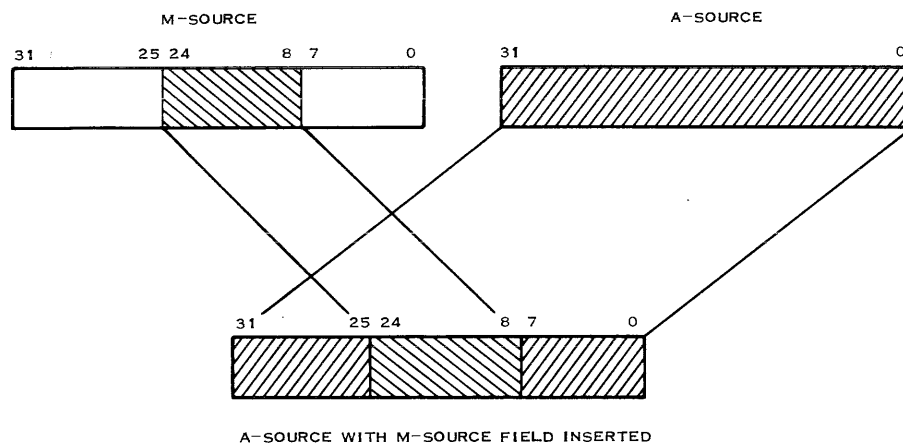
Figure 4-20   **Deposit Byte Microinstruction**

A-SOURCE WITH M-SOURCE FIELD INSERTED
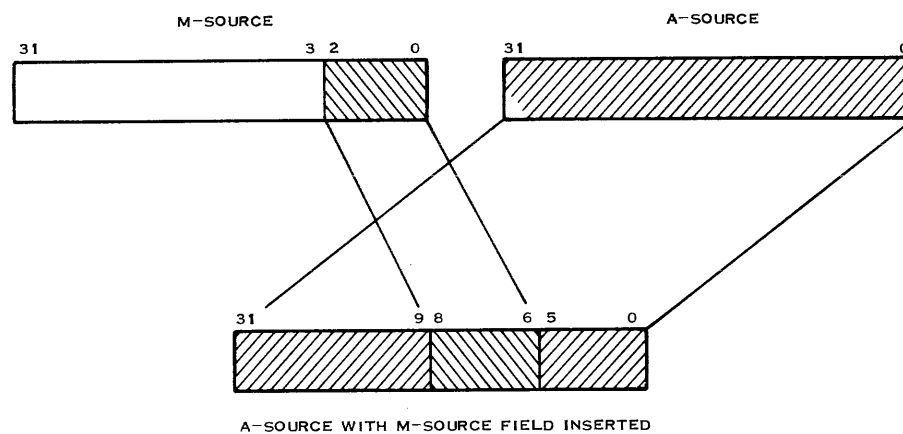
Jump Format   **4.5.8**   The jump microinstruction specifies a new 14-bit micro-PC and three transfer type bits. The micro-PC is specified in IR(31:18). The three transfer type bits are specified with the R (IR(07)), P (IR(06)), and N (IR(05)) bits. The jump instruction forces the ALU operation to a subtract mode so that the ALU related test condition flags are meaningful. The format of the jump microinstruction type is shown in Table 4-22.

Table 4-22   **Jump Microinstruction Format**

Bits	Function
IR(55:54)	Opcode = 10 = jump instruction
IR(53:51)	Abbreviated jump field (Must be 000, 110, or 111)
IR(50)	Parity
IR(49)	Halt
IR(48:42)	M Source
IR(41:32)	A Source
IR(31:18)	New PC
IR(17)	MSEL, select M source for jump test bit
IR(16)	Rotation direction (1 = right)
IR(15:10)	Jump condition, sense
IR(09)	When set, causes a control store write operation
IR(08)	When set, causes a control store read operation
IR(07)	R bit (1 = pop new PC off μPCS)
IR(06)	P bit (1 = push return PC on μPCS)
IR(05)	N bit (1 = inhibit execution of next instruction if jump is successful)
IR(04:00)	Rotation count

The three transfer type bits are individually defined as follows:

■ N Bit — When N equals 1, the execution of the next microinstruction loaded into the microinstruction register is inhibited (NOPed). No registers or memories that are destinations of a microinstruction are written to. No stack pointers are modified.

■ P Bit — When P equals 1, the return micro-PC is pushed onto the μPCS.

■ R Bit — When R equals 1, the return micro-PC is popped off of the μPCS.

The R, P, and N bits taken together define the operations as shown in Table 4-23.

Table 4-23   **Transfer of Control Bits**

RPN	Operation
000	Branch to the specified location after executing the microinstruction following the branch. PC(13:00) = IR(31:18)
001	Branch to the specified location after one additional clock cycle. The microinstruction following the branch is not executed. PC(13:00) = IR(31:18)
010	Call the routine at the specified address after executing the microinstruction following the call. PC(13:00) = IR(31:18) Top-of-μPCS = call-micro-PC-plus-2
011	Call the routine at the specified address after one additional clock cycle. The microinstruction following the call is not executed. PC(13:00) = IR(31:18) Top-of-μPCS = call-micro-PC-plus-1
100	Return from the subroutine after executing the microinstruction following the return. PC(13:00) = μPCS(13:00)
101	Return from the subroutine after one additional clock cycle. The microinstruction following the return is not executed. PC(13:00) = μPCS(13:00)
110	Same as 000.
111	Same as 001.

A sequence of two jump instructions is used to write to or read from the control-store memory. The first set of two microinstructions that perform a control-store memory write operation are as follows: (Data is supplied by the A and M memories.)

Write to control store memory:
    (CALL-EXECUTE-NEXT (CONTROL-STORE-TARGET-ADDRESS))
    RETURN-INHIBIT-EXECUTION-NEXT JUMP-ADDRESS-DONT-CARE
        A-CONTROL-STORE-LOC M-CONTROL-STORE-LOC WRITE-IRAM)

The second set of two microinstructions performs a control store read operation and saves a portion of the data in the address specified in the new PC field. The data in this field is interpreted as a destination address in this situation.

Read instruction memory bits (55:32):
    (CALL-EXECUTE-NEXT (CONTROL-STORE-TARGET-ADDRESS))
    (RETURN-INHIBIT-EXECUTION-NEXT
        A-CONTROL-STORE-SAVE-IN-THE-NEW-PC-FIELD
        A-DONT-CARE M-DONT-CARE READ-IRAM)

A third pair of instructions saves the remaining portion of the control-store data in the M memory.

Read instruction memory bits (31:00):
    (CALL-EXECUTE-NEXT(CONTROL-STORE-TARGET-ADDRESS))
    (RETURN-INHIBIT-EXECUTION-NEXT
        M-CONTROL-STORE-SAVE-IN-THE-NEW-PC-FIELD A-DONT-CARE
        M-DONT-CARE READ-IRAM)

When performing a control store read, IR(31:19) contains the destination address. IR(18:16) must be 011 binary. When reading and writing control store memory, A(23:00) maps to I(55:32) and M(31:00) maps to I(31:00).

These same operations are specified to the microassembler in symbolic form:

Write to control store memory:
    (Access-I-Mem < I memory label>)   Specify I-memory-location.
    (Write-I-Mem < A source> < M source>)   Perform the write.

Read instruction memory bits (55:32)
    (Access-I-Mem < I memory label>)   Specify I-memory location.
    (Read-I-Mem < A memory destination>)

Read instruction memory bits (31:00):
    (Access-I-Mem < I memory label>)   Specify I-memory location.
    (Read-I-Mem < M-memory destination>)

Dispatch Format   **4.5.9**   The dispatch microinstruction allows a multiway transfer of control. This transfer is based on the address produced by logically ORing the following:

■ The seven LSBs of the dispatch address field

■ The seven LSBs from the selected dispatch address source

If the selected dispatch address source is the R bus, the MSBs of the dispatch address are taken from the dispatch address field of the microinstruction.

If the selected address is from the MF bus, only five bits (MF(29:25)) from the bus are available to be selectively ORed with the dispatch address field of the microinstruction. These bits are fed into dispatch address positions (05:01) where they are inclusively ORed with bits (25:21) of IR. Bit position (00) is set equal to IR(20). The LSB can become the inclusive OR of one of the following:

■ IR(20)

■ A bit from the LVL1 map (the old space bit)

■ The GC volatility fault bit

This form of dispatch is called the transport dispatch.

If the selected address is from the IBUF register, only the seven low-order bits of the 10-bit field from IBUF can be selectively ORed with the dispatch-address field of the microinstruction register. The three next MSBs of IBUF replace the bits from the IR field. The next MSB of the dispatch address is generated by ORing the corresponding bit in the IR and the status of the MISCOP decode. The MSB of the dispatch address is generated by the corresponding bit in the IR. A dispatch based on the contents of IBUF is called an instruction-decode-dispatch. Note that MISCOP detection can be disabled under the control of two bits in the MCR.

If the MSB of the dispatch address source select field (IR(13)) is 1, then the most significant address bit into the dispatch memory is forced to 1. If the MSB of the dispatch address source select field is set to 0, the address selected is generated from either the R bus or the MF bus.

The dispatch address field is 12 bits long and is found in IR(31:20). After the OR takes place, the address produced is an index into the dispatch memory that contains a 14-bit micro-PC and 3 transfer type bits. The 3 bits are the same R, P, and N bits described in the jump microinstruction. The 3 transfer type bits describe the type of transfer to take place and use the 14-bit micro-PC if required. The O bus multiplexer is forced to select the A bus as the source of data for all dispatch operations. Table 4-24 shows the dispatch microinstruction format.

Table 4-24   **Dispatch Microinstruction Format**

Bits	Function
IR(55:54)	Opcode = 11 = dispatch instruction
IR(53:51)	Abbreviated jump field (Must be 000, 110, or 111, must be 000 if writing dispatch memory)
IR(50)	Parity (odd)
IR(49)	Halt
IR(48:42)	M source
IR(41:32)	Dispatch constant (also the A memory source when writing to the dispatch memory)
IR(31:20)	Dispatch address
IR(19)	Unused
IR(18)	Unused
IR(17)	Stack-own-address alters the return address pushed on the μPCS by the call transfer type. If the N bit is set, the address of this instruction should be stacked rather the next instruction.
IR(16)	Rotation direction (1 = right)
IR(15)	Enable instruction stream hardware
IR(14)	Unused
IR(13:12)	Dispatch address source 00 — R(06:00) 01 — MF(29:25):0 1x — IBUF(09:00) or IBUF(15:06) — Auto selected by the macroinstruction opcode if MISCOP decoding is enabled.
IR(11)	Old space enable — When IR(11) is set, the LSB of the dispatch address is set to 1 when the old space bit (LVL1(10)) is set to 1.
IR(10)	GC volatility enable bit — When IR(10) is set, the LSB of the dispatch address is set to 1 if the GC volatility bit is 1.
IR(09)	Write dispatch memory
IR(08)	Read dispatch memory
IR(07:05)	Length of field from the selected M source (0 to 7 bits) to logically OR into the dispatch address of the microinstruction register
IR(04:00)	Rotation count

The operation of the R, P, and N bits and the POPJ function in the dispatch instruction are identical to that in the jump instruction. With both R and P set to one, the dispatch operation is ignored and the execution of the next instruction is based on the state of the N bit. Refer to Table 4-23, Transfer of Control Bits.

The dispatch format (if IR(11) = 1) also allows the next to MSB of the LVL1 map (the old space bit) to be inclusively ORed with the LSB of the dispatch address field from the instruction register (IR(20)). The GC volatility fault bit can also be selected to be inclusively ORed into the LSB of the dispatch address (if IR(10) = 1). If both IR(11) and IR(10) are set, then the LSB of the dispatch address is the inclusive OR of the old space bit, the GC volatility fault bit, and IR(20).

LIST OF ACRONYMS

AL	A-memory, last-result (data register)
ALU	arithmetic logic unit
AREFI	array reference immediate
ARGS	arguments
ASCII	American Standard Code for Information Interchange
BR	branch (instructions)
CDR	contents of decrement portion of register
DIN	Deutsches Institut fuer Normung (German Institute for Standardization)
FEF	functions entry frame
GC	garbage collector
IBUF	macroinstruction buffer
ILLOP	illegal operation
IMOD	instruction modify
IOR	immediate OR
IR	instruction register
IRAM	instruction RAM
IVAR	instance variable
LAN	local area network
LED	light-emitting diode
Lisp	list processing (a computer language)
L	last-result (register)
LSB	least significant bit or byte
LVL	level
MCR	machine control register
MD	memory data (register)
MISCOP	miscellaneous operations
ML	M-memory, last-result (data register)
MSB	most significant bit or byte
MSEL	M-source select
NDOP	nondestination operation
NOP	no operation
NVRAM	nonvolatile RAM
NXTIR	next instruction register
O bus	output bus

PC	program counter
PDL	push-down list
POPJ	pop jump
PROM	programmable, read-only memory
RAM	random-access memory
ROM	read-only memory
TM0	transfer mode zero
TM1	transfer mode one
μ**PCS**	microprogram counter stack
Vcc	collector voltage
VMA	virtual memory address

GLOSSARY

A

acknowledge cycle The last period of a transaction (one clock period long) during which the ACK– signal is asserted.

address A hexadecimal number indicating memory slot space, a slot ID number, and a data address. An example is F(S)FFFFFF.

address cycle The first period of a transaction (one clock period long) during which the START– signal is asserted. The Address cycle is the same as the start cycle.

address space The space set aside in memory for a specific group of address numbers.

arbitration To select between the different circuit boards that are available on a bus using a priority system.

asserted This expression indicates logic low or true.

assertion edge The rising edge (low to high) of the central system clock.

B

backplane A circuit board that connects all the circuit board slot connectors in a chassis together to make a bus connection.

block transfer The movement of groups of consecutive 32-bit words over the NuBus.

bus A group of one or more signal lines that are used to transfer information from one or more sources to one or more destinations.

byte A group of eight parallel bits of data.

C

chip An integrated circuit device containing a large number of electronic elements in a single package.

cycle A periodically repeated sequence of operations.

E

ejector
A mechanical device on a circuit board that provides leverage to help remove a circuit board from the connectors on the backplane.

H

hardwired
This describes wires that are permanently connected to terminals.

halfword
A group of 16 parallel bits of data.

I

injector
A mechanical device on a circuit board that provides leverage to help insert a circuit board into the connectors on the backplane.

J

jumper
A connection between two or more terminals.

L

Lisp
A high-level computer programming language used for artificial intelligence.

local bus
A bus that is related to a particular group of circuit boards that operate under the same signal protocols.

M

master
The controlling circuit in a master/slave communication protocol over a bus.

N

NuBus   A high-speed synchronous bus that multiplexes 32-bit data words with 32-bit address codes and uses master/slave communication protocols.

P

period   One clock cycle of either the NuBus clock or the local bus clock.

S

sample edge   Falling edge (logic high to low) of the central system clock.

slave   The device that is controlled by a master device in a master/slave communication arrangement over a bus.

slot   A physical channel in a chassis where a circuit board is inserted.

slot ID   A binary code that identifies the slot into which a circuit board is inserted.

static-sensitive   This refers to devices that can be damaged by static electricity.

synchronous bus   A bus that has a separate clock line to provide synchronization for data that is transferred over the bus.

T

transaction   A completed bus operation; for example, a read or write operation.

U

unasserted   A term that is synonymous with logic high or false.

W

word   A group of 32 parallel bits of data.

USER RESPONSE SHEET

Our manuals are written for you. Please help us improve them by completing and mailing this form. List any discrepancy or other problem that you found in this manual by page, paragraph, figure, or table number in the space provided, and add any suggestions you have. Thank you.

Manual Title: _____

Manual Date: _____ **Today's Date:** _____

Your Name: _____ **Telephone:** _____

Company: _____ **Department:** _____

Company Address: _____

Product Purchased From:

Company	City	State

Location in Manual   **Comment/Suggestion**

_____   _____

                        _____

                        _____

_____   _____

                        _____

                        _____

_____   _____

                        _____

                        _____

The manual's good points: _____

                        _____

What needs to be improved: _____

                        _____

No postage necessary if mailed in U.S.A.
Fold on two lines (located on reverse side), tape and mail

TAPE EDGE TO SEAL

FOLD

BUSINESS REPLY MAIL
FIRST-CLASS PERMIT NO. 7284 DALLAS, TX

POSTAGE WILL BE PAID BY ADDRESSEE

TEXAS INSTRUMENTS INCORPORATED
DATA SYSTEMS GROUP

ATTN: PUBLISHING CENTER
P.O. Box 2909 M/S 2146
Austin, Texas 78769-9990

FOLD