HEWLETT PACKARD

HP 12025A/B Input/Output Extender

Hardware Support Manual

HP 1000 A-Series

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

HEWLETT PACKARD

Computer Museum

HP 12025A/B Input/Output Extender

Hardware Support Manual

Options Covered

This manual also covers Options 001, 002, and 015 of the HP 12025A/B I/O Extender.

Data Systems Division
11000 Wolfe Road
Cupertino, California 95014

Part No. 12025-90001
E0385

Printed in U.S.A. March 85

Federal Communications Commission Radio Frequency Interference Statement

The Federal Communications Commission (in Subpart J, of Part 15, Docket 20780) has specified that the following notice be brought to the attention of the users of this product.

WARNING: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for Class A computing devices pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company.

Copyright © 1985 by HEWLETT-PACKARD COMPANY

Printing History

The Printing History below identifies the Edition of this Manual and any Updates that are included. Periodically, Update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this Printing History page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past Updates, however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all Updates.

To determine what manual edition and update is compatible with your current software revision code, refer to the appropriate Software Numbering Catalog, Software Product Catalog, or Diagnostic Configurator Manual.

First Edition . . . . Mar 1985. . . . .

Safety Considerations

GENERAL - This product and related documentation must be reviewed for familiarization with safety markings and instructions before operation.

SAFETY SYMBOLS

⚠	Instruction manual symbol: the product will be marked with this symbol when it is necessary for the user to refer to the instruction manual in order to protect the product against damage.
⚡	Indicates hazardous voltages.
⏚	Indicates earth (ground) terminal (sometimes used in manual to indicate circuit common connected to grounded chassis).
WARNING	**The WARNING sign denotes a hazard. It calls attention to a procedure, practice, or the like, which, if not correctly performed or adhered to, could result in injury. Do not proceed beyond a WARNING sign until the indicated conditions are fully understood and met.**
CAUTION	The CAUTION sign denotes a hazard. It calls attention to an operating procedure, practice, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product. Do not proceed beyond a CAUTION sign until the indicated conditions are fully understood and met.

CAUTION

STATIC SENSITIVE DEVICES

Some of the semiconductor devices used in this equipment are susceptible to damage by static discharge. Depending on the magnitude of the charge, device substrates can be punctured or destroyed by contact or mere proximity to a static charge. These charges are generated in numerous ways such as simple contact, separation of materials, and normal motions of persons working with static sensitive devices.

When handling or servicing equipment containing static sensitive devices, adequate precautions must be taken to prevent device damage or destruction. Only those who are thoroughly familiar with industry accepted techniques for handling static sensitive devices should attempt to service the cards with these devices. In all instances, measures must be taken to prevent static charge buildup on work surfaces and persons handling the devices. Cautions are included through this manual where handling and maintenance involve static sensitive devices.

SAFETY EARTH GROUND - This is a safety class I product and is provided with a protective earthing terminal. An uninterruptible safety earth ground must be provided from the main power source to the product input wiring terminals, power cord, or supplied power cord set. Whenever it is likely that the protection has been impaired, the product must be made inoperative and be secured against any unintended operation.

BEFORE APPLYING POWER - Verify that the product is configured to match the available main power source per the input power configuration instructions provided in this manual.

If this product is to be energized via an auto-transformer (for voltage reduction) make sure the common terminal is connected to the earth terminal of the main power source.

SERVICING WARNING

Any servicing, adjustment, maintenance, or repair of this product must be performed only by qualified personnel.

Adjustments described in this manual may be performed with power supplied to the product while protective covers are removed. Energy available at many points may, if contacted, result in personal injury.

Capacitors inside this product may still be charged even when disconnected from its power source.

To avoid a fire hazard, only fuses with the required current rating and of the specified type (normal blow, time delay, etc.) are to be used for replacement.

WARNING

EYE HAZARD

Eye protection must be worn when removing or inserting integrated circuits held in place with retaining clips.

Table of Contents

Chapter 6
Removal and Replacement

Illustrations

Tables

Chapter 1
Product Information

Introduction

This chapter provides a general description of the HP 12025A/B Input/Output Extender which is used with HP 1000 A600+, A700, and A900 Computers.

NOTE

Hewlett-Packard has not tested the 12025A/B I/O Extender with the A600 computer, and use of the extender with the A600 computer is not supported by HP.

Product Description

The HP 12025A I/O Extender adds 12 I/O slots to an HP 1000 A-Series Computer System, and the HP 12025B I/O Extender adds 17 or 18 I/O slots, depending on configuration. More than one extender can be used to provide up to 48 I/O channels in a system. (HP RTE-A revision 2440 or later is required for RTE-A support of 25 to 48 I/O channels.)

The HP 12025A/B (Figure 1-1) consists of the following items:

a. 12025A only: A 16-slot box (part no. 02430-60014) with power supply.

b. 12025B only: A 20-slot box (part no. 12151-60001) with power supply.

c. I/O Control (IOC) card, part no. 12025-60001.

d. Extender Control (EXT) card, part no. 12025-60002.

e. Interrupt Chain Jumper (ICJ/12) card, part no. 12025-60003 (12025A).

f. Interrupt Chain Jumper (ICJ/18) card, part no. 12025-60004 (12025B).

g. A 2.5m (8.2 ft) CPU-to-extender connecting cable, part no. 12025-60007.

h. A 2.28m (7.5 ft) power cord, part no. 8120-1378.

i. 12025A/B I/O Extender Hardware Support Manual, part no. 12025-90001.

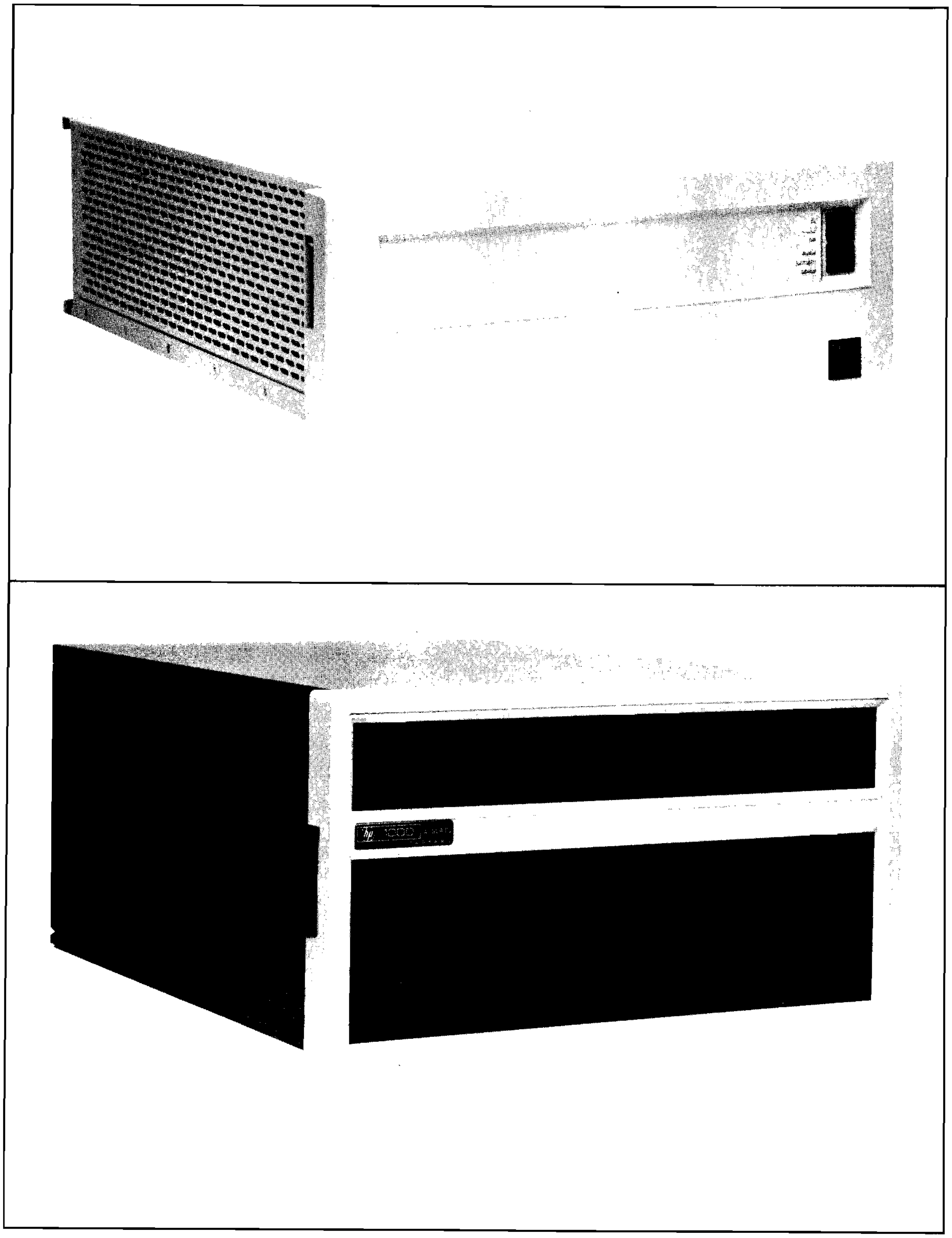


Figure 1-1. HP 12025A and 12025B I/O Extenders

The IOC printed circuit card must be installed in the host computer card cage and the EXT card in the extender card cage; the cable connects these cards together. The ICJ card is only required when the IOC card in the CPU card cage is not in the lowest I/O-priority slot used. When more than one extender is used, there is no connection between the extenders and all I/O-priority arbitration occurs on the CPU backplane. A single computer can host both 12025A and 12025B I/O Extenders at the same time.

Operation of the I/O extender is transparent to the operating system and to applications programs. Programming requirements for I/O cards that are installed in the extender are the same as for I/O cards installed in the host computer.

Options

The 12025A/B I/O Extender is available with the following options:

a. Option 001 provides an upgrade card for an A900 computer having a serial number prefix less than 2500, and enables the computer to be used with the extender. The option consists of a new HP 12203A Cache Card. Chapter 3 includes card installation instructions.

b. Option 002 provides an upgrade kit for an A700 computer having a serial number prefix less than 2500. The kit enables the computer to be used with the extender. The upgrade kit consists of some new firmware for the Lower Processor card. Chapter 3 includes kit installation instructions.

c. Option 015 enables the extender to operate from a single-phase 230-volt AC power source.

Accessories

The accessories for the I/O extender are as follows:

a. For the 12025A, the HP 12159A 25 kHz Power Module enables A-Series measurement and control cards to operate in the extender.

b. For the 12025A, the HP 40025A Vertical Floor Mount provides convenient roll-about mobility.

c. For the 12025B, the HP 12158A 25 kHz Sine Wave Module enables A-Series measurement and control cards to operate in the extender.

Specifications

The functional specifications for the 12025A/B I/O Extender are listed in Table 1-1 and the site preparation specifications are listed in Table 2-1.

Identification

The I/O extender boxes are identified by the product number (12025A/B) on a decal affixed to the rear of the box. The decal also shows a 10-digit serial number, for example, 2445A-00123.

The I/O extender printed circuit cards are identified by a part number marked on the cards. In addition, a letter and a 4-digit date code (e.g., A-2449) are placed below the part number. The letter identifies the version of the etched circuit on the card; the date code identifies the electrical characteristics of the assembled card.

Supported Equipment

The HP 12025A/B I/O Extender operates with the following A-Series computers:

a. All HP A600+ computers.

b. All HP A700 and A900 computers having a serial number prefix of 2500 or greater.

NOTE

All A700 and A900 computers with serial number prefixes less than 2500 must have an upgrade kit installed for operation with the I/O extender. The upgrade kits are supplied as extender option 001 for the A900 and extender option 002 for the A700.

The I/O extender supports all standard A-Series I/O interface cards. Standard A-Series measurement and control cards require that the extender have an optional 25-kHz module installed.

Table 1-1. Functional Specifications

I/O CHANNELS ADDED BY I/O EXTENDERS

I/O EXTENDER PRODUCT NUMBER	NET I/O CHANNELS ADDED BY		
	ONE EXTENDER	TWO EXTENDERS	THREE EXTENDERS
12025A	12	24	36
12025B	18	35*	52**

* If the I/O control card is not the last one in the CPU, a jumper card uses one of the 18 card cage slots that would otherwise be available for an I/O card, reducing the net I/O channels added by a second or third extender to 17.

** The HP 1000 instruction set limits the number of I/O channels usable by the host computer to a maximum of 48, including those added by I/O extenders. Subject to operating system table limitations, 48 I/O channels are supportable with HP RTE-A having Rev. 2440 or later.

NOTE: The number of usable I/O channels may be significantly limited by the RTE-A system table space.

MAXIMUM DMA DATA TRANSFER RATES

Input: Same as the DMA input rate of the host computer; Mbytes per second as follows:

w/A600+	w/A700	w/A900
4.27	4.0	3.7

Output: Approximately 40 percent of the DMA output rate of the host computer (50 percent for A900); Mbytes per second as follows:

w/A600+	w/A700	w/A900
1.7	1.6	1.25

Chapter 2
Site Preparation and Requirements

Introduction

This chapter provides site preparation information for the HP 12025A/B I/O Extender.

Electrical Specifications

The 12025A/B I/O Extender is shipped with the power supply set to operate from a single-phase AC power source of either 86 to 138 volts (standard) or 178 to 276 volts (option 015) as specified in the purchase order. Changing from 115-Vac to 230-Vac operation (or vice versa) is described in Chapter 3. The input power specifications are given in Table 2-1.

The female power outlet to be used to supply ac power to the extender must be checked by a qualified electrician to ensure that it furnishes the proper voltage for which the extender is set. The outlet and its associated wiring and fuses (or circuit breakers) must be capable of carrying the current specified on the rear of the extender.

Environmental Specifications

Environmental limitations for operating and non-operating conditions of the I/O extender are specified in Table 2-1. The environmental limitations imposed by peripheral devices and associated components must be taken into consideration when the extender is located in the same area.

There are no external cooling requirements for the extender. The internal fans provide adequate ventilation when the extender is operated within the specified environmental limitations.

Physical Specifications

The I/O extender may be used either as a freestanding device or mounted in an EIA standard 483-millimeter (19-inch) equipment rack. Dimensions for the extender are specified in Table 2-1.

Table 2-1. Specifications

ELECTRICAL SPECIFICATIONS	
AC Power Requirements	
Standard:	115V -25%/+20% (86-138V), 47.5-66 Hz.
Option 015:	230V -23%/+20% (178-276V), 47.5-66 Hz.
12025A Power Consumption:	500W, maximum.
12025B Power Consumption:	700W, maximum.
DC Current Available and Required for I/O Interfaces:	The I/O extenders provide enough current for any combination of A-Series I/O interfaces that they can contain.
Optional 25 kHz Power Available and Required for Measurement and Control Interfaces:	12158A 25 kHz Module: supplies 50W. 12159A 25 kHz Card: supplies 30W. 12060A Analog Input: requires 7.3W. 12061A MUX for 12060A: requires 2.0W. 12062A Analog Output: requires 7.6W. 12063A Digital I/O: requires 11.4W 37203L HP-IB Extender: requires 0.8W.
ENVIRONMENTAL SPECIFICATIONS	
Temperature	
Operating:	0° to 55°C (32° to 131°F) to 3048 metres (10,000 ft); 0° to 45° C (32° to 113° F) to 4572 metres (15,000 ft).
Non-operating:	-40° to 75° C (-40° to 167° F).
Relative Humidity:	5% to 95%, without condensation.
Altitude	
Operating:	To 4.6 km (15,000 ft).

Table 2-1. Specifications (Continued)

Non-operating:	To 15.3 km (50,000 ft).
Vibration and Shock:	HP 12025A and 12025B I/O Extenders are type tested for normal shipping and handling shock and vibration. Contact factory for review of any application that will continuously vibrate the computer.
Safety and EMI Compliance	
Safety Qualification:	The 12025A and 12025B I/O Extenders comply with recognized international safety standards.
EMI Compliance:	The 12025A and 12025B I/O Extenders comply with Federal Communications Commission (FCC) and Verband Deutscher Elektrotechniker (VDE) standards for electromagnetic interference (EMI).
PHYSICAL CHARACTERISTICS	
Dimensions	
12025A:	17.8 cm (7 in.) high, 48.3 cm (19 in.) wide, 64.8 cm (25.5 in.) deep.
12025B:	26.6 cm (10.5 in.) high, 48.3 cm (19 in.) wide, 61.2 cm (24 in.) deep.
Weight	
12025A:	14.1 kg (31 lb).
12025B:	26.8 kg (59 lb).
Ventilation:	Four fans provide airflow for the 12025A or 12025B I/O Extender. In the 12025A, airflow is left-to-right through the card cage. In the 12025B, airflow is front-to-rear.

Chapter 3
Installation and Configuration

Introduction

This chapter provides installation and configuration information for the HP 12025A/B I/O Extender.

NOTE

A700 and A900 computers having a serial number prefix less than 2500 must have an upgrade kit installed for operation with the I/O extender. This chapter includes instructions for installing the upgrade kits.

Storage Before Unpacking

CAUTION

The extender must be protected from rapid temperature changes that cause condensation within it.

Prior to unpacking, the I/O extender may be stored in moderate environments within the following limits:

Temperature:	10 to 30 degrees C (50 to 86 degrees F)
Humidity:	20% to 70%, non-condensing
Altitude:	Up to 15.3 km (50,000 ft)

Unpacking and Inspection

The I/O extender box and printed circuit cards may be shipped in more than one container. When the shipment arrives, check to ensure the receipt of all containers as specified by the carrier's papers. Inspect each shipping container immediately upon receipt for evidence of mishandling during transit. If any container is damaged or waterstained, request the carrier's agent be present when the container is opened.

Open the shipping container marked "MANUALS AND ACCESSORIES." One of the items in this package is a list of equipment supplied. Compare this list against the purchase order to verify that the shipment is correct. Unpack the shipping container(s) and inspect each item for external damage. Look for damage such as broken controls and connectors, dented corners, bent panels, scratches, and loose components. Also check the rigid foam-plastic cushioning (if used) for signs of deformation which could be indicative of rough handling during transit.

If visual examination reveals any damage to the I/O extender, follow the damage claims procedure described in the next paragraph. Retain the shipping container(s) and packing material for examination in the settlement of claims or for future reuse.

Claims Procedure

If the shipment is incomplete or if the equipment is damaged or fails to meet specifications, notify your nearest Hewlett-Packard Sales and Service Office. If damage occurred in transit, notify the carrier also. Hewlett-Packard will arrange for replacement or repair without waiting for settlement of claims against the carrier. In the event of damage in transit, retain the packing carton and packaging materials for inspection.

Repacking For Shipment

Shipment Using Original Packaging

The same containers and materials used in factory packaging can be used for repacking the I/O extender. Alternatively, containers and packing materials may be obtained from Hewlett-Packard Sales and Service Offices. If the extender is being returned for servicing, attach a tag to the extender specifying the type of service required together with the extender model number and full serial number. Mark the container "FRAGILE" to ensure careful handling. In any subsequent correspondence, refer to the extender by model number and full serial number.

Shipment Using New Packaging

The following instructions should be used as a guide when packaging the I/O extender with commercially available materials:

a. Wrap the extender in heavy paper or sheet plastic. If shipping the extender back to Hewlett-Packard, first attach a tag to the extender with your return address and indicating the type of service required.

Include the extender model number and full serial number.

b. Use a strong shipping container. A double-wall carton constructed of 2.41 MPa (350-psi) test material is adequate.

c. Use sufficient shock-absorbing material on all sides of the extender to provide a firm cushion and to prevent movement inside the container. Use particular care to protect the extender corners and front and rear panels.

d. Seal the shipping container securely and mark it "FRAGILE."

e. In any subsequent correspondence with Hewlett-Packard refer to the extender by model number and full serial number.

Extender Switches

The I/O extender has two switches, either on its front (HP 12025A) or rear (HP 12025B). The switches are:

a. BATTERY or BACKUP. This two-position switch has no function when the extender box is used for the extender and may be set to either position. (When the +5VM voltage is activated the box is used for computers.)

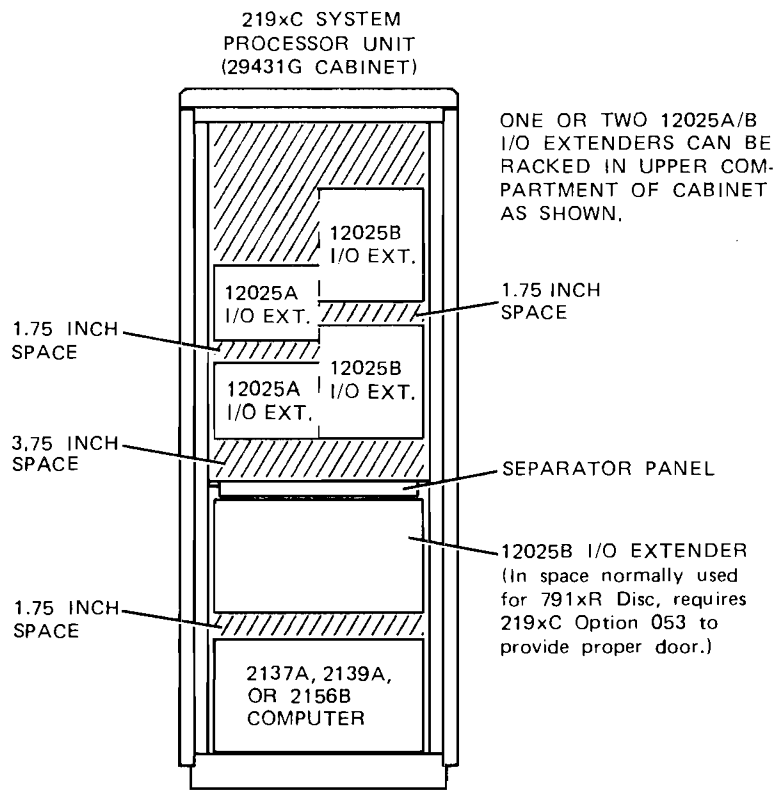
b. LINE. When this switch is set to the ON position, it applies ac line power to the extender power supply and fans. When this switch is set to OFF, it removes ac power from the power supply and fans.

Rack Mounting Instructions

Figure 3-1 shows the locations for rack mounting a 12025A/B I/O Extender in an HP 219xC/D SPU or 29431G/29429A Cabinet.

Cable Limitations In Older Cabinets

HP 219xA/B/C/D , 29431F, or 29429A Cabinets shipped before March 1, 1985 may not be able to accommodate more than 24 cables exiting the cabinet. Systems expanded so that 24 or more cables are exiting the cabinet may have to be racked in an HP 29431G or current 29429A Cabinet, which can accommodate 48 cables.


219xC SYSTEM PROCESSOR UNIT (29431G CABINET)
ONE OR TWO 12025A/B I/O EXTENDERS CAN BE RACKED IN UPPER COMPARTMENT OF CABINET AS SHOWN.
12025B I/O EXT.
12025A I/O EXT.
1.75 INCH SPACE
1.75 INCH SPACE
12025B I/O EXT.
12025A I/O EXT.
3.75 INCH SPACE
SEPARATOR PANEL
12025B I/O EXTENDER (In space normally used for 791xR Disc, requires 219xC Option 053 to provide proper door.)
1.75 INCH SPACE
2137A, 2139A, OR 2156B COMPUTER

8500-18

NOTES:

1. Only 12025A or only 12025B I/O Extenders may be installed in the upper compartment.

2. 12025B I/O Extenders in the upper compartment require removal of the upper front door of the cabinet (order 219xC or 29431G Option 051) to permit front-to-rear self-ventilation. (Use an HP 40027A Trim Kit to upgrade an existing cabinet.)

Figure 3-1. I/O Extender Mounting Positions In HP 219xC/D SPUs (HP 29431G/29429A Cabinets)

HP 12025A Rack Mounting

To install the 12025A I/O Extender in an HP 219xC SPU or an HP 29431G Cabinet, use HP 12679C Rack Rails and follow the instructions given in the 12679C Rack Rail Installation Manual, part no. 12679-90001. Note the following:

a. When installing only one 12025A, install each of the rails with screws in the eighth complete holes above the cabinet separator panel. This will leave a 3-1/2 inch space below the 12025A which can be covered by a filler panel.

b. When installing a second 12025A, install the rails with screws in the fifth complete holes above the first 12025A. This will leave a 1-3/4 inch gap between the 12025As which can be covered by a filler panel.

HP 12025B Rack Mounting

To install the 12025B I/O Extender in an HP 219xC-051 SPU or an HP 29431G-051 Cabinet, use HP 12679C Rack Rails and follow the instructions given in the 12679C Rack Rail Installation Manual, part no. 12679-90001. Note the following:

a. When installing only one 12025B, install each of the rails with screws in the eighth complete holes above the cabinet separator panel. This will leave a 3-1/2 inch space below the 12025B which can be covered by a filler panel.

CAUTION

Do not install a 12025A, or other equipment that has side-to-side air flow, in the upper compartment of the cabinet when a 12025B is installed there.

b. When installing a second 12025B, install each of the rails with screws in the fifth complete holes above the first 12025B. This will leave a 1-3/4 inch gap between the 12025Bs which can be covered by a filler panel.

To install the 12025B I/O Extender in an HP 219xD-053 SPU or an HP 29429A-053 Cabinet, use HP 12679C Rack Rails and follow the instructions given in the 12679C Rack Rail Installation Manual, part no. 12679-90001. Note that there must be a 1.75-inch space between the 12025B and the computer or other device; the space must be filled with an HP 12680B Filler Panel.

Mounting In Standard EIA Rack

To install the 12025A/B I/O Extender in a standard EIA 19-inch (483-millimetre) equipment rack, proceed as follows:

CAUTION

The rack in which the extender is mounted must allow for adequate ventilation of the extender. Refer to Table 2-1 for ventilation specifications.

a. Install equipment mounting rails in the rack.

b. Install the I/O extender on the rails.

c. Secure the extender in place with screws inserted through mounting holes in its front frame.

Floor Mounting (HP 12025A)

The HP 40025A Vertical Floor Mount may be used for vertical floor mounting of an HP 12025A I/O Extender box. Install the 12025A box in a 40025A by following the instructions provided with the 40025A.

Extender Cable

The I/O extender includes a cable (part no. 12025-60007) that connects between the IOC card in the CPU and the EXT card in the extender. The cable provides address, data, and control information between the two cards.

Installing the I/O Extender Cards

I/O Priority

The I/O extender IOC card must be installed in the CPU card cage and the EXT card must be installed in the extender card cage.

If the IOC card is not the lowest I/O priority card in the CPU card cage then the ICJ card must be installed in the extender in order to return the interrupt chain to the lower priority plug-in cards in the CPU card cage. If two extenders are hosted by a CPU, the first extender (that is, the extender whose IOC card is closest to the CPU processor cards) must have the ICJ card installed.

Note that the position of the IOC card, relative to the I/O cards in the CPU card cage, determines the I/O priority of all the I/O cards in the extender. If the IOC card has higher I/O priority than all I/O cards in the CPU, then all I/O cards in the extender will have higher priority than all I/O cards in the CPU. If the IOC card has the lowest I/O priority in the CPU, then all I/O cards in the extender will have lower priority than all I/O cards in the CPU. Note also that the IOC card may be installed anywhere among the I/O cards in the CPU.

NOTE

The maximum DMA output transfer rate via the I/O cards in the extender is only about 40/50 percent of the DMA output rate of I/O cards in the CPU card cage. Therefore it is recommended that I/O interfaces that are critical to the system performance, such as the system disc interface, be installed in the CPU and not in the extender.

If you are installing the HP 12025B I/O Extender, we recommend that you install the IOC card in the CPU card cage immediately to the left of the lowest I/O-priority card in the card cage. Doing so will eliminate the necessity of installing the Interrupt Chain Jumper (ICJ/18) card in the I/O extender and give you 19 I/O slots in the extender instead of 18.

NOTE

If any plug-in card is installed to the left of the IOC card in the CPU card cage, the ICJ/18 card must be installed immediately to the left of the lowest priority I/O card in the 12025B I/O Extender.

Installation

To install the IOC and EXT cards, do the following:

CAUTION

The contents of memory will be lost when the computer mains (line) and battery voltages are both off. Therefore, before proceeding, ensure that any contents of memory to be saved are stored on another medium for later retrieval.

NOTE

Correct computer operation requires that there not be any vacant slots between plug-in cards either in the CPU or in the I/O extender. Also, there must be at least one I/O interface card in the CPU card cage; you must not put all your I/O cards in the extender.

a. Set the computer Power switch to OFF. Set the computer battery backup switch to DISABLE.

b. Open the card cage doors of the computer.

c. Install the IOC card in the desired slot. The component side of the card must be facing either up or to the right.

d. Set the I/O extender Power switch to OFF.

e. Open the card cage doors of the I/O extender.

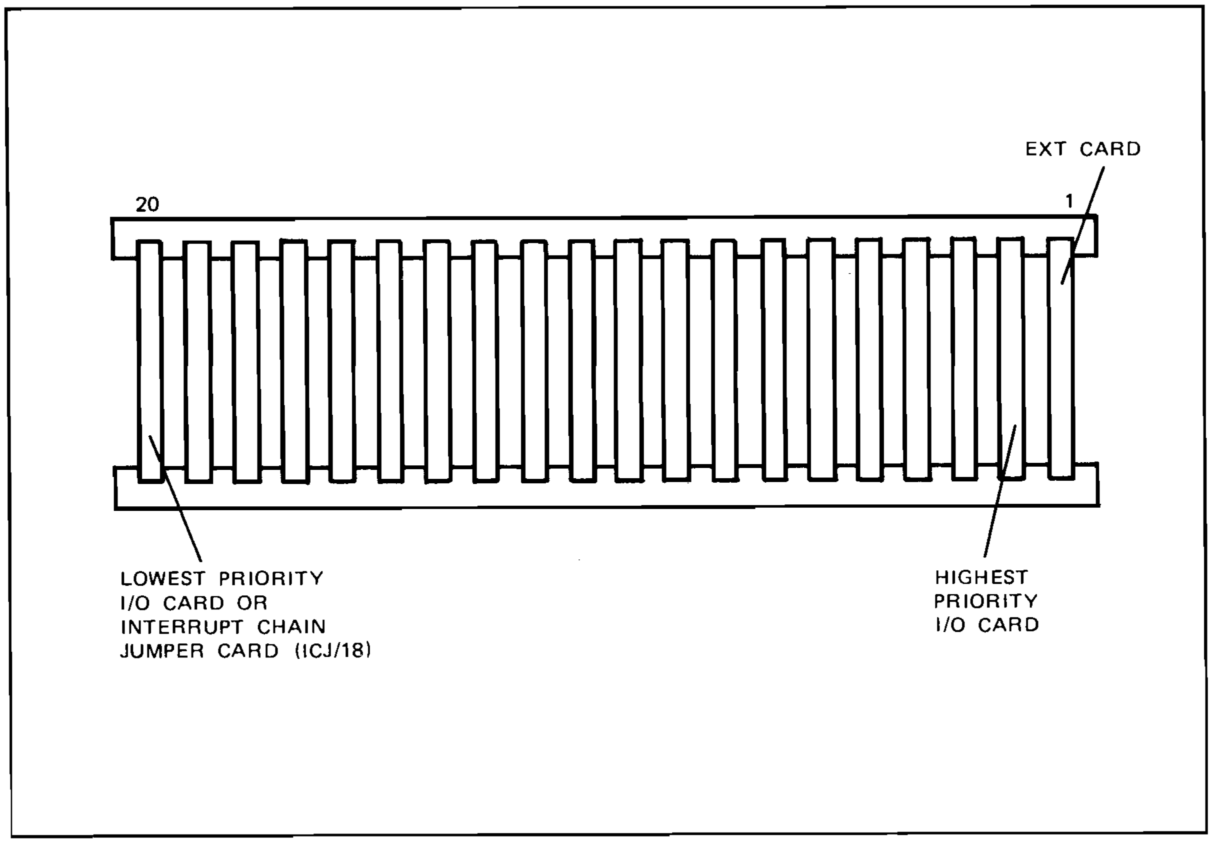
f. For the 12025B I/O Extender, install the EXT card in the rightmost slot (slot 1) with the component side of the card facing to the right. As noted in the preceding section, if the IOC card is not the lowest I/O-priority card in the CPU, install the ICJ/18 card immediately to the left of the last card in the extender. See Figure 3-2.

g. For the 12025A I/O Extender, the position of the EXT card depends on whether or not the IOC card has the lowest I/O priority in the CPU card cage.

   1. If the IOC card is the lowest I/O-priority card in the CPU, install the EXT card in the top right-hand slot (slot 1) of the extender, with the component side facing up. Install the ICJ/12 card in the lowest left-hand slot (slot 16) so that its voltage test points can be used.

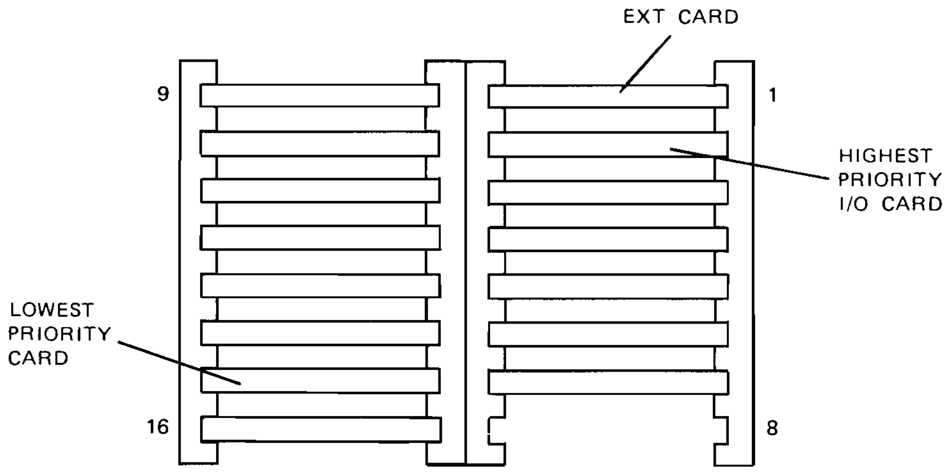
2. If the IOC card is not the lowest I/O-priority card in the CPU, install the ICJ/12 card in slot 16. Count backwards from slot 16 the number of slots required for I/O cards and install the EXT card in the next slot. For example, if 10 I/O cards are used, they will be installed in slots 15 through 5 and the EXT card goes into slot 4. (Slot 8 is reserved for the HP 12159A 25 kHz Power Module.) See Figure 3-3.

h. Install the I/O extender cable on the IOC and EXT cards. Connect the cable grounding straps to grounding lugs on the CPU and extender chassis. Figure 3-4 shows typical I/O extender-computer interconnections.


EXT CARD
20
1
LOWEST PRIORITY I/O CARD OR INTERRUPT CHAIN JUMPER CARD (ICJ/18)
HIGHEST PRIORITY I/O CARD

8400-81

Figure 3-2. Typical Location of Cards in HP 12025B



Notes:
1. Slot 8 is reserved for the optional HP 12159A 25 kHz Power Module.
2. Slot 16 is used only by the Interrupt Chain Jumper card (ICJ/12).

8300-50A

Figure 3-3. Typical Location of Cards in HP 12025A

Note that in the above example, the EXT card could be installed in slot 1 and the I/O cards in slots 2 through 12 provided that the ICJ/18 card is installed in slot 13. This would be an unusual installation since the ICJ/18 card is not supplied with the 12025A, but you should know that it is a workable installation.

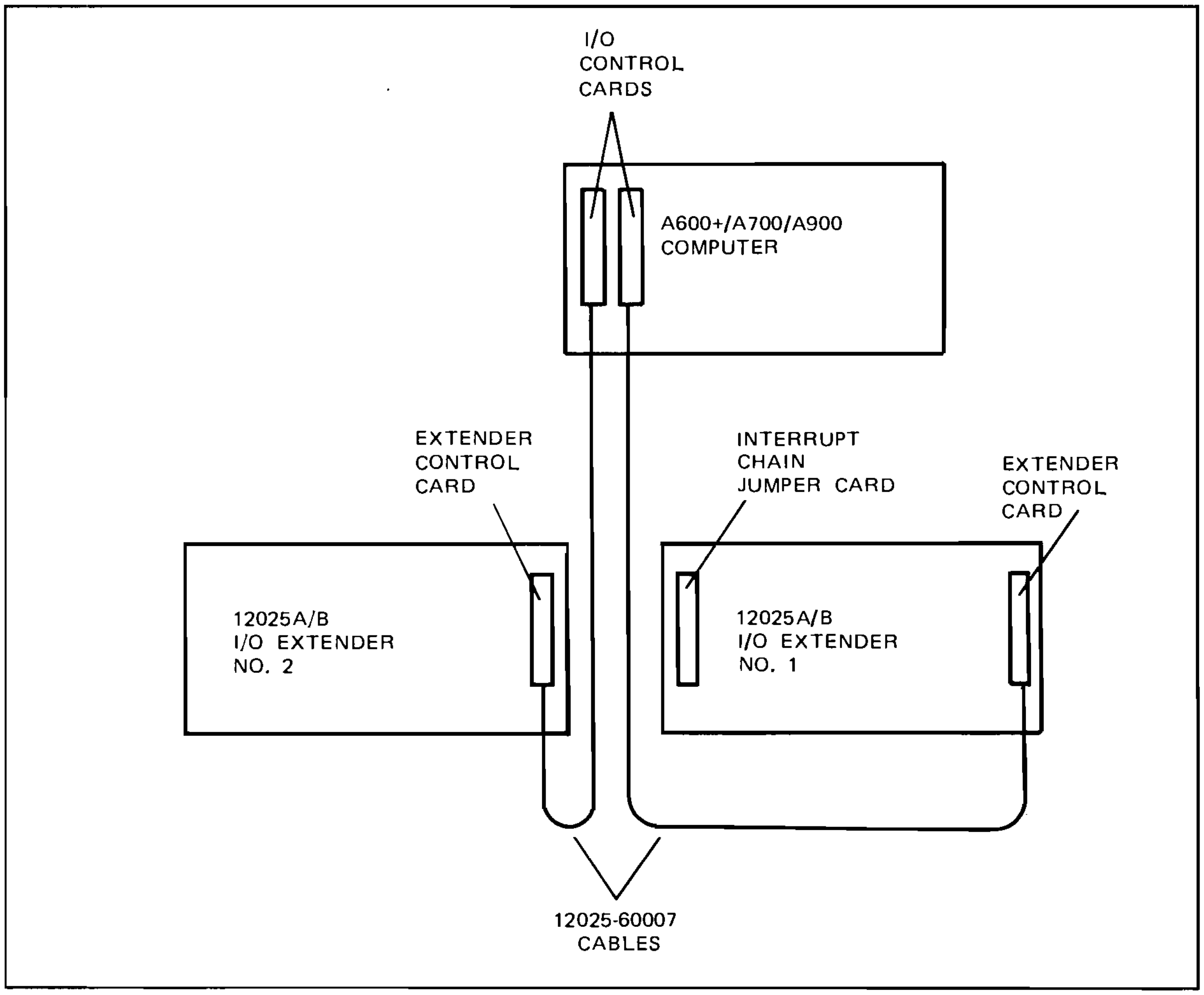
I/O Priority Within the I/O Extender

In the I/O extender, as in the CPU, a priority chain connects all interface cards in series to set the priority between simultaneous interrupt or DMA requests from two or more peripherals. The I/O priority of an interface card is determined by the slot that the card occupies, with slot 1 having the highest priority and slot 15 (in the 12025A) or slot 20 (in the 12025B) having the lowest priority. Interrupts from a higher priority device inhibit lower priority interrupts by breaking the priority chain. From the standpoint of system response time, it is more efficient to assign the higher priorities to high-speed peripheral devices.

NOTE

Correct computer operation requires that there not be any vacant slots between plug-in cards in the I/O extender. Note, however, that slot 8 in the HP 12025A I/O Extender is reserved for the HP 12159A 25 kHz Power Module and may be left vacant.

Refer to individual interface card documentation for installation details concerning card switches and priority considerations. Then consult the system manager to establish I/O device priority and install the interface cards accordingly. Note that the component side of the cards must be facing either up or to the right.



8400-84

Figure 3-4. Typical I/O Extender-Computer Interconnections

Interface Cabling

CAUTION

When connecting cables to the plug-in cards in the I/O extender, be sure to connect each cable to its appropriate card. Connect the I/O cable ground lug to the chassis.

Cable requirements to interconnect interface cards and associated peripherals are specified in the appropriate interface documentation. Set the Power switch to OFF and install the hooded connector of each cable onto the edge connector of the appropriate interface card, with the cable extending either down or to the right. Connect the cable grounding wire, if present, to a grounding strip on the extender chassis. Connect the other end of each cable to the appropriate peripheral device. Close the rear of the extender.

Installation of Upgrade Options

Option 001 Upgrade Card

Option 001 provides an upgrade card that enables an A900 computer having a serial number prefix less than 2500 to be used with the extender, provided that the HP 12201A Sequencer Card has suitable PROMs. The option consists of an HP 12203A Cache Control Card, part no. 12203-60011. If the Sequencer card does not have the following PROM part numbers or higher, contact your nearest HP Sales and Service Office.

Reference Designation	PROM Part No.
U803	12201-80084
U802	12201-80085
U801	12201-80086
U1103	12201-80087
U1102	12201-80088
U1101	12201-80089

To install the Option 001 card, proceed as follows:

CAUTION

The contents of memory will be lost when the computer mains (line) and battery voltages are both off. Therefore, before proceeding, ensure that any contents of memory to be saved are stored on another medium for later retrieval.

STATIC-SENSITIVE DEVICE. Use anti-static handling procedures when removing or installing plug-in cards.

a. Set the computer Power switch to OFF.

b. Open the card cage doors of the computer.

c. Remove the old Cache Control card by pulling outward on its extractor handles.

d. Install the new Cache Control card in the same slot. Push the card into the backplane connector by pressing on the extractor handles; make sure it is firmly seated in the backplane connector.

e. Execute the computer self-test using the instructions given in the A900 Computer Installation and Service Manual.

Option 002 Upgrade Kit

Option 002 provides an upgrade kit that enables an A700 computer having a serial number prefix less than 2500 to be used with the extender. The kit consists of the following PROMs for the 12152 Lower Processor Card.

PROM Part No.	Reference Designation
12152-80053	U91
12152-80054	U101
12152-80055	U111
12152-80056	U121

To install the Option 002 kit, proceed as follows:

CAUTION

The contents of memory will be lost when the computer mains (line) and battery voltages are both off. Therefore, before proceeding, ensure that any contents of memory to be saved are stored on another medium for later retrieval.

STATIC-SENSITIVE DEVICE. Use anti-static handling procedures when removing or installing plug-in cards or PROMs.

a. Set the computer Power switch to OFF.

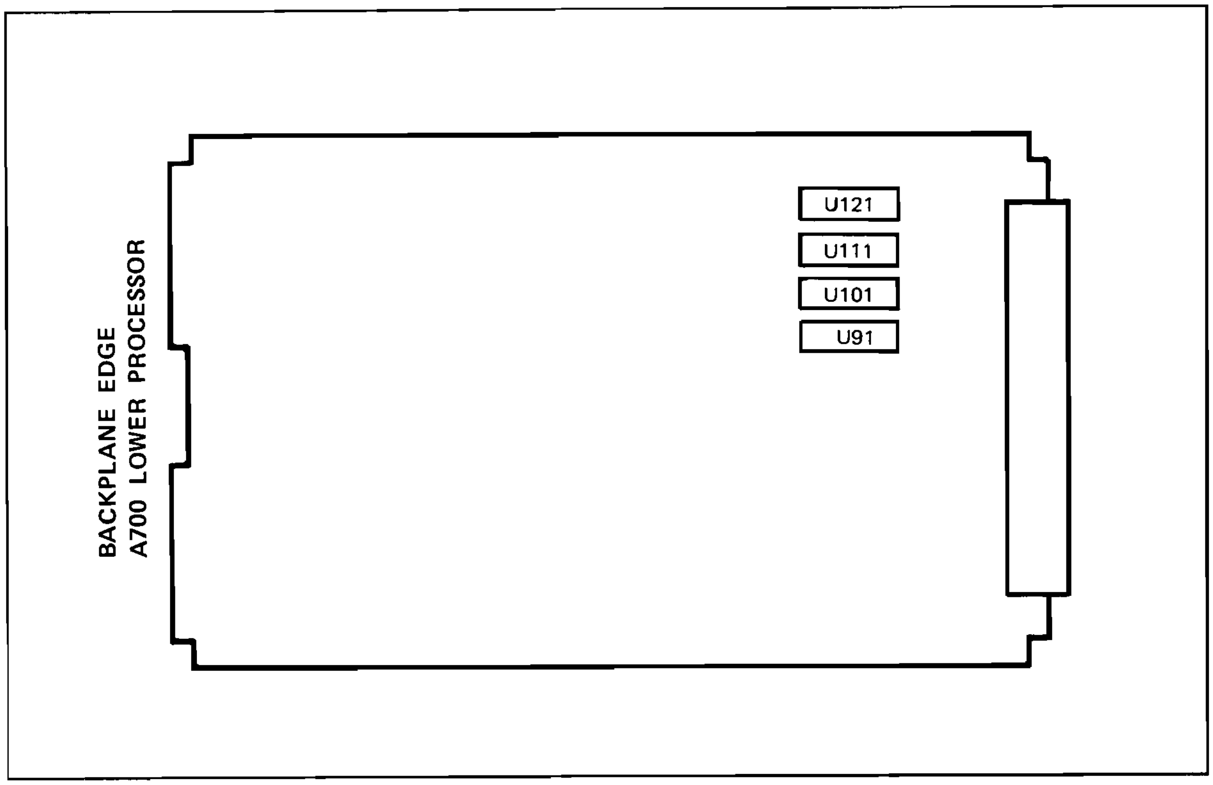
b. Open the card cage doors of the computer.

c. Remove the processor frontplane. Remove the Lower Processor card by pulling outward on its extractor handles. Remove the PROMs indicated above. (See Figure 3-5.)

d. Install the new PROMs, making sure that each one is in its correct socket and that the notch on each points to the backplane connector.

e. Reinstall the Lower Processor card. Push the card into the backplane connector by pressing on the extractor handles; make sure it is firmly seated in the backplane connector. Reinstall the processor frontplane.

f. Execute the computer self-test using the instructions given in the A700 Computer Installation and Service Manual.



8300-47 Figure 3-5. Location of New PROMs for A700 Lower Processor Card

Initial Checkout Procedure

After the I/O extender has been installed, check out the extender by doing the following.

a. Set the extender Power switch to OFF and connect the power cord to a power outlet having the electrical characteristics specified on the rear of the extender.

CAUTION

The 12025A I/O Extender is cooled by air being drawn in through vents in the left side and exhausted through vents in the right side. Make sure that these vents are not blocked while the extender is turned on. Also, do not operate the extender for more than a few minutes while either rear cover is removed.

b. Set the Power switch to ON.

c. Set the computer Power switch to ON.

d. Observe the green LEDs on the IOC card and the EXT card. Both these LEDs should be lit, indicating that the extender and computer dc power is up and that the extender-computer cabling is correct. If either green LED is not lit, refer to the troubleshooting information in Chapter 8.

Checkout Using Diagnostics

There are no diagnostics that can be used to verify proper operation of the I/O extender. However, you can make a confidence check of extender operation by running diagnostics on two or more I/O cards installed in the extender. To run the diagnostics, use the HP 24612A diagnostics and manuals. The diagnostic operating procedures are the same for I/O cards in the extender as they are for I/O cards in the CPU card cage. If the diagnostic tests fail, refer to the troubleshooting information in Chapter 8.

NOTE

The diagnostic for the HP 12009 HP-IB interface card must have revision 2440 or later in order to work with HP-IB cards installed in the I/O extender.

115/230 Vac Reconfiguration

WARNING

Reconfiguring the I/O extender to operate from 115- or 230-Vac line voltage must be done only by qualified personnel. Before changing from 115 Vac to 230 Vac configuration or vice versa, set the Power switch on the I/O extender to OFF and disconnect the power cord. Failure to observe this precaution may result in serious injury to personnel or damage to the power supply.

HP 12025A I/O Extender

To reconfigure the ac line power capability of the 12025A I/O Extender, proceed as follows:

a. Set the extender Power switch to OFF and disconnect the power cord.

b. Remove the extender front cover by grasping it at the sides and firmly pulling it away from the computer.

c. Remove the cable connector cover from the front of the power supply. See Figure 6-1.

d. Disconnect the voltage configuration cable connector from the front of the power supply by squeezing both ends of the connector. Insert the connector into the appropriate receptacle.

e. Install the metal cover plate over the unused connector.

f. Reinstall the cable connector cover.

g. Using the alternate power label supplied with the extender, placard the new power requirements next to the LINE receptacle on the rear of the extender.

HP 12025B I/O Extender

The ac line voltage configuration of the 12025B I/O Extender is determined by the position of the line configuration/fan power connector. The connector must be plugged into P8 in the power supply if the input line voltage is 115 Vac. (See Figure 6-2.) The connector must be plugged into P7 on the power supply if the input line voltage is 230 Vac.

If it is necessary to change the position of the line configuration/fan power connector, proceed as follows:

a. Set the LINE switch to OFF and disconnect the power cord.

b. Grasp the front panel by the two indented handles at the sides of the panel. Firmly pull the panel away from the I/O extender chassis.

c. Remove the four screws, four lock washers, and four flat washers securing the cover plate to the fan panel and remove the plate.

d. Remove nine screws, nine lock washers, and nine flat washers, and remove the fan panel from the extender chassis.

e. Remove the line configuration/fan power connector by squeezing the tabs at the ends of the connector.

CAUTION

When connecting the line configuration/fan power connector to the power supply, use connector P7 (see Figure 6-2) if the ac line input voltage is 230 Vac.

f. Insert the connector in the appropriate receptacle.

g. Replace the fan panel cover plate and the front panel.

h. If you have reconfigured the power supply from 115 Vac to 230 Vac operation, apply the 230 Vac label to the right rear door of the extender.

i. If you have reconfigured the power supply from 230 Vac to 115 Vac operation, remove the 230 Vac label from the right rear door of the extender.

Chapter 4
Preventive Maintenance

Introduction

This chapter provides preventive maintenance procedures for the HP 12025A/B I/O Extender.

Preventive Maintenance Procedures

Maintenance schedules for the I/O extender should be set up according to the quality of the environment in which the extender is operating. An extender in a clean and air-conditioned atmosphere requires much less preventive maintenance than one that is located in an atmosphere laden with dust, smoke, moisture, or other particulate matter.

Perform the following steps as often as necessary.

a. Clean the extender cabinet exterior and interior.

b. Check ventilating fans for proper operation. (For the 12025A, set the Power switch to OFF while observing the fans spin. The fans should continue to spin for at least 22 seconds.)

c. For the 12025A, remove the air filter and clean it by vacuuming its intake surface. To remove the filter, remove the front cover by grasping it at both sides and pulling it away from the extender; then slide the filter out of the box. See Figure 4-1.

d. For the 12025B, remove the air filter and clean it by washing it in a solution of warm water and mild soap. Thoroughly dry the filter before reinstalling it in the extender.

The ventilating fans in the I/O extender have sealed bearings and require no lubrication.

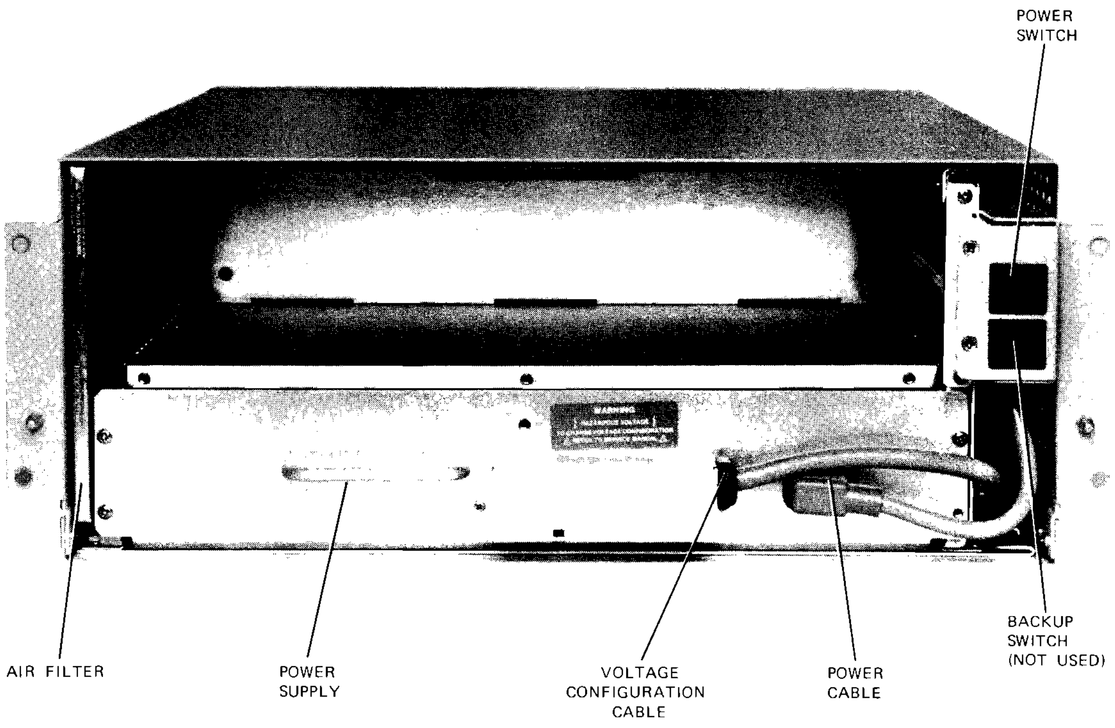


Figure 4-1. HP 12025A Front View, Cover Removed

Chapter 5
Functional Description

Introduction

This chapter provides functional descriptions of the IOC and EXT cards of the HP 12025A/B I/O Extender. It is assumed that the reader is familiar with the electrical and logical operation of the A-Series I/O architecture and with the operation of the A-Series computers. The documents listed in Chapter 10 provide general and detailed information on the A-Series computers. A working knowledge of the 20-pin and 24-pin PALs (MMI and AMD Programmable Array Logic) and the Series 20 IFL PLAs (Signetics Integrated Fuse Logic Programmable Logic Arrays) will aid you in understanding the operation of the I/O extender state machines. Refer to the A600+ or A700 Engineering Reference Documentation listed in Chapter 10 for detailed descriptions of the extender backplanes and power supplies. (Schematic diagrams of the IOC and EXT boards are in Chapter 12.)

Functional Overview

I/O Instruction Processing

The I/O extender utilizes two modes of operation to recognize and process I/O instructions.

In the A600+/A700 mode, the I/O extender acts as a coprocessor by decoding every instruction fetched. When an I/O instruction is detected during a processor instruction fetch, the I/O extender puts the CPU on hold for several clock cycles until it has the opportunity to initiate a DMA shutdown in the extender box. Current and queued DMA requests are allowed to run to completion. When the CPU finally broadcasts the I/O instruction, the I/O extender also broadcasts the same instruction on the extender backplane. Any response (I/O handshake request) from an I/O card in the extender results in a similar action by the IOC card on the CPU backplane.

In the A900 mode, all I/O extenders keep the CPU off the backplane bus until the CPU demands control. I/O cards in all extenders are asked to postpone further DMA transfers. When all DMA activity ceases, the A900 CPU sends out the I/O instruction and waits up to six backplane cycles for a response from the selected I/O card. The propagation of data and control signals to the extender and back incurs a four cycle overhead, thus requiring the CPU to wait beyond the current three cycle specification.

For either processor mode, the I/O extender always performs data input transfers (LI*, MI*) faster than output transfers (OT*). All input handshakes and data movements are pipelined for maximum throughput. Output transfers cannot benefit from this technique because data must be returned to the I/O card at the completion of the handshake. Output transfers require both an input operation (request for data) and an output operation (send data to interface card).

DMA Processing

The I/O extender handles Direct Memory Accessing (DMA) in the same manner for all A-Series processor types. As with I/O instruction processing, input transfers (DMA writes into main memory) are faster than output transfers (DMA reads from main memory). Again, the trick is to queue up all one-way operations at every stage.

In a DMA write, the I/O card sends the data and address simultaneously to the control logic in the extender box (EXT). This information is then passed to the I/O Control board (IOC) in the host CPU box. A DMA write into the computer's main memory is performed as the last step in this long sequence of events. Because the entire procedure is pipelined, it is not necessary to wait until the first DMA write has written to main memory before the second DMA write begins. In fact, up to three consecutive DMA writes may be active or queued up at the same time. The peak DMA write bandwidth is equal to the maximum DMA bandwidth of the host computer.

For a DMA read, the requesting I/O card sends only the address to the extender box control logic. The extender backplane is suspended until the data has been read from main memory and sent to the extender. The current DMA read cycle is completed only after the I/O card receives the data it has requested. Therefore, only one DMA read cycle may be active at any time in a given I/O extender. The overall DMA read bandwidth is significantly lower than that of the DMA write bandwidth. However, this bandwidth limit applies only to the extender backplane; the host computer DMA (read or write) bandwidth is not affected.

If a DMA read cycle immediately follows a series of full bandwidth DMA write transfers, the extender backplane is suspended until data is returned for that DMA read cycle. At the CPU backplane, all DMA writes complete in sequence prior to servicing the DMA read request. In this scenario, the resultant DMA read waiting period is much longer than the quiescent case where there are no preceding DMA writes.

Interrupt and Slave Processing

When interrupts are requested by an I/O interface card in the extender, a corresponding request is made on the CPU backplane. Interrupt requests from all I/O cards in an extender are condensed into a single interrupt request for the CPU backplane. Likewise, the interrupt priority chain is processed at the point in the CPU backplane where the interrupt is requested. If the extender makes an interrupt request, I/O cards in the host backplane with a lower priority than the extender are disabled.

An interrupt acknowledge from the CPU is passed to the extender backplane only if it has an interrupt pending and is not disabled by a higher priority interface card. When there are multiple interrupts within an extender, priority is resolved again using the extender backplane's own interrupt priority chain. The CPU processes the instruction in the trap cell when the interrupting I/O card vectors (causes an instruction "jump") the CPU to the memory location corresponding to its select code.

Slave processing is very similar to interrupt processing. A request is made but acknowledgement is sent down the slave chain. Instead of using a memory read protocol to vector to a trap cell, the interface card initiates a series of I/O handshakes to change the state of the CPU.

General Description

I/O Extender Components

The following components make up the I/O extender subsystem:

a. IOC 12025-60001 I/O Control board.

b. EXT 12025-60002 Extender Control board.
c. ICJ/12 12025-60003 (for use in 12025A Extender) or
   ICJ/18 12025-60004 (for use in 12025B Extender)
   Interrupt Chain Jumper board (included in the product, but use is optional; read the section detailing ICJ board).

d. Shielded round cable with 80-pin connector hood on both ends.

e. A 16-slot or 20-slot box with power supply.

The IOC board resides in the host computer backplane and occupies one I/O slot. It interfaces with the CPU backplane and arbitrates between CPU backplane resources and the extender control board requests. The I/O Control board generally conforms with the A/L-Series backplane handshaking protocols and may be treated as an I/O card which does not have its own select code.

The EXT board resides in the extender box as the highest priority card in the card cage. It may be thought of as a processor-memory subsystem since it controls the actions of the I/O cards in the extender. The EXT board is responsible for broadcasting I/O instructions and for handling DMA requests.

The ICJ board resides in the extender box as the lowest priority card in the card cage. Its purpose is to return the interrupt priority chain information from the last I/O interface card to the EXT board. This data is used by the I/O extender subsystem only during the execution of Diagnose Mode 1 or 2. This card is not required if the extender's IOC card is the lowest priority card in the host CPU backplane.

The interconnecting cable is a shielded round cable carrying 75 separate signals. The cable is terminated on both ends with 80-position PC (printed circuit) card edge connectors. The system clock and three power supply status lines are constructed as twisted pair wires while the remaining 72 signal lines are connected point-to-point.

Naming Conventions

The following prefixes are used throughout this chapter to indicate the origin (which board) of the named signal:

BP.xxxxx is a CPU box I/O backplane signal.
XP.xxxxx is an extender box backplane signal.

IOC.xxxxx is a signal on the IOC board.
EXT.xxxxx is a signal on the EXT board.

IO.xxxxx is signal on the cable driven by the IOC, received by EXT.
EX.xxxxx is signal on the cable driven by the EXT, received by IOC.

IOC to EXT Communication

Communication between the IOC and the EXT boards is divided into the following categories:

EXT to IOC

15 bits	Address Bus
5 bits	Address Extension Bus
1 bit	Self-Configuration DMA (AE5+)
1 bit	Read/Write Bit
16 bits	Data-In Bus
9 bits	Control-In Bus

IOC to EXT

16 bits	Data-Out Bus
1 bit	Parity Error Status
1 bit	System Clock
10 bits	Control-Out Bus

The 22 bits of address, address extension, self-configuration indication, and read/write status are latched versions of the extender backplane signals. The data bus is not bidirectional because of timing conflicts which may result from clock skew. The data-in bus is a latched version of the extender backplane data bus, and the data-out bus is a latched version of the CPU backplane data bus. A parity error status bit is also sent to the extender at the end of every memory read operation.

The CPU backplane clock, BP.SCLK-, is buffered and sent to the EXT board for use on the extender backplane. There are four buffer delays from the CPU backplane to the extender backplane. All communication between the EXT and the IOC board are synchronous with respect to BP.SCLK-.

There are a total of nineteen control lines between the EXT and IOC board. All control signals change state at the start of the long half cycle. The EXT board samples these signals at the start of the short half cycle (135 nsec to 167 nsec settling time permitted from transition to signal sampling). The IOC board always samples the control signals at the start of the next long half cycle, with a settling time of one cycle, minus clock skew (IOC to EXT) and signal delay (EXT to IOC), which works out to approximately 170 nsec.

The IOC board issues the following control signals:

a. IO.ACKDMA- acknowledges that the requested DMA cycle has started.

b. IO.DATAAV- informs the EXT that valid data will be on the Data-Out bus during the next cycle.

c. IO.INSTRAV- commands the EXT to perform an I/O instruction broadcast using the data previously received with IO.DATAAV-.

d. IO.ACKINT- commands the EXT to generate an interrupt acknowledge for an I/O interrupt requested in the extender.

e. IO.ACKSLV- commands the EXT to generate a slave acknowledge for a slave handshake requested in the extender.

f. IO.ACKIOHS- acknowledges that the requested I/O handshake cycle has started (also signals whether current I/O instruction to be broadcast is of the OT* 0 or OT* 2 variety).

g. IO.CTURNF- commands the EXT to generate XP.CPUTURN- in the extender.

h. IO.IOCICHOD- is the interrupt chain priority status on the CPU box backplane sent to the EXT for proper sequencing of I/O cards during the execution of Diagnose Mode 1 or 2.

i. IO.CNTLRST- informs the EXT to generate a control reset (XP.CRS-) pulse on the extender backplane.

j. IO.IOCPON- informs the EXT about the power-on status of the host box.

The EXT board issues the following control signals:

a. EX.REQDMA- commands the IOC to assert BP.MRQ- on the host backplane.

b. EX.ABDBAV- informs the IOC that the address and data-in buses are valid beginning with the start of the next cycle.

c. EX.INTFLG- commands the IOC to assert BP.INTRQ- on the host backplane and arbitrate the interrupt priority at the CPU backplane.

d. EX.REQSLV- commands the IOC to initiate a slave handshake request.

e. EX.REQIOHS- commands the IOC to initiate an I/O handshake (typical response to an I/O instruction or slave acknowledgement).

f. EX.OTAFLG+ informs the IOC about the direction of data flow during multiple I/O handshake operations (LI*, MI*, OT*, HLT and break).

g. EX.EXTICHOD- informs the IOC about the status of the interrupt chain following the last I/O card in the extender (useful only during the execution of Diagnose Mode 1 or 2).

h. EX.EXTPFW+ informs the IOC about an impending power failure in the extender box.

i. EX.EXTPON- informs the IOC about the power-on status of the extender box.

Priority Chain Lines

There are three priority chain lines in the A/L-Series I/O architecture:

a. DMA priority chain.

b. Interrupt priority chain.

c. Slave mode processing priority chain.

During normal operation of the extender (non-diagnose mode), the CPU backplane priority chain lines are not coupled to the extender priority chain lines. Whenever the extender is inactive (no DMA nor interrupt nor slave request pending/in progress), priority information from the card above the IOC is passed directly to the interface card below the IOC. If a card in the extender desires a DMA cycle, the IOC disables lower priority cards in the host backplane when it makes a request for a memory cycle. At the completion of that DMA data transfer, the IOC allows lower priority cards to perform DMA again. Therefore, all protocols which rely on a priority chain for operation (except certain diagnose modes) treats the IOC as a regular interface card.

Within the extender box, the DMA and interrupt priority chains start out with the top priority card always enabled. The slave handshake priority chain is always disabled except during a one cycle deassertion in response to an IO.ACKSLV- message from the IOC.

I/O Control (IOC) Board Description

The IOC (I/O Control) board is the extender's port to the host CPU. It performs real-time arbitration with other I/O channels for memory resources while serving the CPU by distributing I/O instructions to the interface cards in the extender. Three state machines control all DMA, I/O handshaking, and interrupt/slave processing activities. The balance of the IOC board logic is dedicated to the transfer of address bus and data bus information between the CPU backplane and the Extender Control board (EXT).

Identification of Processor Type

Processor identification is automatically determined by the IOC when the CPU box is powered up. The A600+/A700 processors make the IOC operate in a mode that is slightly different than that used for the A900 processor. The IOC defaults to the A600+/A700 mode of operation when the unit is initialized (on power-up or when CPU is reset). The A600+/A700 always fetches an instruction before broadcasting to the I/O cards, while the A900 processor broadcasts only I/O instructions over the I/O backplane. Based on this known handshaking sequence, the IOC sets two flags. The IOC.MODESEL- flag indicates that the IOC has successfully determined the processor type as reported by the IOC.A900MODE- flag. This information is utilized by the backplane arbitration logic to properly synchronize the CPU to the extender box.

DMA State Machine

The DMA state machine generates the backplane signals BP.MCHOD-, BP.MCHODOC-, BP.MRQ-, and BP.MEMGO-. The only information received from the extender are EX.REQDMA- (request DMA) and EX.ABDBAV- (address and/or data bus available). The IOC generates BP.MRQ- at the earliest convenient cycle after the receipt of EX.REQDMA-. BP.MEMGO- is generated only after EX.ABDBAV- is received. In either case, if the host backplane is busy during the cycle the requests are received, an MRQpending (IOC.MRQPD-) and/or MEMGOpending (IOC.MEMGOPD-) flag is set. When the backplane is again available for a DMA cycle, those pending flags cause BP.MRQ- and/or BP.MEMGO- to be generated.

The DMA handler examines the state of the backplane using the BP.IAK-, BP.IOGO-, BP.BUSY-, and BP.MCHID- signals. Whenever BP.IAK- and BP.IOGO- is asserted just prior to the IOC's generation of BP.MRQ-, no MRQ is generated until BP.IAK- and BP.IOGO- return to their inactive states. Likewise, once the IOC has asserted BP.MRQ-, it cannot assert BP.MEMGO- until BP.BUSY- and BP.MCHID- are both inactive. Once BP.MEMGO- is generated by the IOC, an IOC.DMACYC- flag is set to indicate that a DMA cycle is currently in progress. BP.VALID- is asserted by the memory controller to indicate that valid data is on the CPU box backplane and/or that the memory cycle has completed.

The CPU backplane signals BP.MCHOD- and BP.MCHODOC- are asserted every time the IOC generates BP.MRQ- to service an extender DMA request. These signals disable lower priority I/O cards from performing a DMA transfer during the next cycle. There are other occasions when the IOC needs to assert BP.MRQ- (to "freeze" the CPU), but those operations do not affect the two DMA chain signals.

I/O Handshake State Machine

The I/O handshaking logic generates BP.IORQ- in response to an EX.REQIOHS- (request I/O handshake) message from the EXT for all handshaken I/O instructions except OTA/B 0/2. The EXT board also issues an EX.OTAFLG+ message for all I/O handshakes which require data to flow from the CPU backplane to the extender. The default data flow is from the extender to host, allowing a control word from an I/O card in the extender to pass to the host backplane without prior decoding (for speed). The I/O handshaking control word may be decoded during the idle cycle following the first handshake. If a bus reversal is required, the IOC is informed to read data off the CPU backplane data bus rather than writing to that bus.

Since slave processing is similar to I/O processing, the I/O handshake state machine also handles slave mode handshaking. The handshaking procedure is always handled as consecutive control word/data transfer pairs.

Address and Data Bus Transfers

The address register, address extension register, self-configure bit flip-flop, write-enable bit flip-flop, and the data-in bus register are simultaneously updated one cycle after an EX.ABDBAV- (address bus and data bus available) message from the EXT board. All registers are updated even if some of the registers are not used during the current operation. The unused registers contain invalid information (that's why they're unused!). The EX.ABDBAV- message is utilized for DMA read, DMA write, I/O handshake read, and I/O handshake write operations. In the case of the I/O handshake write operation, none of the updated registers are used.

The IOC card sends data bus-out and memory parity information to the EXT card. The IO.DATAAV- message informs the EXT card to latch the data-out bus and memory parity bit during the next clock cycle. Again, IO.DATAAV- is issued without knowledge of the data's usefulness. An IO.DATAAV- is issued during DMA read, DMA write, I/O handshake read, and I/O handshake write operations. The IO.DATAAV- message results in meaningless data being transferred to the I/O extender for DMA write or I/O handshake read operations.

I/O Instruction Processing

The original L-Series I/O architecture called for distributed op-code decoding. All I/O cards suspended their DMA activities when the CPU fetched instructions. Each card independently determined whether the instruction must be executed. If an average program contained approximately 5% mix of I/O instructions, the I/O cards wasted a significant amount of time decoding other instructions.

In the A-Series I/O architecture, normal I/O interface cards do not decode every instruction fetched by the CPU. If the instruction fetched by an A600+/A700 CPU is an I/O instruction, those CPUs then proceed to or "broadcast" that I/O instruction to the I/O cards. Using this scheme, the I/O cards can attend to DMA activities, only to be bothered if the CPU must communicate with the I/O card. The A900 processor fetches its instructions over an internal bus between the CPU and memory, so that first instruction fetch does not even appear on the backplane.

The A-Series I/O extender pretends to be smarter than ordinary I/O cards. It carries out I/O instruction execution in two stages. During the first stage, it ensures that all of the I/O cards in the extender are able to perform DMA whenever they choose, until the extender detects that there is an impending I/O instruction in the program flow. This is where the A600+/A700 mode of operation differs from the A900 mode for arbitrating I/O instruction execution and DMA activities. The second stage is the actual broadcast of the I/O instruction on the extender backplane. The sequence of events during the second stage is the same for all processor types.

In the A600+/A700 mode, a fetched I/O instruction triggers two simultaneous events. In the CPU box, the extender makes a fictitious assertion of BP.MRQ- to keep the processor from broadcasting the I/O instruction to the I/O cards. This is done for at least two clock cycles so that any DMA transfers in progress (or initiated during the instruction fetch) can be completed before the I/O cards in the extender box turn their attention to interpreting the I/O instruction. Meanwhile, the extender asserts XP.CPUTURN- on the extender backplane to force the suspension of all DMA activity after all pending DMA transfers have completed. The extender manipulates the host backplane as well as its own backplane in order to gracefully mediate DMA and instruction processing activities.

For the A900 mode, the extender always has BP.MRQ- asserted. Thinking that DMA is always in progress, the CPU attempts to suspend that activity by asserting BP.CPUTURN-. Upon receiving this request, the I/O extender also asserts XP.CPUTURN- in the extender backplane. As mentioned above, the I/O extender keeps BP.MRQ- asserted for at least two cycles after the assertion of BP.CPUTURN- to keep the CPU from continuing until all pending DMA transfers are completed.

At the end of the first stage for the handling of I/O instructions, the extender backplane is guaranteed to be free of any interfering DMA while the instruction is broadcast to all of the I/O cards in the extender. After the extender has released the processor to broadcast the I/O instruction, that instruction is passed to the extender backplane for the extender's I/O cards to interpret.

The actual instruction is transferred to the EXT card from the IOC when the EXT receives the IO.DATAAV- message from the IOC. The IO.INSTRAV- message informs the EXT that the data received should be used to generate an instruction broadcast memory cycle on the extender backplane. If the I/O instruction is of the OT* 0 or OT* 2 variety, then an IO.ACKIOHS- message is sent with the IO.INSTRAV- message. The OT* 0/2 instruction is still broadcast, but the subsequent I/O handshaking is done using special processing on the IOC and EXT boards.

From the time the CPU broadcasts a SFS/C, OT*, LI*, MI*, or HLT I/O instruction, it waits up to three BP.SCLK- cycles before timing out and executing the instruction as a NOP. Because there is a delay in getting the instruction out to the extender and then a delay for getting the response (XP.IORQ- to BP.IORQ-) back to the CPU, the CPU should now wait five to seven BP.SCLK- cycles before timing out. The A600+ CPU does not have a problem with this because they already spend approximately eight cycles performing household chores before sampling BP.IORQ-. The A700 and A900 CPUs however, check BP.IORQ- almost immediately after the I/O instruction broadcast. These two processors required a microcode change that gives the I/O extender a chance to perform an I/O handshake with the CPU.

Interrupts and Slave Processing

Computer Museum

Although there may be simultaneous interrupts from several I/O cards in the extender box, only one interrupt at a time is processed by the extender. Within each extender, concurrent interrupts are processed serially, from the highest priority to the lowest priority. From the extender's point of view, it handles these concurrent interrupts as a sequence of interrupts. The handshaking procedure is identical; the only difference is the address of the trap cell fetch.

The EXT board generates an EX.INTFLG- message for the IOC when there is an interrupt request in the extender. The IOC converts this message into the backplane BP.INTRQ- signal to flag the processor that an interrupt request condition exists. Some time later, the CPU acknowledges that interrupt with a BP.IAK- signal. If the extender's IOC is the highest priority interrupt requester, the IOC issues an IO.ACKINT- message for the EXT. This message becomes XP.IAK- on the extender backplane. The interrupting interface card then begins a memory cycle with XP.MEMGO-, causing the EXT to issue an EX.ABDBAV- message for the IOC. The IOC then generates a BP.MEMGO- on the host backplane using the interrupting card's select code as the address for the memory read cycle. The processor detects the occurrence of this memory cycle and uses the returned data as the trap cell instruction.

Slave processing is similar to interrupt processing. The extender backplane slave signal (XP.SLAVE-) is converted into the EX.REQSLV- message for the IOC. The IOC then generates the BP.SLAVE- signal on the host CPU backplane. The processor acknowledges the request by deasserting BP.SCHOD- for one cycle. The IOC detects this and issues an IO.ACKSLV- message to the EXT. The IO.ACKSLV- message causes the extender backplane's XP.SCHOD- signal to be deasserted for one cycle. Some time later, the slave requesting interface card begins an I/O handshake cycle with the CPU via the extender.

Status LEDs and Extender Not Ready Flag

The IOC board utilizes two LEDs to report its operating status. The IOC is initialized with the red LED on at the time the host CPU is powered up.

The red LED is lit under any of the following conditions:

a. DC power not within regulation in mainframe.

b. Processor being manually reset.

c. Extender powered down.

d. DC power not within regulation in extender.

e. IOC to EXT cabling damaged or not connected.

The green LED is lit only when both the IOC and EXT are properly powered up and the interconnecting cable is properly connected.

Related to the status reporting LEDs, there is an EXTNRDY flag (Extender Not Ready). This flag is a flip-flop which remembers whether the extender has ever powered up correctly after the host CPU has powered up. If the extender is off when the CPU is powered up, the IOC keeps the CPU from completing the power-up self-test until the extender is powered up. At some later time, the extender may be powered off (causing the red LED on the IOC to light up), which causes the host box to undergo a "power fail" condition. The simulated power failure is needed for the operating system to shut down gracefully. If the extender were to lose power without informing the CPU, it would be possible for the extender to get stuck in an illegal state (IOC starts DMA request in the host backplane but never gets the associated data from the EXT -- this would hang up the host backplane and appear as a catastrophic system-wide failure).

Extender Control (EXT) Board Description

The EXT (extender Control) board acts as a virtual CPU/memory subsystem for all of the I/O cards in the extender box. In this role, the EXT is an I/O concentrator, collecting all of the DMA and interrupt activities from the individual I/O cards into a single channel for the IOC to present to the real CPU and memory in the host CPU box. Three state machines control all DMA, I/O handshaking, and interrupt/slave processing activities. The balance of the EXT board logic is devoted to the transfer of address and data bus information between the extender backplane and the I/O Control board (IOC). The architecture of the control logic for the EXT board is almost a mirror image of the control logic on the IOC board.

DMA and Instruction Broadcast State Machine

The DMA state machine of the EXT board generates the backplane signals XP.BUSY-, XP.VALID-, XP.RNI-, and XP.CPUTURN-. The XP.MCHOD- signal is pulled high, so that the highest priority card in the extender box is always enabled to perform DMA. The DMA state machine receives the IO.ACKDMA- (acknowledge DMA), IO.DATAAV- (data available), IO.INSTRAV- (instruction available), and IO.CTURNF- (cputurn flag) messages from the IOC card.

XP.BUSY- is asserted in direct response to XP.MEMGO-. XP.VALID- is asserted simultaneously with XP.BUSY- only for DMA writes to memory if the DMA write queue is empty. For memory reads, XP.VALID- comes only after the receipt of an IO.DATAAV- message from the IOC, indicating that data is available on the cable. Queued memory write cycles are terminated with XP.VALID- after an IO.ACKDMA- message is returned by the IOC card, indicating that the previous DMA request is now being processed.

The DMA state machine has the capability to handle more than one DMA transfer simultaneously. If the CPU backplane is temporarily saturated and unavailable to perform DMA transfers at high speeds, the extender is able to handle a backlog of DMA writes into memory. A status flag flip-flop (EXT.DMAQUEUE-) keeps track of how deep the backlog has become to help determine when to start accepting new addresses/data from the extender backplane. In the worst case, one DMA transfer is in progress in the CPU backplane, another transfer is ready to be accepted by the IOC card, and a third transfer is ready to use the extender backplane address/data bus. The use of this pipelining technique complicates the design, but pays off tremendously in the area of DMA write performance. The maximum data input transfer rate is equal to the DMA bandwidth of the host mainframe, while the maximum output transfer rate is only 40 (or 50) percent of the host DMA read bandwidth. DMA writes to memory typically require two clock cycles while DMA reads from memory typically take five cycles to complete, as seen by an I/O card in the extender. Viewed from the CPU backplane, the IOC card performs the DMA transfer (read or write) in two clock cycles (the same as any other I/O card).

When the CPU broadcasts an I/O instruction, that instruction is subsequently broadcast over the extender backplane. The instruction is passed to the EXT card from the IOC card using the IO.DATAAV- message. When the IOC has decoded the instruction, it issues the IO.INSTRAV- message. IO.INSTRAV- causes the EXT to broadcast the data received using XP.BUSY-, XP.VALID-, and XP.RNI- asserted for a single clock cycle.

XP.CPUTURN- is the backplane signal utilized by the EXT card to facilitate orderly suspension of DMA prior to and during instruction broadcasting, I/O handshaking, and interrupt acknowledgement. XP.CPUTURN- is generated locally by the EXT card for most cases. The IOC card uses the CTURNF message to supplement the EXT's generation of XP.CPUTURN- because the IOC has access to some signals not available to the EXT.

The CPU's use of BP.CPUTURN- in the host backplane is only significant in the A900 mode of operation. When the CPU generates BP.CPUTURN-, the IOC automatically issues an IO.CTURNF- message to the EXT. In the A600+/A700 mode, the IOC issues an IO.CTURNF- message whenever an I/O instruction is fetched by the CPU. This look-ahead technique is used to clear the extender backplane of DMA activity before the real instruction broadcast occurs. The EXT board generates its own EXT.CPUTURNFF- when it receives the IO.INSTRAV- message from the IOC. XP.CPUTURN- remains asserted if an I/O handshake follows, otherwise it times out after five clock cycles.

I/O Handshake State Machine

The I/O handshake state machine of the EXT board generates the backplane signal XP.IOGO- and monitors the backplane signal XP.IORQ-. It generates the EX.REQIOHS- (request I/O handshake), EX.OTAFLG+ (output data transfer handshake), and EX.ABDBAV- (new address and/or data available) messages for the IOC board, and receives the IO.ACKIOHS- (acknowledge I/O handshake) and the IO.DATAAV- (new data available) message from the IOC board. In multiple handshake protocols, it keeps a record of whether the handshake is an even-numbered or an odd-numbered handshake. During odd-numbered handshakes, control codes are passed from the interface card to the EXT to the IOC and then finally to the CPU. The direction of data flow during an even-numbered handshake is dependent on the control code passed to the CPU during the preceding cycle.

The direction of the data flow during a sequence of I/O handshakes is maintained by the EXT.OTAFLAG- flip-flop. This flag is always false during the odd-numbered (first) handshakes because data (the control word) is always going to the CPU. It is true during the even-numbered (second) handshakes only if data is to flow from the CPU to the I/O extender and eventually, to the interface card. The EXT.OTAFLAG- becomes true only during the execution of the OTA/B and HLT instructions and during a VCP terminal break (slave mode I/O transfers). No other I/O instructions invoke the assertion of EXT.OTAFLAG-.

Like the DMA state machine, the I/O Handshaker also operates more efficiently for input (towards the CPU/Memory) operations than output operations. Instructions such as LIA/B and MIA/B execute in fewer clock cycles than the OTA/B instruction. For an LIA/B or MIA/B, the I/O state machine handshakes the control word with the interface card, and if the control word indicates that an input handshake occurs next, the state machine will accept the data as soon as possible. For an OTA/B instruction, the state machine must wait until the control word has travelled to the CPU, and for the CPU to deliver the data to the IOC card and then to the EXT card, before the second handshake may even be acknowledged. The difference in speed between an LIA/B instruction and an OTA/B instruction is approximately four clock cycles.

Interrupts and Slave Processing

The EXT board receives all interrupt requests and issues a EX.INTFLG- message for the IOC board. When the CPU acknowledges the interrupt, the EXT receives an IO.ACKINT- message from the IOC board. The EXT asserts XP.IAK- on the extender backplane and waits for the interrupting card to respond with a memory read to the memory location associated with its select code. As explained earlier, this process eventually yields an instruction fetched from that memory location.

All slave requests are received by the EXT board, which generates an EX.REQSLV- message for the IOC board. When the CPU acknowledges the slave request, the EXT card receives an IO.ACKSLV- message from the IOC card. The EXT deasserts XP.SCHOD- for one clock cycle and waits for an I/O handshake request (XP.IORQ-) to come from the slave-requesting card. A series of I/O handshakes takes place using the same protocol that is used for handling I/O instructions.

Status LEDs

There are two LEDs reporting the operating status of the EXT board. These LEDs simply monitor the condition of the XP.PON+ line on the extender backplane.

The red LED is lit under any of the following conditions:

a. DC power not within regulation in the extender box.

b. DC power not within regulation in the CPU box.

c. CPU box powered down.

d. CPU box is being manually reset (using RESET switch on the CPU).

e. IOC to EXT cabling damaged or not connected.

The green LED is lit whenever the red LED is off. This indicates that the extender box is in an operating state.

Interrupt Chain Jumper (ICJ) Board Description

The ICJ (Interrupt Chain Jumper) board has only one function. It connects the XP.ICHOD- output from the lowest priority I/O card in the extender box to an unused bus line (used as BP.FETCH- on the A-Series Computer backplane) on the extender backplane. The EXT board receives this signal, synchronizes it to the clock, and passes this information to the IOC board.

The ICJ board allows the extender subsystem to participate in the "diagnose mode" operation of the A/L-Series I/O architecture. Each interface card in the system sequentially informs the CPU of its select code in response to an LIA/LIB 2 instruction. The response starts with the highest interrupt priority card and ripples through the interrupt chain. The ICJ board allows I/O cards in the host to reside below the extender's IOC card.

The ICJ board is not necessary whenever there are no I/O cards in the host below the extender subsystem. If there are two or more extenders occupying the last consecutive slots in the host backplane, only the very last extender may operate without an ICJ board.

There are two versions of the ICJ board. One version (12025-60003) has the same physical shape as the Memory Voltage Jumper card used in the Micro/1000 box. The 12025-60003 ICJ/12 replaces the MVJ board when the Micro/1000 box is used as a 12-slot extender. The second version is the 12025-60004 ICJ/18, which features the standard A/L-Series backplane edge connector. This ICJ may be used in ALL A-Series I/O extender backplanes, but it does reduce the total available I/O slots by one.

Introduction to Detailed Description

Before giving a detailed discussion of the internal operation of the I/O extender, we will discuss the evolution of the A/L-Series I/O architecture into the expanded A-Series I/O architecture, and examine some of the major operational characteristics of the I/O extender when it is utilized with each A-Series CPU. Even though the A600+, A700, and A900 CPUs deal with the same I/O interface cards, each CPU implements the backplane protocol "standard" using slightly different interpretations.

The A/L-Series I/O Architecture

The A-Series I/O architecture is based on the protocols which were developed and implemented on the L-Series CPU. This architecture requires all interface cards to utilize a standard backplane interface handler called the I/O Master. The I/O Master is based on an LSI component known as the IOP (I/O Processor) which manages the direct memory access operation for the interface logic, and interprets every instruction fetched by the CPU.

Instruction interpretation by the IOP causes DMA to suspend temporarily so that the CPU can fetch/broadcast the current instruction. This technique is acceptable for the moderate performance L-Series CPU, but is quite inappropriate for the much higher performance A-Series CPUs. The A-Series I/O architecture differs from the L-Series I/O architecture in the method of broadcasting I/O instructions. For this reason alone, the A-Series I/O extender is incompatible with the L-Series CPU.

Here is a brief explanation of the A-Series versus L-Series I/O differences. The L-Series CPU relies on the backplane signal BP.RNI- to simultaneously fetch and broadcast the instructions. BP.RNI- tells the IOPs to interpret the data on the backplane data bus as an instruction. The A-Series CPUs use BP.FETCH- to fetch the instruction, and if it turns out to be an I/O instruction, the CPU broadcasts the instruction with BP.RNI-. The BP.FETCH- instruction fetch memory cycle is transparent to the IOPs; to them it appears as another DMA cycle. This scheme enhances overall system performance by keeping DMA active unless an I/O instruction is encountered in the instruction stream.

Another A-Series difference is really a consequence of the new BP.FETCH- scheme. The single-step feature of the L-Series is no longer available on the A-Series. Single stepping requires that the IOP which provides the Virtual Control Panel operation monitor the instruction flow by counting BP.RNI- pulses. That IOP halts the CPU after three RNIs, which is three instructions on the L-Series (one for user code and two more to set up the single step), but three I/O instructions on an A-Series CPU.

Since both processor families are bound to the same I/O interface cards, they share the same architectural limitations for I/O. Neither processor family can fully utilize the maximum of 48 addressable I/O cards because of the physical and electrical constraints of the CPU box backplane.

From the physical viewpoint, the largest CPU backplane available, containing 20 card slots, automatically limits the system to less than 20 I/O cards because the CPU/Memory cards share the backplane with the I/O cards. Increasing the number of backplane slots seems to be a possible solution, but this solution has its own electrical problems.

From the electrical viewpoint, the 20-slot backplane is the maximum size supported by the TTL buffers which drive the backplane. Placing more than 20 cards on this backplane would force these buffers to exceed their rated fan-out capability, eventually leading to data loss and possible component failure.

The Expanded A-Series I/O Architecture

For many years, high performance CPUs have been able to solve the propagational delay problems for synchronous systems with a technique called pipelining. The A-Series I/O extender borrows from this technology to control the conflict between maximum backplane throughput and maximum I/O configuration. For most backplane operations not directly involving I/O cards in the extender, there is little or no additional delay for having an I/O extender in the system. The backplane clock cycle time remains the same as before.

When communication is established between the CPU and an I/O card in the extender, additional clock cycles may be required to complete the data/command transfer. Therefore, a time penalty in the form of additional clock cycles is assessed for going into and out of the I/O extender.

As with any pipeline mode logic, special circumstances require that the pipeline be flushed. The I/O extender must guarantee that the CPU backplane be able to accommodate all extender I/O card requests and vice versa. When bus contention develops, the I/O extender must gracefully complete all current activity while suspending the CPU in order to handle the exception cases.

With this background knowledge of the basic A-Series I/O architecture and the enhanced version for the I/O extender, let's examine the differences among the A-Series processors and how they affect the I/O extender.

The A600+ Environment

Instruction Recognition

Instructions are fetched by the CPU using BP.FETCH-, with I/O instructions being broadcast using BP.RNI-. There is at least five clock cycles between BP.FETCH- and its corresponding BP.RNI-.

There are occasions when the A600+ CPU fetches an I/O instruction, but does not broadcast it. This uncertainty prevents the I/O extender from using the CPU-fetched instruction before the CPU broadcasts the same instruction on the host backplane. For example, if the CPU detects an interrupt after fetching an instruction, the processor's microcode directs the machine to service the interrupt and ignore the fetched instruction.

I/O Instruction Processing

The CPU time-out for I/O instructions is approximately six clock cycles, instead of the mandatory three clock cycles. The I/O extender is used with the A600+ processor without any CPU microcode modifications.

Direct Memory Access

Unlike the other A-Series processors, the A600+ departs from the standard DMA arbitration protocol which allows DMA activity to supersede CPU related activities if there is a bus conflict. For example, the CPU must abort its memory access if it detects a simultaneous DMA request from an I/O card.

If the A600+ CPU is ready to perform an operand read/write or an instruction fetch (but not an instruction broadcast), and it senses that DMA is in the bus priority check cycle (first cycle of BP.MRQ- low), the processor steals control of the bus for its own memory access. Consequently DMA must wait until that memory cycle is finished. This feature increases CPU performance at the expense of a slight degradation in DMA latency time.

Interrupts

The A600+ CPU completes the execution of an instruction, fetches a new instruction, then checks the interrupt status. If there are no interrupts pending, it executes the previously fetched new instruction, otherwise the interrupt is processed. Backplane monitors, such as the Logic Analyzer interface card and the I/O extender, are led to believe that interrupts are handled during the execution of an instruction. The I/O extender handles this situation because it uses the fetched instruction only to temporarily suspend DMA in anticipation of an instruction broadcast.

The A700 Environment

Instruction Recognition

The A700 CPU also follows the BP.FETCH- and BP.RNI- sequence for instruction fetching and broadcasting except for instruction execution out of the A- or B-Register. The A700 CPU does not fetch A/B-addressed instructions over the I/O backplane before proceeding to the broadcast. Therefore, I/O instructions in the A/B-register do not allow the I/O extender to arbitrate between the instruction broadcast and DMA.

I/O Instruction Processing

Because the A700 CPU uses the mandatory three cycle time-out for handshakeable I/O instructions, its microcode had to be altered to generate a longer time-out (seven clock cycles) for proper operation with the I/O extender.

All A700 CPUs manufactured with a serial number prefix of 2500 and higher operate properly with the I/O extender without modification. Older A700 CPUs require an upgrade.

Direct Memory Access

Direct memory access with the A700 CPU is performed according to the standard protocol.

Interrupts

Other A-Series processors acknowledge I/O interrupts by asserting BP.IAK- at the start of the long half cycle. This means that BP.IAK- must be arbitrated with the potential assertion of BP.MRQ- by any I/O card desiring a DMA cycle. The A700 CPU departs from this interrupt protocol standard by asserting its BP.IAK- at the start of the short half cycle.

The A900 Environment

Instruction Recognition

Unlike the A600+ and A700 processors, the A900 CPU does not access memory over the I/O backplane. Therefore, the A900 processor generally ignores any activity over the I/O backplane unless it needs to interact with the I/O cards. The CPU does not use BP.FETCH-; only BP.RNI- is used to indicate that an instruction is present on the data bus. Because there exists only one opportunity to interpret and to respond to the broadcast I/O instruction, the I/O extender must guarantee that it is not trying to process a DMA request from an I/O card in the extender while the CPU is broadcasting the I/O instruction. Such a situation would result in a deadlock condition.

When used with the A900 CPU, the I/O extender assumes control of the CPU backplane until the processor makes a bus request. The I/O extender uses BP.MRQ- to keep the CPU off the backplane. The CPU makes a bus request by asserting BP.CPUTURN-. After all current and pending DMA requests have been serviced, the I/O extender releases BP.MRQ- so that the CPU can proceed with the instruction broadcast. By manipulating BP.MRQ- and using BP.CPUTURN-, the I/O extender can guarantee that current DMA requests finish and new DMA requests suspend, before an I/O instruction is broadcast over the CPU backplane.

Only A900 CPUs with a serial number prefix of 2500 or higher operate properly with the I/O extender (older A900 CPUs may be upgraded.) The older A900 Cache board is unable to recover from a BP.MRQ- holdoff when it needs to broadcast an I/O instruction; it treats the instruction broadcast as a DMA read request and therefore the I/O cards never see the I/O instruction. The microcode upgrade is necessary to add a few extra cycles to the I/O instruction time-out to cover the I/O handshake overhead incurred by using the I/O extender, as explained in the next paragraph.

I/O Instruction Processing

Once the A900 CPU has broadcast the I/O instruction, it waits up to at least six backplane clock cycles for a BP.IORQ- response from an I/O card before timing-out. Older A900 CPUs time-out after three backplane cycles, and therefore require the upgrade previously mentioned.

Direct Memory Access

For DMA writes into memory, the A900 CPU performs a different handshaking protocol than the A600+ and A700 processors. If a refresh cycle coincides with a memory cycle, the A900 responds to the memory request with the simultaneous assertion of BP.BUSY- and BP.VALID-. BP.VALID- is released one cycle later, but BP.BUSY- remains asserted until the concurrent refresh cycle is also finished. The original L-Series I/O architecture, as implemented in the A600+ and A700, calls for a BP.BUSY- assertion at the start of the memory cycle, a BP.VALID- assertion at the start of the last cycle of BP.BUSY-, and the deassertion of both signals one cycle after BP.VALID- is asserted.

Even though the A900's unique trait does not affect the I/O cards, the I/O extender must be aware of these protocol differences in order to monitor CPU activity successfully. The I/O extender incorporates a protocol look-ahead feature in the DMA state machine to anticipate handshaking sequences to speed up DMA operations.

Interrupts

Interrupt processing with the A900 CPU is performed according to the standard protocol.

Extender Architectural Characteristics

Some of the more significant architectural limitations of the extended A-Series I/O system are discussed in the following paragraphs.

Maximum Select Codes

The maximum number of I/O cards that can be configured into any A-Series computer is 48. Although the instruction set calls for a system total of 64 select codes (six bits in the op code), the lowest 16 select codes are used/reserved for CPU and system level I/O operations. Included in the list of system level I/O activities is the interrupt mask, LEDs and configuration switches on the CPU board, Global Register, time base generator, and the parity/ECC address registers. Any number of I/O extenders may be configured to a single CPU, but the total number of I/O cards in all extenders and the CPU backplane may not exceed 48.

Cannot Have Only I/O Extenders on CPU Backplane

The I/O extender uses the principle known as pipelining. Pipelining is used to break up a large I/O system into smaller domains, with each domain controlled by a dedicated I/O extender. Activity originating in the CPU backplane begins on the extender backplane two cycles later. Any responses from I/O cards in the extender backplane likewise appear on the CPU backplane two cycles later. Because of this four cycle overhead, the I/O cards in the I/O extender cannot participate in certain global I/O handshaking operations. The I/O instructions OT* 0 and OT* 2 comprise the special-I/O cases which force the I/O extender to passively observe the I/O handshake, yet retrieve the data for the I/O cards in the extender. It is therefore necessary for at least one real I/O card (I/O Master equipped) to reside in the CPU backplane to provide the mandatory I/O handshaking for these two I/O instructions.

Custom Designed I/O Cards

Custom designed I/O cards also work in the extender if the I/O card's backplane interface is based on the I/O Master or if that card can generate XP.IORQ- within 275 nsec from the rising edge of XP.SCLK- which deasserts XP.RNI-. The I/O extender samples XP.IORQ- at 1.4 clock cycles after the deassertion of XP.RNI-. If an I/O card generates XP.IORQ- too soon, it is ignored for awhile because the standard protocol allows for up to 325 nsec of settling time (from data bus valid; on the extender backplane, the data bus is valid immediately after the assertion of XP.RNI-) where there may be false assertions of XP.IORQ-. For slow I/O cards which require more than 1.4 clock cycles to generate XP.IORQ-, the I/O extender still passes the I/O handshake request to the CPU backplane, but a handshake may not be granted by the CPU if it has timed-out of the I/O handshake wait loop.

Why I/O Extenders Cannot Have I/O Extenders

I/O extenders may not contain other I/O extenders. The second I/O extender cannot generate an IORQ- signal fast enough (refer to previous paragraph) for the first I/O extender to pass to the CPU. The CPU's microcode can be modified to accommodate two or three levels of I/O extenders at the expense of increasing the I/O handshake time-out. However, it is more efficient to place several I/O extenders in parallel over the CPU backplane. Since all propagational delays occur in parallel, having two or more I/O extenders does not increase handshaking overhead over having only one I/O extender.

I/O Instruction Throughput

There are two types of overhead associated with I/O instruction broadcasting. The first type of overhead penalizes all broadcast I/O instructions. The I/O extender pauses the CPU for two cycles between the fetch of an I/O instruction (for A900, the assertion of BP.CPUTURN-) and the broadcast of that I/O instruction. The A600+ CPU is not hindered by this pause because it normally uses at least five cycles (varies from one instruction to the next) to get from the fetch of an I/O instruction to the broadcast of that instruction.

The second type of overhead for an I/O instruction is only assessed if the instruction's expected handshake times-out or if the handshake is with an I/O card in the extender. The additional time used for a handshake which times-out is dependent on processor type. It is 0 for the A600+, and 3 cycles for both the A700 and A900 CPUs. These figures show that the A700 and the A900 CPUs with the new base set microcode process I/O instructions slower than before, even if the I/O extender is not used. For all processors, the I/O extender's BP.IORQ- latency (I/O extender samples XP.IORQ- 1.4 cycles after the deassertion of XP.RNI-) is four cycles. All handshaken I/O instructions executed by an I/O card in the I/O extender require four additional clock cycles to complete. This overhead applies to single and double handshakes; multiple handshake operations (HLT and slave handshakes) incur some slight additional delays between pairs of handshakes.

DMA Throughput

From the perspective of an I/O card in the extender box, some DMA cycles appear to take an unusually long time to complete. The I/O extender treats DMA reads (from memory) and DMA writes (into memory) differently to take advantage of the pipelining capabilities built into the enhanced I/O architecture.

The nature of DMA reads requires that data be extracted from memory and sent back to the requesting I/O card. Pipelining cannot be used to start another DMA cycle while the current DMA read is in progress. DMA reads require a minimum of five cycles to complete. One cycle apiece is required to transfer data from I/O card to the extender, between extender control cards via the cable, and from the CPU backplane back to the extender backplane. Only two cycles (minimum) is actually spent obtaining the data over the host processor box backplane.

DMA writes into memory are pipelined. As soon as the current DMA address and data is passed to the next pipeline stage, another address and data set is acquired, if available. For I/O cards in the extender performing DMA write operations, they typically complete the transfer in two cycles. More than two cycles are needed if the DMA write occurs while the pipeline queue is full and not advancing.

The I/O extender's maximum achievable transfer rate depends on the actual mix of DMA reads and DMA writes. Pure DMA writes may occur at the full bandwidth of the host's DMA bandwidth while pure DMA reads can only attain 40 percent of that bandwidth (50 percent of host DMA read bandwidth if A900 CPU). In actual practice, the aggregate achievable bandwidth is between 40 and 100 percent of the host DMA bandwidth. For example, alternating DMA reads with DMA writes yields a maximum transfer rate equal to 57% of the CPU backplane DMA bandwidth. In this particular case, with the extender backplane running at its top speed, the CPU backplane still has a 43% reserve bandwidth for the CPU (A600+/A700) to work with.

Interrupts and DMA Interaction

A small side effect of the extender's pipeline mode of operation is the time shift incurred when moving from the CPU backplane to the extender backplane and vice versa. All CPU access to its backplane is arbitrated by the I/O extender to guarantee that no deadlock situations arise. One such case might be the simultaneous bus requests made by an I/O card in the extender to perform DMA and the CPU wanting to acknowledge an interrupt request initiated by another I/O card in the same extender. The DMA request, once initiated, must be carried through to completion. By the time that DMA request makes its way to the CPU backplane, it cannot continue without confusing the CPU into thinking that a trap cell fetch for the interrupt acknowledge has just been performed. By the same token, neither can the interrupt acknowledge be passed onto the extender backplane. Any response to an interrupt acknowledge on the extender backplane would invariably destroy any pending DMA request information residing in a holding register in the I/O extender.

The I/O extender prevents this situation by guaranteeing that all DMA activity on the extender backplane is suspended whenever an I/O card in the extender makes an interrupt request. After all interrupt requests have been serviced, DMA is allowed to continue from where it left off. The typical duration of a DMA shutdown for interrupt handling is generally in the order of several tens of microseconds. A potential benefit from using this technique is that interrupts may be serviced more quickly because of reduced DMA activity (DMA has priority over interrupt acknowledgement except for the A700 CPU) on the host backplane.

Maximum Number of Concurrent DMA Channels

Although it is possible for the hardware to support simultaneous DMA activity from 48 interface cards, the RTE-A operating system never permits more than 24 DMA channels to be active simultaneously. The original A-Series memory management architecture established 32 map sets for memory mapping functions. RTE-A reserved 8 maps for system utilization, and left the remaining 24 maps for the DMA channels. This plan permitted each I/O card to own a dedicated map for DMA because the number of I/O channels could never exceed the number of available maps.

With the I/O extender this static scheme is inadequate. The number of available maps could not be increased because the address extension bus (used by the DMA channel to select the DMA map) is only five bits wide, and can only be used to select one of 32 maps. With only 24 maps and up to 48 DMA channels, RTE-A (with A.85 and later revisions) uses a map allocation scheme to distribute the 24 maps to up to 48 DMA channels. Important DMA channels can be assigned dedicated maps while all other DMA channels must share.

Document Reading Guide

Reading the Theory of Operation

Signal naming conventions are used to clarify the origin of signals without having to refer to the schematic diagrams. Two or three character prefixes, followed by a period, precede the signal name. As a signal passes from a board to a cable then to another board, the signal name remains the same but its prefix changes.

The following prefixes are used in the manual text, and in the schematic and timing diagrams:

BP.xxxxxx	signal on the CPU box backplane
XP.xxxxxx	signal on the extender box backplane
IOC.xxxxxx	signal on the IOC board (12025-60001 PCA)
EXT.xxxxxx	signal on the EXT board (12025-60002 PCA)
IO.xxxxxx	signal on the (12025-60007) cable, originating from IOC
EX.xxxxxx	signal on the (12025-60007) cable, originating from EXT

The schematic diagram location of integrated circuits (ICs) is given by a two character prefix followed by a four digit row/column grid reference:

IOrrcc	IC located on the IOC board at row rr and column cc
EXrrcc	IC located on the EXT board at row rr and column cc

Miscellaneous components are identified by a reference designation:

Rxx	resistor
Cxx	capacitor
CRx	diode or light-emitting diode (LED)

Printed circuit board edge connectors are identified by a two character connector reference followed by a dash which is followed by pin count number. A supplemental single character may be used to indicate either the component side of the board or its circuit side :

P1-xx	one of 50 pins on backplane connector P1
P2-xx	one of 50 pins on backplane connector P2
J1-xxB	one of 40 pins on frontplane connector J1, component side
J1-xxA	one of 40 pins on frontplane connector J1, circuit side

Note that odd pin numbers on connectors P1 and P2 are located on the component side.

During the discussion of specific circuit functions, it is necessary to make reference to various parts used. These references are of the form [U-number]@[schematic location], as in I00805@19A, which means that the IC at U-number U0805 on the IOC board can be found at grid location 19A on the IOC schematic diagrams. Use the chart in the following paragraph to associate the schematic grid numbers to the figure number.

Reading the Schematic Diagrams

The schematic diagrams for the 12025-60001 (IOC) and 12025-60002 (EXT) boards are Figures 12-12 and 12-13, and are organized similarly.

Schematic Grid	Figure Number	Function
IOC board		
10 through 19	12-12, Sheet 1	Control state machines
20 through 29	12-12, Sheet 2	Address bus
30 through 39	12-12, Sheet 3	Data bus
40 through 49	12-12, Sheet 4	Spare gates/miscellaneous parts
EXT board		
10 through 19	12-13, Sheet 1	Control state machines
20 through 29	12-13, Sheet 2	Address bus
30 through 39	12-13, Sheet 3	Data bus
40 through 49	12-13, Sheet 4	Spare gates/miscellaneous parts

Reading the Timing Diagrams

Figures 12-1 through 12-10 are timing diagrams that are provided to assist you in understanding the operation of the I/O extender. All timing diagrams show the overall operation, relating activities on the extender backplane to those on the CPU backplane, by way of the EXT and IOC.

Clock Signals

The I/O extender provides two global clocks on the extender backplane. One clock, XP.SCLK-, is a replica of the CPU backplane system clock. A communications clock called XP.CCLK-, is also supplied for those I/O cards with asynchronous communication ports for their baud rate generators.

System Clock

A system clock, named BP.SCLK-, is maintained by the CPU for the synchronous operation of all backplane handshaking protocols. The A600+ and A700 CPUs use system clocks with a 40% high duty cycle and a frequency of 4.4 MHz and 4.0 MHz respectively. The A900 CPU I/O clock has a 37% high duty cycle and a frequency of 3.75 MHz. The I/O clock for the A900 is actually half the frequency of the processor's internal clock rate of 7.5 MHz. The backplane system clock is terminated with a pull-up/pull-down resistor network near the lowest priority I/O slot in the CPU backplane.

The IOC card receives BP.SCLK- with an inverting buffer [IO1104@24D] to generate the IOC.BCLK+ board clock used to clock many of the flip-flops on the IOC board. IOC.BCLK+ is received by another set of two inverting buffers [IO1104@24D] to generate IOC.BCLK- and IO.CLOCK-. IOC.BCLK- is also used as the clock inputs to flip-flops on the IOC board. IO.CLOCK- is the clock signal passed to the EXT board through the frontplane connector cable.

The EXT card receives IO.CLOCK- from the front plane connector with an inverting buffer [EX0604@21E]. A resistor network provides termination for the IOC board clock driver to reduce signal ringing. If the connecting cable is not installed, the resistor network sets the IO.CLOCK- signal to 3.4 volts, preventing a floating input line condition to the buffer. The buffered clock is passed through another inverting buffer [EX0807@23E] to create the extender backplane system clock, XP.SCLK-. This clock has a pull-up/pull-down termination near the lowest priority card slot on the extender backplane.

To keep all EXT clock skews referenced to the backplane clock, the EXT card receives XP.SCLK- into an inverting buffer [EX0807@23E] to obtain the EXT.BCLK+ board clock, and routes that into another inverting buffer [EX0807@23E] to generate the inverted board clock EXT.BCLK-.

Communications Clock

Although the CPU backplane carries the BP.CCLK- communications clock signal, the I/O extender generates it own version to avoid driving the cable with another high frequency signal. This is possible because of the completely asynchronous nature of the communications clock. It runs at 14.7456 MHz and has a nominal duty cycle of 50%. The EXT card uses a crystal oscillator [EX1207@24E] and an output inverting buffer [EX1107@25E] to provide XP.CCLK- for the extender backplane. A resistor network terminates the XP.CCLK- signal near the lowest priority card slot on the extender backplane.

What Happens at Power Up and Power Down

The I/O extender is designed to allow either the CPU box or the extender box to be powered up first. The power sequencing logic of the I/O extender guarantees proper start up for the entire system independent of power on sequence.

PON Initialization

All synchronous logic on the IOC and EXT boards is properly initialized at power on using the power supply control signal BP.PON+ and XP.PON+.

The IOC board receives BP.PON+ with an inverter/buffer [IO1104@24D] to clean up the signal. A flip-flop [IO0804@25E] synchronizes the buffered IOC.BPON- to the start of the long half cycle, thus guaranteeing that the initialization pulse always ends on a clock edge. All of the registered PALs (programmable array logic) receive the initialization signal as input data which must meet the applicable setup and hold times for those parts. Finally, the synchronized initialization pulse is inverted and buffered [IO1104@24D] to become the generalized reset signal IOC.BPONL+. All flip-flops (except the PON synchronization flip-flop) are cleared to their default states when IOC.BPONL+ is low. At power-up, IOC.BPONL+ is low and goes high only after the power supply asserts BP.PON+ to signal that all supply voltages are in regulation.

At the EXT board, XP.PON+ receives a similar treatment before being used as EXT.BPONL+. XP.PON+ is buffered [EX1107@22D], synchronized to the start of the long half cycle (referenced to XP.SCLK-) [EX0706@24D], and buffered again [EX0906@25D]. As is the case on the IOC, EXT.BPONL+ is primarily used to set EXT board flip-flops to their default states.

All of the A-Series processors are equipped with a reset switch which pulses BP.PON+ to simulate a power-up condition. The I/O extender should also be reset if the CPU backplane is reset. This feature is supported. IOC.BPONL+ is inverted and buffered [IO1204@19D], and becomes IO.IOCPON- on the I/O extender cable. The EXT card receives IO.IOCPON- with a 74S240 [EX0704@11D] to invert and clean up the signal. It undergoes another level of inversion [EX0404@12D] before reaching the backplane buffer, an open-collector 74S38 [EX0907@20D] to drive XP.PON+. The merged XP.PON+ is the method used to reset all cards in the backplane when the CPU is reset or powered up.

Note that the EXT card receives IO.IOCPON- with a pull-up/pull-down resistor pair. During normal operation, these resistors can be ignored. When the CPU box is powered down or when the I/O extender cable is removed, these resistors provide a known signal level of approximately 3.4 volts for the input buffers. This voltage level is interpreted to mean that the CPU box is not ready for the I/O extender, and the EXT treats this as a reset condition.

Extender Box Power Transitions

TTL (transistor-transistor logic) device outputs exhibit erratic voltage transitions during power up and power down. If power is transitional in the extender box while XP.PON+ is low, no negative effect is experienced by cards in the extender box because of the reset state of the extender. The same is true for the host CPU box. While BP.PON+ is low on the CPU backplane, the CPU and the I/O system is in a reset state. However, when two or more boxes are interconnected, the interface circuitry between (or among) them must account for the possibility of one box powering down while the others remain powered up.

Because each I/O extender is unaware of the presence of another extender, the analysis is reduced to treating the second extender as a copy of the first extender. Between an extender and a CPU box, there are four power transition environments:

CPU BOX	EXTENDER BOX	COMMENTS
Power good	Power good	Normal operating environment
Power good	Powering up/down	Extender box power transition
Powering up/down	Power good	CPU box power transition
powering up/down	powering up/down	System power transition

For the third and fourth cases, the power transitions in the CPU box cause the IOC control lines to glitch. Since the entire system (CPU, I/O cards, and I/O extender) is in a reset state because BP.PON+ is low, these glitches are harmless. Case number 1 is not a power transition case.

Only case number 2 exhibits the glitching problem attributed to power transitions. Since XP.PON+ is low when the extender box power is unreliable, glitches generated and received in the extender box (between I/O cards and the EXT) are not a problem. Glitches passed from the EXT to the still-functioning IOC are a problem. Any signal transition on the addressing and data bus lines to the IOC are insignificant unless the IOC loads the information off of the cable and performs a data transfer to the CPU or memory system. The IOC does not do so unless the EXT control lines make a data transfer command.

The EXT control lines are susceptible to signal glitches when power is not within regulation on the extender backplane. If these control line glitches are received by the IOC backplane handshaking state machines, an erroneous data transfer would result. This situation is avoided by putting the IOC backplane handshaking state machines to sleep whenever the IOC detects that the EXT card has lost power. IOC.EXTPONB+ (explained in more detail in the next section) is the IOC version of the extender box XP.PON+ signal. IOC.EXTPONB+ is sent through a D-type flip-flop [IO0806@12B] to synchronize the signal to the start of the long half cycle. The resulting signal, IOC.IOPON+, represents the combined effects of IOC.BPONL+ (through the flip-flop reset input) and IOC.EXTPONB+ (through the flip-flop data input). If either IOC.BPONL+ or IOC.EXTPONB+ goes low, IOC.IOPON+ also goes low to indicate that either the CPU box or extender box is powered down.

IOC.IOPON+ is utilized by the IOCP1 DMA Control PAL [IO0905@15A], the IOCP2 I/O Handshake PAL [IO1004@15B], and the Slave Mode Flag [IO0504@14E]. For IOC.IOPON+ low, these devices are forced to their defaulted inactive states. Therefore, any control line glitches received by the IOC after XP.PON+ goes low do not cause an erroneous DMA handshake, I/O handshake, interrupt request, or slave request on the CPU backplane. The same circuitry is used to prevent extraneous activity resulting from EXT power up glitches.

LED Status Indicators

The I/O extender is designed so that simple (and usually common) errors in configuration and power up can be detected and corrected quickly. A set of green and red LEDs are provided on the IOC and EXT boards to assist the installer.

Beginning with the simpler case on the EXT board, two open-collector inverter/buffers [EX0104@11E] drive the two LEDs, CR1 (red) and CR2 (green), using the EXT.BPON- and EXT.BPON+ power status signals, respectively. EXT.BPON- is the inverted/buffered version of the backplane XP.PON+, and EXT.BPON+ is the inverse of EXT.BPON- processed through an inverter [EX1206@25D]. The red LED is lit when XP.PON+ is low, while the green LED is lit when XP.PON+ is high. Both LEDs cannot be lit simultaneously. It is possible for either LED to glitch at power up because XP.PON+ may glitch when power is first applied to the inverter/buffer [EX0907@20D] reset driver. All TTL parts are susceptible to output glitching during power up.

On the IOC board, an open-collector 4-bit to 10-line decoder [IO0304@13E] interprets two input signals to determine which LED to drive, CR1 (red) or CR2 (green). The two input lines, IOC.EXTNRDY- and IOC.EXTPONB+ decode into four possible combinations. The green LED is lit only if IOC.EXTNRDY- is high (extender is ready) and IOC.EXTPONB+ is high (DC power in extender box OK). All other combinations cause the green LED to extinguish and the red LED to light. There are no situations which cause both LEDs to be on at the same time.

IOC.EXTPONB+ is the output of an inverting buffer [I00704@12D] receiving EX.EXTPON-. If XP.PON+ on the extender backplane ever goes low, EX.EXTPON- subsequently goes high to indicate that power has gone down in the extender box and that the extender box is now "off-line". Note that EX.EXTPON- is terminated at the IOC board with pull-up/pull-down resistors. If the extender cable is not properly installed on either the IOC or EXT end, these resistors default the extender to a powered-down, "off-line" case. Whenever EX.EXTPON- is high on the cable, the IOC board has its red LED on.

IOC.EXTNRDY- is the output of a J-K flip-flop [I00504@12E] whose J-input is IOC.EXTPONB+. The ExtNrdy flip-flop is set (IOC.EXTNRDY- goes low) with IOC.BPONL+ at initial power up or upon a CPU hardware reset. The purpose of this flag flip-flop is to remember if the extender box has ever powered up correctly in the past. If the extender box was turned on before the CPU box, IOC.EXTNRDY- makes a transition from its default low state to the high state one cycle after IOC.BPONL+ has gone high (IOC.EXTPONB+ already high at CPU box power up). If the CPU box is turned on with the extender box off, the IOC.EXTNRDY- flip-flop output stays low until the extender box is powered up.

Processor Freeze at Power Up

Besides its LED application, the IOC.EXTNRDY- flag causes the CPU to stay off the host processor backplane until the extender box is ready to communicate with the host. A gate [I00707@18A] inverts the signal for the open-collector backplane driver [I01107@19A] for BP.MRQ-. Whenever IOC.EXTNRDY- is low, BP.MRQ- is also low, causing the processor to be held off the backplane. This scheme causes the CPU to suspend execution of any backplane activity until the extender box is ready.

Because the A600+ and A700 CPUs need to acquire pretest instructions using the backplane data bus, these processors freeze at the completion of their microcoded self-tests. The A900 obtains its instructions over a dedicated CPU-memory backplane, so it does not freeze until it attempts to broadcast an I/O instruction over the CPU-I/O backplane. BP.MRQ- keeps the CPU on hold until the extender box is available, and serves to synchronize the two boxes independent of their power up order and of the processor type involved.

Power Fail Auto Restart Features

Power fail auto restart recovery is provided to permit a graceful shutdown of the I/O system in the event that power in the I/O extender box is interrupted. Without these features, the operating system would be completely unaware that all interface cards in the extender box have been taken "off-line" by a powered-down I/O extender. If the I/O extender was performing DMA or an I/O handshake at the moment of the power failure, any activity with the I/O cards in the I/O extender box may terminate abnormally, potentially hanging the CPU backplane with it.

Power Fail

The EXT card utilizes non-TTL devices for detecting the power fail and powered-down conditions because these circuits must operate reliably below TTL supply voltage levels. XP.PON+ and XP.PFW- have pull-up resistors [R9@21D and R8@20D, respectively] to guarantee a good logic-one level when the open-collector line drivers in the power supply are in their cutoff state. A pull-down resistor was avoided in order to provide as much noise margin as possible for the receiving circuitry. Both of the power supply status signals are received by stabistor diodes [CR4@21D and CR3@21D, respectively] which drive the base of NPN transistors [EX0407 transistor array DIP @22D]. These stabistor diodes are rated with a forward biased voltage drop of 1.85 volts minimum to 2.05 volts maximum. If the base-to-emitter voltage drop is 0.7 volts, then XP.PON+ and XP.PFW- need to be greater than 2.7 volts before the transistors turn on.

These transistors directly drive the cable signals EX.EXTPON- and EX.EXTPFW+. Each signal is terminated on the IOC board with a pull-up/pull-down resistor pair [IO0702@10E/10C, respectively] providing the open-collector transistor drivers with some collector resistance. These terminators also provide a default signal level when the extender box is not powered up or when the extender cable is disconnected. For both situations, the logic one default indicates that the extender box is signaling a power fail warning as well as a power down condition.

EX.EXTPFW+ is received into an inverting buffer [IO0904@12C], creating the signal IOC.EXTPWRFAIL-. This signal is immediately inverted by a gate [IO0404@13D] to generate IOC.EXTPWRFAIL+, which is received by an open-collector NAND gate buffer [IO1107@19E]. When properly qualified by IOC.EXTNRDY-, this buffer drives the CPU backplane power supply status signal BP.PFW- to inform the CPU that there is an impending power failure somewhere in the system. This technique allows existing software to purposely terminate activity in the entire system so that power-failed extenders can lose power without concern for lost data transfers.

The BP.PFW- driver on the IOC board is qualified with IOC.EXTNRDY- to avoid a power fail warning status when the extender box is powering up. The backplane protocols require BP.PFW- to be high before BP.PON+ goes high. If the extender is powered on before the CPU box is turned on, there is no problem guaranteeing that sequence. However, the reverse case is more difficult to control. A powered down extender forces the cable default levels to be used; the default for EX.EXTPFW+ is true, indicating a power failure. IOC.EXTNRDY- is low whenever the extender box has not yet powered up, making that status flag a natural choice for qualifying IOC.EXTPWRFAIL+ in generating BP.PFW-. With this circuit, power fail warnings are issued to the CPU only at extender power down, never at power up.

Power Up Auto Restart

Once the processor has been alerted to the power fail warning, it shuts down the I/O system and saves the state of the machine in battery-powered memory. At this point, one of two events may occur. The CPU may continue to be powered up, or it may also lose power.

The first case involves a downed extender and an operating CPU. The functioning IOC card prevents the CPU from doing anything useful with the continued assertion of BP.PFW- for as long as the extender box is powered off. This is considered a brown-out situation for the CPU because it is still powered up but waiting for an impending power shutdown that may not occur in the main box. When line power is restored to the extender box, XP.PFW- goes high before XP.PON+ goes high. This means that IOC.EXTPWRFAIL+ goes low before IOC.EXTPONB+ goes high. A gate [IO0306@13D] detects this situation and sets the signal IOC.EXTDOWN+ high. As the K-input to the IOC.EXTNRDY- J-K flip-flop [IO0504@12E], IOC.EXTDOWN+ causes IOC.EXTNRDY- to be true again. This action causes the BP.PFW- power fail warning status to be removed and simultaneously imposes a BP.MRQ- hold on the CPU. As explained earlier, IOC.EXTNRDY- keeps BP.MRQ- asserted until XP.PON+ has gone high on the extender backplane.

The second case involves powered down extender and CPU. This is known as the black-out case, and its power up scenario is identical to the initial power up sequence for which either box may power up first. If the extender comes up first, everything will be considered "on-line" by the time the CPU box is powered up. For the reverse case, the IOC.EXTNRDY- status signal is initialized to a low state at CPU box power-up. This serves to keep the CPU off the backplane with BP.MRQ- and also disqualifies the defaulted IOC.EXTPWRFAIL+ signal.

There is a subtle difference between the brown-out case and the black-out case. The brown-out case merely causes a suspension of the operating system code whereas the black-out case forces the execution of the microcode and firmware self-tests before returning to the operating system code. In the latter case, a backplane status signal called BP.MLOST- is tested by the firmware self-test to determine if control of the CPU is to be returned to the program (operating system) in battery backed-up memory, to the Virtual Control Panel, or to a selected boot device as directed by the processor BOOT SEL switches.

Maintaining MLOST Status

BP.MLOST- is a power supply/battery backup subsystem status signal that is only valid during the first 5 msec after the power supply has allowed BP.PON+ to go high. The default state of this open-collector-driven signal is high, owing to a pull-up resistor located on the CPU. Its value is read as part of the LI* 1 (load accumulator with processor BOOT SEL switch setting) instruction. The various A-Series processors utilize two methods to guarantee that they sample BP.MLOST- while it is valid. Since the A700 and A900 CPUs have lengthy microcoded self-tests, these processors sample BP.MLOST- and the BOOT SEL switches during the self-test and retain the information in an internal register. An LI* 1 assembly language instruction references the internal version of the switch register rather than the physical switches. The A600+ CPU has a simplified microcoded self-test, so it can forego the more elaborate procedure by passing control to the firmware-based self-test, which uses assembly language instructions to access the physical switch register and BP.MLOST-.

The narrow window for a valid BP.MLOST- means that the A600+ CPU does not work with the I/O extender under certain circumstances. If the installed battery backup subsystem has been depleted, the battery backup controller drives BP.MLOST- low for the first 5 msec after BP.PON+ has gone high. Assume the extender box is powered off. The A600+ CPU completes its microcoded self-test and then is denied access to the firmware-based self-test (which contains the LI* 1 instruction). After 5 msec, BP.MLOST- reverts to its default state. Some time later, the extender box powers up and the CPU is allowed to continue processing. When LI* 1 is finally executed, the CPU is told that battery power has preserved the memory-based program, even though that is not true. The firmware-based self-test eventually detects this fault, and reports it as a self-test error. The A700 and A900 CPUs are unaffected by the inclusion of the I/O extender in the system.

The IOC card contains circuitry to extend BP.MLOST- low for at least 950 msec after the release of BP.MRQ- to allow the CPU to enter the firmware-based self-test. The MLOST- extend circuit is based on a portion of the IOCP3 PAL [IO1105@15C] and a programmable counter IC. BP.MLOST- is received into an inverting buffer [IO0704@12D] to create IOC.MLOSTB+ for the IOCP3 PAL.

The IOCP3 PAL maintains a permanent version of the quickly disappearing BP.MLOST- signal.

```
--- PAL equation for IOC.MLOSTFF+
/MLOSTFF := /BPONL                        ;MLOSTFF+=0 when BPONL+=0
          + /MLOSTFF*/MLOSTB*BPONL        ;after BPONL+=1, and
                                          ; MLOSTB+=1, then
                                          ; make MLOSTFF+=1 forever
          + /MLOSTFF*MODESEL*BPONL        ;after BPONL+=1, and
                                          ; MLOSTB+=0, and
                                          ; MODESEL-=0, then
                                          ; keep MLOSTFF+=0 forever
```

Since registered PALs normally generate low-true outputs, it is necessary to apply De Morgan's theorems to the Boolean equations to arrive at high-true outputs. Before power is valid in the CPU box, IOC.MLOSTFF+ is initialized to a low output as indicated in product term 1. Product terms 2 and 3 provide the means to keep IOC.MLOSTFF+ low.

If BP.MLOST- remains high after BP.PON+ has gone high, this keeps IOC.MLOSTB+ low, and serves to keep IOC.MLOSTFF+ low. IOC.MLOSTB+ is sampled up to the moment that IOC.MODESEL- becomes true; if IOC.MLOSTFF+ is still low at that time, IOC.MODESEL- takes over and keeps IOC.MLOSTFF+ low permanently.

IF BP.MLOST- is low for 5 msec beyond BP.PON+ going high, IOC.MLOSTB+ is high for that period of time. According to product term 2, this condition forces the IOC.MLOSTFF+ flip-flop to go high. There are no product terms in the IOC.MLOSTFF+ PAL equation for a return to a low output, so IOC.MLOSTFF+ remains high unless the CPU is reset or power cycled.

The IOC drives its version of BP.MLOST- with a NAND gate open-collector buffer [IO0104@23E]. When IOC.MLOSTFF+ is low, the buffer is left inactive. When IOC.MLOSTFF+ is high, the buffer drives BP.MLOST- until the MLOST time-out counter finishes counting 4194304 ((2^23)/2) IOC.BCLK- cycles after IOC.EXTNRDY- goes high.

The MLOST time-out counter is implemented as a single 32-bit programmable counter IC [IO0105@21E]. That device has two clock inputs, and a low-to-high transition on either input advances the count by one. Counting is inhibited whenever either input is at a high level. A low IOC.EXTNRDY- causes the counter to remain in reset, and counting begins from zero after IOC.EXTNRDY- goes high. The count is hard-wired for a divide by 2^23 (2 to the 23rd power), so the Q-output goes high when half that count is reached. The output goes to the counter's second clock input to disable further counting, and also to an inverter [IO0404@22E] whose output disables the IOC's BP.MLOST- driver.

No Battery Backup in the Extender Box

Although the 12154A/12157A Battery Backup Modules may be used to provide +5VM (memory voltage) for special I/O cards in the 12025A/12025B I/O Extenders, respectively, the feature is not supportable in all situations. If the backup module is able to sustain memory voltage throughout a power outage, everything works as planned. The operating system auto restarts and continues operation.

However, should the extender battery backup module be depleted during a power outage while the CPU's battery backup continues to sustain memory, there is no way to alert the operating system that backup power failed in the extender box. When the operating system restarts after the power outage, it assumes that I/O interfaces requiring battery backup power would operate properly, continuing from the point of power failure. This assumption is based on the idea that if the operating system survived the power failure, then all I/O cards which require battery backup power should have also maintained their volatile information. Of course, that assumption is false for this case.

Using the battery backup modules in the extender box to maintain memory voltages for special I/O interfaces is hardware compatible, but the software may not be tolerant to selective I/O cards losing backup power. The best solution is to place all I/O cards requiring the +5VM memory voltage in the CPU box.

The Processor Mode Flag

The I/O extender maintains a processor mode flag for the benefit of the processor arbitration circuitry. The arbitration logic behaves differently for the A600+/A700 CPUs than for the A900 CPU because the two classes of processors utilize the processor-I/O backplane differently. The A600+ and A700 CPUs rely on the backplane for all of their memory needs whereas the A900 CPU bypasses the backplane by directly interfacing with cache memory, which directly interfaces with the memory controller.

The processor type can be determined automatically because the A600+/A700 CPUs always use the BP.FETCH- before BP.RNI-, whereas the A900 CPU only uses BP.RNI-.

The IOCP4 PAL [IO1005@15D] uses two flip-flops to implement the processor mode flag. The IOC.MODESEL- flip-flop indicates that the IOC.A900MODE- flip-flop is valid. When IOC.A900MODE- is true (low), the I/O extender has determined that it is communicating with an A900 host CPU.

```
--- PAL equation for IOC.MODESEL-
MODESEL := /MODESEL*QFETCHL*BPONL   ;MODESEL-=0 when QFETCHL+=1 or
         + /MODESEL*QRNIL*BPONL     ; QRNIL+=1, whichever is first
         +  MODESEL*BPONL           ;remember that forever (almost)
```

IOC.QFETCHL+ and IOC.QRNIL+ are both outputs of a multiplexer [IO0706@13A] which has been wired to perform as a complex AND gate. The multiplexer's data select address lines are connected to IOC.MP-. IOC.MP- is an inverted/buffered version of the backplane BP.MP+ signal which indicates the current status of the memory protection system. If the memory protection system is enabled, all I/O instructions must be ignored by the I/O cards. If memory protect is turned on, the multiplexer selects the grounded IOA and IOB inputs. IOC.QFETCH+ and IOC.QRNIL+ remain low, inhibiting the I/O extender from interpreting any instructions. The EA and EB inputs on the multiplexer enables the two outputs, allowing the selected data input lines to pass through to the outputs. The enable is IOC.VALIDL-, so the net effect is the truncation of IOC.FETCHL+ and IOC.RNIL+ to a single cycle pulse with the timing of IOC.VALIDL-. IOC.FETCHL+ and IOC.RNIL+ may be one cycle long, or as long as three or four cycles. The multiplexer circuit allows only the last cycle to pass through, thus normalizing the signal to remove timing dependencies. Product terms 1 and 2 of the PAL equation for IOC.MODESEL- shows that the flip-flop is set whenever the first assertion of IOC.QFETCHL+ or IOC.QRNIL+ is detected. Product term 3 keeps IOC.MODESEL- set in the next cycle if the flip-flop is already set. IOC.BPONL+ is used to initialize the flip-flop to a false state when power is first turned on or when the CPU is reset.

The IOC.A900MODE- flag is defaulted to high (false, or NOT A900 mode) at system hardware reset or power on. The A900 mode of operation is more intrusive on the processor. In fact, defaulting to the A900 mode at power up would cause the A600+/A700 CPUs to hang up at the end of microcoded self-test. The A900 CPU initially works in the IOC A600+/A700 mode because it is impossible to have DMA interference from the extender box at power up or hardware reset until after the first I/O instruction. Immediately following the first instruction broadcast on the A900 CPU, IOC.A900MODE- goes low, and remains low permanently until the power is cycled or the system is reset. If the I/O extender is talking with an A600+/A700 CPU, no mode change occurs because the defaulted mode was correct.

```
--- PAL equation for IOC.A900MODE-
A900MODE := /A900MODE*/MODESEL*QRNIL*BPONL
                                   ;A900MODE-=0 when QRNIL+=1
          +  A900MODE*BPONL        ; comes first (no QFETCH+=1)
                                   ; subtle difference between the
                                   ; A600+/A700 and the A900 CPUs
```

According to product term 1, the IOC.A900MODE- flip-flop is set only if IOC.QRNIL+ is detected before the IOC.MODESEL- flip-flop is set. Once set, the IOC.A900MODE- flip-flop remains set (product term 2) until power is cycled or the CPU reset. If IOC.QFETCHL+ occurs first, IOC.MODESEL- sets and prevents the IOC.A900MODE- flip-flop from being set.

CPU/Extender Arbitration

The A-Series I/O extender generally behaves as an ordinary I/O card until it must seize control of the backplane to finish incomplete DMA activity, and subsequently prepares the extender to receive instructions from the CPU.

In preventing a backplane protocol deadlock, the I/O extender uses BP.MRQ- on the CPU backplane and XP.CPUTURN- on the extender backplane. BP.MRQ- is employed to delay the processor's access to its own backplane while XP.CPUTURN- prevents the I/O interface cards from beginning any new DMA activity. When BP.MRQ- and XP.CPUTURN- are used together, this results in the CPU being paused while the I/O extender is curtailing new DMA transfers.

The A600+/A700 Case

CPU Warns Extender of Impending Instruction Broadcast

I/O instruction processing is always handled in a two step format, that of fetching the instruction and then the broadcasting of I/O instructions. Regular I/O cards are oblivious to the instruction fetch and treat those as regular data transfers from the memory to the CPU. The I/O extender's IOC card, acting as a coprocessor on the backplane bus, monitors every move made by the CPU. It treats the fetched instruction as an opportunity to see the future. If the processor fetches an I/O instruction, it should predictably broadcast that same instruction several cycles later.

This method for detecting the occurrence of potential I/O instruction broadcasts is always true for the A700 processor, but only mostly true for the A600+ processor. A unique microcode feature of the A600+ CPU allows that processor to fetch the next instruction while executing the current instruction. At the completion of the current instruction, but after the fetch of the next instruction, interrupt requests are checked. If no interrupts are pending, the next instruction (already loaded into the CPU's instruction register) is decoded and executed. If an interrupt is present, the prefetched instruction is discarded in favor of servicing the interrupt request.

The look-ahead logic is reduced to detecting the possibility of an instruction broadcast occurring in the near future. Because of this uncertainty, it is not possible to utilize the fetched I/O instruction for an instruction broadcast on the extender backplane, to get a "jump" on the CPU. Instead, the fetching of an I/O instruction triggers a "traffic cop" mode, directing the CPU to wait until current DMA activity finishes, with guarantees that no new DMA starts.

Extender Freezes the CPU

Immediately following the detection of a fetched I/O instruction, the IOC card applies a two-cycle BP.MRQ- hold on the CPU. The signal responsible for this version of BP.MRQ- is traceable back to IOC.PROCHLDL- of the IOCP3 PAL [IO1105@15C].

```
--- PAL equation for IOC.PROCHLDL-
PROCHLDL := /PROCHLDL*/A900MODE*IOINSTR*/QRNIL*BPONL
                                      ;A600+ arbitrate
       +  PROCHLDL*/A900MODE*/S2*S1*/S0*BPONL
                                      ;hold processor one more
                                      ; cycle after State=2
       + /PROCHLDL* A900MODE*S2*S1*S0*/HOLDWAIT*/IOGOSL*
         /CPUTURNL*BPONL              ;A900 set MRQ when going from
                                      ; State=7 to State=0
       +  PROCHLDL* A900MODE*/S2*/S0*BPONL
                                      ;hold processor if previous
                                      ; cycle is State=0,2
```

Product term 1 causes IOC.PROCHLDL- to go low if IOC.PROCHLDL- is currently high while an I/O instruction is detected in the instruction register during an instruction fetch (IOC.QRNIL+ low, false). These conditions also cause the three state variables S2, S1, and S0 to jump directly to State=2 (S2=false,S1=true, S0=false). The operation of the arbitration state machine is explained in detail after the A900 discussion. For now, it is only necessary to know that when IOC.PROCHLDL- goes low, the arbitration state machine goes from State=0 to State=2. Product term 2 keeps IOC.PROCHLDL- low for one more cycle because State=2 at the end of the first cycle of IOC.PROCHLDL- low. Product terms 3 and 4 are not used in the A600+/A700 CPU modes. In this application, the arbitration state machine acts only as a two-cycle counter for IOC.PROCHLDL-.

After two cycles, IOC.PROCHLDL- goes high. Two cycles is enough time for any extender backplane DMA to have its presence registered with the IOC DMA state machine. If there is DMA to complete, BP.MRQ- remains low; as IOC.PROCHLDL- goes high to relinquish control, IOC.MRQFF- goes low to start a real DMA transfer. BP.MRQ- goes high at the conclusion of the last pending DMA transfer. At that time, the CPU is allowed to broadcast the I/O instruction.

While the CPU is prevented from making the I/O instruction broadcast, the IOC card is informing the EXT card to stop new DMA activity from starting. It achieves this with a warning signal known as IO.CTURNF- on the extender cable. The origin of IO.CTURNF- also comes from the IOCP3 arbitration control PAL, as the output IOC.CPUTURNFLG-.

```
--- PAL equation for IOC.CPUTURNFLG-
CPUTURNFLG = /A900MODE*/S2*S1        ;A600+: CPUTURNFLG-=0
           + /A900MODE*/S2*S0        ; at State=1..3
           +  A900MODE*S2            ;A900 : CPUTURNFLG-=0
           +  A900MODE*S1            ; at State=2..7
           +  A900MODE*S0            ;
           +  IOGOSL                 ;both: tracks IOGOSL and
                                     ; IORQFF (externally)
```

Only product terms 1, 2, and 6 are used in the A600+/A700 mode of operation. Product terms 1 and 2 are used together to generate IOC.CPUTURNFLG- low if the current state is State=1, State=2, or State=3. IOC.CPUTURNFLG- is a combinatorial signal, so it decodes the current state of the state machine. Since the fetch of an I/O instruction triggers a jump to State=2, which is immediately followed by State=3, and ultimately to State=1 after the instruction broadcast, IOC.CPUTURNFLG- goes into effect at the same time that the CPU is held off by IOC.PROCHLDL-. Product terms 1 and 2 keep IOC.CPUTURNFLG- low until after the broadcast instruction has been passed to the EXT. At that time, control for generating the extender's XP.CPUTURN- passes to the EXT, but the IOC may help out in one additional case. IOC.CPUTURNFLG- also goes low whenever there is an I/O handshake acknowledge (BP.IOGO- low) occurring on the backplane. A gate [IO0705@17D] combines IOC.CPUTURNFLG- with IOC.IORQFF- to create IOC.CTURNF+. IOC.CTURNF+ represents all of the conditions for which the IOC card supplies a CPUTURN component for the EXT card's generation of XP.CPUTURN-. IO.CTURNF+ is inverted and buffered [IO1204@19D] to create the IO.CTURNF- cable signal, which is used by the EXT card to drive XP.CPUTURN- until it is capable of generating CPUTURN by itself.

The A900 Case

Extender Keeps CPU Off the Backplane

Unlike the other A-Series processors, the A900 CPU does not drop any hints that an I/O instruction is coming. However, the I/O extender is still required to hold off the CPU until it can guarantee that there is no extender DMA interference with the instruction broadcast. Without some sort of a predictive mechanism, the I/O extender falls back on a preventive mechanism for CPU/extender arbitration in the A900 mode of operation.

Instead of the IOC waiting for the A900 CPU to take action first, IOC.PROCHLDL- goes low, causing BP.MRQ- to stay low, until the CPU asks the IOC to get off the bus.

```
--- PAL equation for IOC.PROCHLDL-
PROCHLDL := /PROCHLDL*/A900MODE*IOINSTR*/QRNIL*BPONL
                                         ;A600+ arbitrate
       +  PROCHLDL*/A900MODE*/S2*S1*/S0*BPONL
                                         ;hold processor one more
                                         ; cycle after State=2
       + /PROCHLDL* A900MODE*S2*S1*S0*/HOLDWAIT*/IOGOSL*
         /CPUTURNL*BPONL                 ;A900 sets MRQ when going from
                                         ; State=7 to State=0
       +  PROCHLDL* A900MODE*/S2*/S0*BPONL
                                         ;hold processor if previous
                                         ; cycle is State=0,2
```

The first two product terms apply only to the A600+/A700 mode of operation. The last two product terms are associated with the A900 mode of operation. In product term 3, IOC.PROCHLDL- is currently high, and goes low on the next rising edge of IOC.BCLK+ if the arbitration state machine is currently in State=7 and IOC.QRNIL+, IOC.HOLDWAIT-, IOC.IOGOSL-, and IOC.CPUTURNB+ are all false. IOC.HOLDWAIT- is the combination of IOC.IORQFF-, IOC.CRSL-, IOC.IAKS-, IOC.IOGOS-, and IOC.REQIOHS+. IOC.HOLDWAIT- also causes the arbitration state machine to jump immediately to State=7, or if in State=7, to remain there. If the state machine is to be kept in State=7, then IOC.PROCHLDL- is also not allowed to go low. A state transition from State=7 to State=0 occurs when all of the CPU's bus usage conditions are false. IOC.PROCHLDL- goes low to follow the state transition from State=7 to State=0.

Product term 4 keeps IOC.PROCHLDL- low for the next clock cycle if the current state is State=0 or State=2. State=0 is the CPU inactive state. IOC.PROCHLDL- is low for State=0 and keeps BP.MRQ- low. A state transition to State=2 is made after the A900 CPU makes a BP.CPUTURN- bus request and the IOC card arbitration circuitry "wakes up". State=2 lasts only one cycle because the state machine automatically advances to State=3 to wait for the instruction broadcast. IOC.PROCHLDL- goes high one cycle after entering State=3 because product term 4 no longer applies. Two cycles after BP.CPUTURNB- goes low, IOC.PROCHLDL- goes high. As with the A600+/A700 case, if any DMA transfers were in the DMA queue during the two cycle arbitration period, they are allowed to hold off the CPU and complete the transfers.

CPU Asks Extender for Utilization of the Backplane

Due to the I/O extender's use of BP.MRQ- as a preventive measure against backplane deadlocks, the A900 CPU is forced to rely on the assertion of BP.CPUTURN- every time it needs to utilize the backplane. These activities are :

1. Assert BP.RNI- to broadcast an I/O instruction.
2. Assert BP.IOGO- to acknowledge an I/O handshake.
3. Assert BP.IAK- to acknowledge an interrupt request.
4. Assert BP.CRS- to perform a CLC 0,C soft reset of the I/O system.

Preceding every BP.RNI-, BP.IOGO-, BP.IAK-, and BP.CRS-, the CPU asserts BP.CPUTURN-. The IOC card receives BP.CPUTURN- into an inverting buffer [I00807@10A], which provides the buffered version, IOC.CPUTURNB+.

IOC.CPUTURNB+ is received by the IOCP3 PAL, and activates the arbitration state machine. The IOC.CPUTURNFLG- output is true whenever the state machine is not at State=0. This activity is similar to the A600+ mode of operation.

```
--- PAL equation for IOC.CPUTURNFLG-
CPUTURNFLG = /A900MODE*/S2*S1          ;A600+:CPUTURNFLG-=0
           + /A900MODE*/S2*S0          ; at State=1..3
           +  A900MODE*S2              ;A900 : CPUTURNFLG-=0
           +  A900MODE*S1              ; at State=2..7
           +  A900MODE*S0              ;
           +  IOGOSL                   ;both: tracks IOGOSL and
                                       : IORQFF (externally)
```

In the A900 mode, IOC.CPUTURNFLG- goes low to shut down extender DMA activity whenever the arbitration state machine is not in its inactivity state (State=0), as indicated by product terms 3, 4, and 5. Product terms 1 and 2 are not used in the A900 mode. As in the A600+/A700 mode, IOC.CPUTURNFLG- also goes low whenever there is an I/O handshake acknowledge (BP.IOGO- low) occurring on the CPU backplane. External to the PAL, IOC.CPUTURNFLG- is combined with IOC.IORQFF- to create IOC.CTURNF+ which becomes IO.CTURNF- after buffering for transmission on the cable.

The Arbitration State Machine

The arbitration state machine consists of three flip-flops in the IOCP3 PAL [I01105@15C], serving to maintain the three state variables IOC.S2-, IOC.S1-, and IOC.S0-. Unlike the other status flip-flops implemented using the registered PALs, the arbitration state machine state variables do not cause direct action. In fact, these flip-flop outputs are unconnected pins on the IC. However, their purpose is to keep track of the current activities of the CPU.

All eight possible states represented by the state variables are significant. A power on reset forces the state machine to State=0 (IOC.S2-, IOC.S1-, and IOC.S0- all false [high]). The registered PAL has active low outputs, so use the following table to associate STATE# with the PAL output levels.

STATE#	IOC.S2-	IOC.S1-	IOC.S0-	COMMENTS
0	high	high	high	default, no activity
1	high	high	low	A600+/A700 after RNI cycle
2	high	low	high	I/O instr. or CPUTURN detected
3	high	low	low	wait for RNI cycle
4	low	high	high	A900 after RNI cycle
5	low	high	low	from State=4
6	low	low	high	from State=5
7	low	low	low	from State=4..7

State=0 is the inactivity state. A transition to State=2 comes after the IOC detects an impending I/O instruction broadcast. Depending on processor type, this is triggered by the CPU fetching a potential I/O instruction, or by the CPU asserting BP.CPUTURN- to prepare for an instruction broadcast. After one cycle at State=2, State=3 occurs automatically. The arbitration state machine remains in State=3 until the I/O instruction broadcast.

If the processor mode is A600+/A700, a one cycle pause occurs with a transition to State=1 before returning to State=0. This pause at State=1 adds one cycle to the IOC's generation of IOC.CPUTURNFLG-.

For the A900 mode, State=4 is entered immediately following the I/O instruction broadcast. A sequential count towards State=7 follows, with the state number increasing by one every cycle. A jump to State=7 may also occur if IOC.HOLDWAIT- is true. Remember that IOC.HOLDWAIT- is the combination of IOC.IORQFF-, IOC.CRSL-, IOC.IAKS-, IOC.IOGOS-, and IOC.REQIOHS+. During states State=3 through State=7 inclusive, the IOC card is not generating IOC.PROCHLDL- to keep the CPU off the backplane. In fact, any of the IOC.HOLDWAIT- conditions forces the CPU hold circuitry to wait in State=7. This allows the CPU to complete its backplane activity without I/O extender interference. As soon as all of the IOC.HOLDWAIT- conditions become false, a transition to State=0 is made. As mentioned earlier, IOC.PROCHLDL- becomes true for a State=7 to State=0 transition.

The following PAL Boolean equations describe the arbitration state machine's operation.

```
--- PAL equations for IOC.S2-, IOC.S1-, and IOC.S0-
S0 := /S2* S1*/S0*BPONL                   ;[2>3] ALL intermediate state
    + /S2* S1* S0*/A900MODE*/IOINSTR*BPONL
                                          ;[3>3] A600+ wait for QRNIL
                                          ;[3>1] A600+ after QRNIL
                                          ;[3>2] A600+ IOINSTR received
    + /S2* S1* S0*/A900MODE*QRNIL*BPONL ; while QRNIL is low...
                                        ; means time-out for
                                        ; fetched instruction
    + /S2* S1*     A900MODE*/QRNIL*BPONL
                                        ;[2>3] A900 unless QRNIL
                                        ;[3>3] A900 wait for QRNIL
                                        ;[3>4] A900 after QRNIL or
                                        ;[3>7] A900 after IOGOS,
                                        ; IAKS, or CRSL
    +  S2*    /S0* A900MODE*/QRNIL*BPONL
                                        ;[4>5] A900 after QRNIL
                                        ;[6>7] A900 after QRNIL
                                        ;[*>4] A900 if another QRNIL
    +              A900MODE*HOLDWAIT*BPONL
                                        ;[*>7] A900 wait if IOGOS,
                                        ; IAKS, CRSL, or IORQFF
                                        ;[7>0] A900 when all done
    +  S2*         A900MODE*CPUTURNB*BPONL
                                        ;[4..7>7] A900 CPUTURNB
                                        : forces State=7
    +  S2* S1* S0* A900MODE*IOGOSL*BPONL
                                        ;[7>7] A900 wait if IOGOSL
                                        ;[7>0] A900 when all done

S1 := /S2*        /A900MODE*IOINSTR*/QRNIL*BPONL
                                        ;[0..3] A600+ go to State=2
                                        ; if fetched
                                        ; instruction is I/O
    + /S2*/S1*/S0* A900MODE*CPUTURNL*/QRNIL*PROCHLDL*BPONL
                                        ;[0>2] A900 start
    + /S2* S1*               /QRNIL*BPONL
                                        ;[2>3] A600+ until QRNIL
                                        ;[3>3] A600+ until QRNIL
                                        ;[3>1] A600+ after QRNIL
                                        ;[2>3] A900 until QRNIL
                                        ;[3>3] A900 until QRNIL
                                        ;[3>4] A900 after QRNIL
    +  S2*/S1* S0* A900MODE*/QRNIL*BPONL
                                        ;[5>6] A900 after QRNIL
                                        ;[*>4] A900 if another QRNIL
    +  S2* S1*/S0* A900MODE*/QRNIL*BPONL
                                        ;[6>7] A900 after QRNIL
                                        ;[*>4] A900 if another QRNIL
    +              A900MODE*HOLDWAIT*BPONL
                                        ;[*>7] A900 wait if IOGOS,
                                        ; IAKS, CRSL, or IORQFF
                                        ;[7>0] A900 when all done
    +  S2*         A900MODE*CPUTURNB*BPONL
                                        ;[4..7>7] A900 CPUTURNB
                                        ; forces State=7
    +  S2* S1* S0* A900MODE*IOGOSL*BPONL
                                         ;[7>7] A900 wait if IOGOSL
                                         ;[7>0] A900 when all done
S2 := /S2*/S1*/S0*/A900MODE*/MODESEL*QRNIL*BPONL
                                         ;[0>4] A900 1st I/O instr.
                                         ; A900mode FF not yet set
    +              A900MODE*QRNIL*BPONL ;[*>4] A900 after QRNIL or
    +  S2*/S1*     A900MODE*BPONL        ;[4>5] A900 after QRNIL
                                         ;[5>6] A900 after QRNIL
    +  S2* S1*/S0* A900MODE*BPONL        ;[6>7] A900 after QRNIL
    +              A900MODE*HOLDWAIT*BPONL
                                         ;[*>7] A900 wait if IOGOS,
                                         ; IAKS, CRSL, or IORQFF
                                         ;[7>0] A900 when all done
    +  S2*         A900MODE*CPUTURNB+BPONL
                                         ;[4..7>7] A900 CPUTURNB
                                         ; forces State=7
    +  S2* S1* S0* A900MODE*IOGOSL*BPONL
                                         ;[7>7] A900 wait if IOGOSL
                                         ;[7>0] A900 when all done
```

Rather than discuss the state machine on the basis of each product term, a state transition diagram provides better information than plain text. (See Figure 12-11.) Comments to the right of each product term in the Boolean equations above can provide some guidance. Keep in mind that all three state variables are updated simultaneously every clock cycle. All state transitions occur at the start of the long half cycle of the clock.

Extender/Extender Arbitration

All arbitration between I/O extenders occurs over the host processor backplane. This design concept simplifies the multi-extender environment because each I/O extender is configured as if it were the only extender in the system. Extenders are added simply as peripherals; for each I/O extender, there is an interface card which plugs into the CPU card cage, a cable, and the box.

Except for the CPU-I/O extender arbitration protocols, an extender's IOC card appears as a virtual I/O card to another IOC or to another I/O card in the CPU backplane. The IOC card receives and generates priority in the same manner as the regular I/O interface cards, so extender-extender arbitration is similar to interface card to interface card arbitration. All CPU-I/O extender arbitration takes place using the wired-OR open-collector BP.MRQ- signal, so all IOC cards arbitrate in parallel. If one IOC needs to keep the CPU off the backplane for a longer period of time to complete its DMA, it keeps BP.MRQ- asserted longer than the other IOC cards.

With a pipelined implementation, the I/O extender system does exhibit one anomaly while establishing priority between simultaneous interrupt requests between two IOC cards. The anomaly is explained in greater detail during the discussion of interrupt handling. Some time after a lower priority IOC card requests an interrupt, the CPU responds with an interrupt acknowledgement. At the start of the acknowledgement, a higher priority IOC card makes an interrupt request while preparing to perform a simultaneous DMA transfer. Since interrupt priority is resolved during the first cycle of the acknowledgement, the higher priority IOC has full protocol rights to respond to the acknowledgement. However, doing so would jeopardize the DMA information held on the IOC buffers because of the subsequent trap cell fetch (which looks like a DMA memory handshake). The DMA cycle cannot start because the CPU is in the middle of an interrupt acknowledgement wait. If the DMA cycle were started, the CPU would mistake the DMA cycle for a trap cell fetch (DMA handshaking protocols are similar to trap cell fetch protocols). The only solution for the higher priority IOC card, is to grant priority to the lower priority IOC card, thus allowing the trap cell fetch to continue. Immediately after the trap cell fetch, the trapped DMA cycle goes to memory, and the higher priority IOC card waits for the next interrupt acknowledgement cycle.

Handling I/O Instructions

After the processor has broadcast the I/O instruction over its backplane, the I/O extender must make use of that instruction. The I/O extender responds differently to the various types of I/O instructions; in some cases, those instructions are even ignored.

Instruction Types

Only a small set of instructions in the HP 1000 instruction set are called I/O instructions. Even within the classification of I/O instructions, many I/O instructions are not even executed by an I/O interface card.

In the A-Series distributed intelligence I/O-architecture, the CPU is responsible for the proper sequencing of the instruction flow, the execution of non-I/O and system level I/O instructions, and the broadcast of I/O instructions to the I/O interface cards. After the instruction broadcast, the execution of the I/O instruction is the sole responsibility of the addressed I/O card(s). Some I/O instructions do not require an acknowledgement as a sign of instruction completion. Another subset of I/O instructions may time-out while waiting for an I/O handshake request. If the time-out is invoked, the processor treats the broadcast I/O instruction as a NOP (no operation).

Non-I/O and System Level I/O

Most of the instructions executed by the central processor are not related to I/O handling. These non-I/O instructions include the memory reference, memory management, bit/byte/word-manipulation, arithmetic, and operating system/language assist instructions. The I/O interface cards are completely ignored by the CPU when fetching and executing non-I/O instructions. Since BP.RNI- is not asserted when fetching and executing non-I/O instructions, the I/O cards also ignore what the CPU is doing. Since there is no unnecessary interaction between the CPU and the I/O cards, the CPU can compute efficiently while the I/O cards perform DMA with very little overhead.

All I/O instructions use the STF, CLF, STC, CLC, SFS, SFC, OTA/B, LIA/B, MIA/B, and the HLT mnemonics. However, not every instruction using these mnemonics directly involve the I/O interface cards. The CPU is responsible for executing some of these pseudo I/O-instructions, or system level I/O instructions. All I/O instructions consist of the instruction itself, and a target select code to address an I/O card. Most select codes less than 20B (octal) refer to system level I/O activities which include the CPU BOOT SEL switches, CPU status LEDs, interrupt system status, memory parity error, memory protection, and time base generator operations.

The only I/O operations with select codes less than 20B that are recognized and executed by the I/O interface cards are the LI*/MI*/OT* 0 (Interrupt Mask), CLF/STF/STC/LI*/MI*/OT* 2 (Global Register), LI*/MI*/OT*3[,C] (Psave and romP), and any HLT xx[,C] (halt) instructions. These lower select code operations (<20B) are handled by the I/O cards in the same way as the higher select code (>=20B) operations except for OT* 0 and OT* 2. When the processor writes to the Interrupt Mask (OT* 0) or to the Global Register (OT* 2), it actually writes into the appropriate register on every I/O card, not just to a specifically addressed I/O card.

Control Reset (CRS or CLC 0)

A system-wide programmable reset is performed with the CLC 0 control reset instruction (sometimes called CRS in other documents). This system level I/O instruction is used for programmatically resetting the state of the processor and all I/O cards to a known operating environment. The CLC 0 instruction is not broadcast to the I/O cards for decoding and execution. Instead, the processor pulses the control reset line on the backplane, BP.CRS-, for one clock cycle. This signal is received by all I/O cards and is used to reset most state machines to their power-on defaults.

The I/O extender receives BP.CRS- into an inverting buffer [IO0807@10A], which is followed by a flip-flop [IO0906@12A] clocked with IOC.BCLK+ to synchronize the signal to the start of the long half cycle. The resulting signal, IOC.CRSL+, splits off in two directions. In one branch of the split, IOC.CRSL+ enters the IOCP4 PAL to generate the single-cycle delayed IOC.CRSLL-.

```
--- PAL equation for IOC.CRSLL-
CRSLL := CRSL*BPONL             ;delayed version of CRSL
```

IOC.CRSL+ also travels to an AND gate [IO0805@17D] where it is masked by IOC.CRSLL-. This operation ensures that the message sent to the EXT board is only one clock cycle long even if the CPU generates a longer BP.CRS-. The AND gate output is sent to an inverting buffer [IO1104@19D] to generate the IO.CNTLRST- cable message to the EXT board.

IO.CNTLRST- terminates in a resistor network [EX1003@10D] on the EXT board which provides a default state when the cable is not installed. The EXT board receives IO.CNTLRST- into an inverting buffer [EX0704@11D] to generate EXT.CNTLRST+, which becomes the J-input to the CRS flip-flop [EX0304@15E]. In addition to synchronizing the control reset message to the start of the short half cycle, the CRS flip-flop also logs pending and current control reset operations. If DMA is active on the extender backplane (XP.MRQ- low, leading to EXT.MRQS+ high), the EXT card must not drive XP.CRS- low; doing so would lead to the abnormal abortion of DMA activity and may cause bad data to be written into memory. The CRS flip-flop output, EXT.CNTLRSTS-, is masked with EXT.MRQS+ [EX1205@16E] to determine if a backplane XP.CRS- should be generated. When DMA completes, EXT.MRQS+ goes low, allowing EXT.CRS+ to go high at the start of the short half cycle following XP.MRQ- going high. EXT.CRS+ is inverted and buffered [EX1107@19C] to create XP.CRS-. The CRS flip-flop is cleared one cycle later because EXT.CRS+ is also routed to the K-input. The J and K inputs are never simultaneously high just before the falling edge of EXT.BCLK+, so proper sequencing of the CRS flip-flop is guaranteed.

Memory Protection (Privileged Level - STC 7 and CLC 7)

Another system level I/O operation executed by the processor, but involving the I/O interface cards, is the memory protection instructions, STC 7 and CLC 7. When executing code with memory protection enabled, not only is protected memory kept safe from accidental modifications, most I/O instructions are prevented from execution. STC 7 and CLC 7 are more appropriately called privileged level instructions.

Some I/O instructions are not destructive to the operating system's handling of I/O, and are allowed to execute even when memory protect is enabled. These instructions are all confined to select code 1, an I/O channel defined to handle the BOOT SEL switches, the processor status LEDs, and the overflow bit operations.

Because the CPU executes the STC 7 and CLC 7 instructions, it advises the I/O cards of the current memory protection status through a backplane signal called BP.MP+. When BP.MP+ is high, all broadcast I/O instructions must be ignored by the I/O cards. The I/O extender receives BP.MP+ into an inverting buffer [IO0807@10A]. The inverted signal is synchronized to the start of the long half cycle by a flip-flop [IO0806@12B] clocked with IOC.BCLK+, generating IOC.MPL-. IOC.MPL- is used as an input selector to a multiplexer [IO0706@13A] which generates the single cycle normalized signals IOC.QFETCHL+ and IOC.QRNIL+. When IOC.MPL- is low (memory protect enabled), IOC.QFETCHL+ and IOC.QRNIL+ are always low, since the selected IOA/IOB inputs are grounded. The I/O extender's instruction decoding logic is effectively disabled when IOC.QFETCHL+ and IOC.QRNIL+ are forced to remain low.

I/O Instructions

The I/O extender recognizes all of the real I/O instructions that are executed by the I/O interface cards. These include all operations dealing with select codes of 20B through 77B, and certain select code functions from 0B through 17B. It is important to keep in mind that the I/O extender does not execute any I/O instructions. All instructions detected by the IOC are passed to the EXT for broadcast and eventual execution by an I/O card. If an I/O card in the extender recognizes the I/O instruction, it is executed by that card. For STF/CLF, STC/CLC, and SFS/SFC (if test result is false), interaction with the CPU is not required, so the instruction is considered executed once it is broadcast. The more complex I/O instructions such as SFS/SFC (when the test result is true), LI*, MI*, OT*, and HLT instructions require communication with the CPU.

When these instructions are received by the addressed I/O card in the extender, it makes an I/O handshake request, using XP.IORQ-. This request is received by the EXT, then the IOC is informed. The IOC follows suit by issuing a BP.IORQ-, mirroring the activity on the extender backplane. The CPU responds through BP.IOGO-, and eventually, XP.IOGO- is generated on the extender backplane. In general terms, every I/O handshake on the extender backplane is recreated on the CPU backplane.

Instruction Detection

I/O processing in the extender begins with instruction detection, which is dependent on processor type.

A600+/A700 Mode

The I/O extender detects the coming of an I/O instruction by decoding every instruction fetched by the CPU over its backplane. Every instruction fetched causes IOC.QFETCHL+ to pulse high for one cycle beginning with the start of the long half cycle following the assertion of BP.VALID-. An OR gate [IO0604@17E] combines IOC.QFETCHL+ with IOC.QRNIL+ to create the generalized IOC.FCHRNIL+, a signal which pulses when there is a valid instruction to decode. Thus, for the A600+ and A700 CPUs, every broadcast I/O instruction is decoded twice, once during the processor fetch, then later during the broadcast.

If the first instruction decode operation signals that an I/O instruction is forthcoming, the arbitration state machine moves from State=0 to State=2. This action triggers the two-cycle arbitration period when IOC.PROCHLDL- and IOC.CPUTURNFLG- both go low. The arbitration state machine then goes to State=3 after one cycle at State=2, and waits until the instruction broadcast occurs.

The A600+ processor may not actually perform an instruction broadcast next if an interrupt is pending. Instead, it services the interrupt by doing a vectored trap cell instruction fetch. On some occasions, the trap cell instruction fetch may be combined with the instruction broadcast cycle (BP.FETCH- and BP.RNI- low simultaneously). The I/O extender can handle this case because it has already performed the necessary arbitration. If an I/O instruction resides in the trap cell, it may be executed without contention although it is not the same instruction decoded earlier. If the trap cell contains a non-I/O instruction, the second pass decode detects this and terminates the arbitration phase as well, with the arbitration state machine proceeding to State=1 and then to State=0.

When the trap cell instruction fetch uses only BP.FETCH-, the arbitration state machine remains in State=3. It waits in State=3 until the next IOC.QRNIL+ is received. If the processor fetches another I/O instruction while in State=3, the arbitration state machine drops back to State=2 to resynchronize itself. Then it waits for the instruction broadcast again.

A900 Mode

Instruction detection on the A900 CPU is very simple. Because the I/O extender is driving BP.MRQ-, the A900 CPU precedes every instruction broadcast with an assertion of BP.CPUTURN-. The arbitration state machine also makes a transition from State=0 to State=2 and then to State=3. As with the A600+/A700, the I/O extender waits until the instruction broadcast occurs.

Instruction Decoding

All instructions fetched and broadcast are interpreted by instruction decode PLA IOCA1 [IO0506@35B]. It receives 15 bits of the instruction and a signal which indicates that a valid instruction is in the instruction register, IOC.FCHRNIL+. The instruction decoder generates two outputs, IOC.IOINSTR+ to indicate that a valid I/O instruction is in the instruction register, and IOC.OT02+ to flag OT* 0 and OT* 2 instructions. Non-I/O and system level I/O instructions are not recognized by the instruction decoder.

Normal I/O Instructions

The following instructions cause the instruction decoder to drive IOC.IOINSTR+ high during the time FCHRNIL+ is high.

Instruction	Select Codes (Octal)
STF	2, 20..77
CLF	2, 20..77
STC	2, 20..77
CLC	20..77
SFS	20..77
SFC	20..77
LIA/B	0, 2, 3, 20..77
MIA/B	0, 2, 3, 20..77
OTA/B	0, 2, 3, 20..77
HLT	0..77

Only 15 bits of the instruction op code are required in the instruction decode process because the 9th bit in the op code (DB9) determines the source or destination CPU register (A or B-register). This information is not significant to the I/O extender since the same operations occur regardless of the register selected.

Instruction decoding is enabled at the same time that the instruction register goes transparent beginning with the start of the long half cycle following the assertion of BP.VALID-. The data-out register [I00605@34C and I00406@34D] and the instruction register [I00606@34B and I00507@34C] are latched according to the IOC.DOLCH+ latch enable signal. Tracing back its roots, IOC.FCHRNIL+ is the initiating component. IOC.FCHRNIL+ is sent into an AND-OR-INVERT gate [I01206@31E], then redundantly (IOC.FCHRNIL+ is already qualified by IOC.VALIDL-) masked by IOC.VALIDL- at a gate [I01106@32E] to generate IOC.PELCH+, the parity error register latch. Parity error status on the backplane, BP.PE-, is valid only at the start of the next long half cycle following the deassertion of BP.VALID-. IOC.PELCH+ is further framed by IOC.VALIDB+ with a NAND gate [I00705@32E]. The backplane data bus has been valid for 50 nsec before the rising edge of BP.VALID-, so it is appropriate to latch the data at the de-assertion of BP.VALID-. The latch signal is combined with IOC.IOLCH- (to handle data transfers during I/O handshakes) at a gate [I00705@33E] to create the generalized IOC.DOLCH+ data-out register latch enable.

IOC.IOINSTR+ goes high prior to the closing of the data-out latch, and remains high until after the start of the next long half cycle. This allows other logic functions on the IOC board to sample IOC.IOINSTR+ at the start of the long half cycle. IOC.IOINSTR+ is qualified with IOC.QRNIL+ at a NAND gate [I00707@37B] to select only the instruction broadcast version of IOC.IOINSTR+. This result is sent into a D-type flip-flop [I00607@38B] which is clocked with IOC.BCLK+. The output of the flip-flop, IOC.INSTRAV+, lasts one clock cycle. IOC.INSTRAV+ is inverted by a buffer [I01204@19D] to generate the cable signal IO.INSTRAV-, which informs the EXT board that an instruction broadcast cycle should be performed. The data-out register contents are passed to the EXT using the IO.DATAAV- operation.

OT* 0 and OT* 2 Instructions

Due to the unusual handshaking protocol for the OT* 0 and OT* 2 I/O instructions, the instruction decoder singles out these two instructions and flags their appearance in the instruction flow.

Although I/O instructions are generally executed by only one I/O card, STF/CLF/STC/CLC instructions may be executed by more than one I/O card at any time because execution of these non-handshaking I/O instructions are completely transparent to the CPU. The Global Register enable/disable instruction, CLF/STF 2, is one such case. All I/O cards must execute this instruction and properly set the Global Register mode flag in order for it to properly interpret the succeeding I/O instructions. Multiple I/O card execution of the STF/CLF/STC/CLC instructions pose no problems for the I/O extender.

SFS and SFC instructions may never be executed simultaneously by two I/O cards (unless the two I/O cards have the same select code). LI* and MI* instructions reference a single I/O card, so simultaneous handshaking is not possible.

Most of the time, the OT* instruction also references a single I/O card. When writing to the interrupt mask (OT* 0) or to the Global Register (OT* 2), it is necessary for all I/O cards on the CPU backplane to participate in the I/O handshaking process so that each card may obtain the data. The I/O cards in the extender backplane cannot get involved with these handshakes directly because the handshaking delays introduced by the I/O extender subsystem would only interfere with the handshaking on the CPU backplane.

The IOC.OT02+ output of the instruction decoder flags these two instructions to force the IOC into a modified handshake mode known as the special I/O handshake. On the extender backplane, all I/O cards still participate in the OT* 0 and OT* 2 handshakes, but the IOC card does not mirror that operation on the CPU backplane. Instead, it merely monitors the handshaking to recover the Interrupt Mask or Global Register data. Once that data is retrieved, the EXT card uses that data for the second (data transfer) handshake.

IOC.OT02+ is synchronized to the start of the long half cycle by a D-type flip-flop [IO0607@38B] to create IOC.OT02L+. IOC.OT02L+ goes high for one clock cycle following the instruction fetch and the instruction broadcast of the OT* 0 and OT* 2 instructions.

Instruction Broadcasting on Extender Backplane

For all processor types, the instruction broadcast sequence begins by passing the instruction from the IOC registers to the EXT board. The data-out register is loaded at the end of the CPU backplane instruction broadcast, so the cable data-out bus is being driven with the current contents of the instruction register. Meanwhile, the appropriate cable message is being formulated to tell the EXT to pick up the cable data. IOC.RNIL+ is combined with another signal at a NOR gate [IO1106@17A], then qualified with IOC.VALIDL- at a gate [IO0604@17C]. The resulting signal has the same timing characteristics as IOC.QRNIL+, but inverted, and is then combined with IOC.IOHS3- at a gate [IO0705@18C] to create the generalized IOC.DATAAV+ message. IOC.DATAAV+ is finally inverted and driven out on the cable by a buffer [IO0704@19C] as IO.DATAAV-, a message telling the EXT to pick up the cable data-out bus during the next clock cycle.

The IO.DATAAV- cable message is sent out while the instruction is being decoded. When the CPU performs an instruction broadcast cycle with BP.RNI-, an I/O instruction always appears on the data bus. The IOC card waits one cycle until it can generate IOC.INSTRAV+ and IOC.OT02L+. IOC.INSTRAV+ is used directly by the inverting buffer [IO1204@19C] to make the IO.INSTRAV- cable message. Although IOC.INSTRAV+ can be generated one cycle earlier, and thus save one clock cycle, IOC.OT02L+ cannot be created until the instruction has been completely decoded. Since the standardized cable message protocol prohibits the use of any signal lasting less than one full clock cycle, IOC.OT02+ cannot be used to create a cable message. IOC.OT02L+ is combined with IOC.ACKIOHSFF+ at an OR gate [IO0604@17D], then inverted by a buffer [IO1204@19D] to generate the overall IO.ACKIOHS- message. Therefore, IO.ACKIOHS- serves a double purpose, primarily to acknowledge I/O handshakes, and secondarily to modify the meaning of IOC.INSTRAV- to signal for a special I/O instruction (OT* 0 and OT* 2) handling procedure.

IO.DATAAV- goes into an inverting buffer [EX0904@11C] on the EXT board to create EXT.DATAAV+, which travels to a D-type flip-flop [EX0106@32D] that samples the signal at the start of the next short half cycle of the clock. The synchronized signal, EXT.DBAVS-, is gated with EXT.BCLK- at a gate [EX0206@33D] to remove the first half cycle of EXT.DBAVS-. This half cycle signal, EXT.DOLCH+, is used by the data-out registers [EX0406@34C and EX0405@34C] and the parity error register [EX0305@32C] to latch the instruction off of the extender cable and to hold its value until new information is sent by the IOC card.

The EXT card receives IO.INSTRAV- with an inverting buffer [EX0904@11C]. The line is terminated with a resistor network [EX1003@10C] to provide a default condition when the cable is removed. The buffer outputs EXT.INSTRAV+, a single clock cycle pulse signal that changes at the beginning of the long half cycle, and is sampled by the EXTP1 DMA Control PAL EXTP1 [EX1004@15B] at the start of the short half cycle. EXT.INSTRAV+ automatically causes EXT.RNICYC- to be asserted for one cycle.

```
--- PAL equation for EXT.RNICYC-
RNICYC := INSTRAV*BPONL                 ;RNICYC when INSTRAV received
```

EXT.RNICYC- is received by three gates: an inverter [EX1206@18B] and two gates [EX0806@18B]. The outputs of these three gates all pass through inverting buffers [EX0807@19B] to create the extender backplane control signals XP.RNI-, XP.VALID-, and XP.BUSY-. EXT.RNICYC- forces XP.VALID- and XP.BUSY- to be generated concurrent with XP.RNI-, which satisfies all of the protocols required to broadcast an I/O instruction to all of the I/O cards on the extender backplane.

No backplane arbitration is required for the XP.RNI- assertion because all DMA has been suspended with XP.CPUTURN- since the initial detection of a pending instruction broadcast on the CPU backplane. XP.CPUTURN- is driven by an inverting buffer [EX0807@19B]. Tracing backwards from the buffer, CPUTURN is actually a combination of two signals at a gate [EX0806@18B]: EXT.CTURNFS- and EXT.CPUTURNFF-. EXT.CTURNFS- is the inverted version of EXT.CTURNFS+ [EX0806@17B], which in turn is a half cycle delayed version of EXT.CTURNF+ [EX1204@13C]. The EXT card receives the EXT.CTURNF- message from the cable into a buffer [EX0904@11C] which inverts the signal to form EXT.CTURNF+. The D-type flip-flop which samples EXT.CTURNF+ is clocked with EXT.BCLK-, which captures the input at the start of the short half cycle when EXT.CTURNF+ is valid.

EXT.CTURNFS+ contributes to XP.CPUTURN- only until the I/O instruction is broadcast on the CPU backplane. EXT.CPUTURNFF- takes over immediately following the receipt of the EXT.INSTRAV+ message.

```
--- PAL equation for EXT.CPUTURNFF-
CPUTURNFF := INSTRAV*BPONL                ;set CPUTURNFF to prevent
                                          ; interference from DMA
                                          ; during an instr. broadcast
           + CPUTURNFF*/CNTR0*BPONL       ;keep CPUTURN for at least
           + CPUTURNFF*/CNTR1*BPONL       ; three cycles after RNICYC
           + CPUTURNFF*IORQL*BPONL        ;or when IORQL comes around
           + DMAWAIT*BPONL                ;set when DMAWAIT is true
                                          ; (INTRQLL+=1 or IOGOFF-=0)
```

The EXT version of CPUTURN takes over with the assertion of XP.RNI-, as indicated by product term 1, which is identical to that used for generating EXT.RNICYC-. Once EXT.CPUTURNFF- is set, product terms 2, 3, and 4 keep EXT.CPUTURNFF- set. Product terms 2 and 3 rely on a two bit counter, CNTR1 and CNTR0.

```
--- PAL equations for EXT.CNTR0- and EXT.CNTR1-
CNTR0 := /CNTR1*/CNTR0*RNICYC*BPONL    ;CNTR0 asserted in cycle 1
       +  CNTR1*/CNTR0*BPONL           ; and cycle 2 after RNICYC
       +  CNTR1* CNTR0*/INSTRAV*BPONL  ;reset to 0 on next instr

CNTR1 := /CNTR1* CNTR0*BPONL           ;CNTR1 asserted in cycle 2
       +  CNTR1*/CNTR0*BPONL           ; and cycle 3 after RNICYC
       +  CNTR1* CNTR0*/INSTRAV*BPONL  ;reset to 0 on next instr
```

According to the 3rd product term for each equation, the counter normally stays in Count=3 (CNTR1=true and CNTR0=true) until EXT.INSTRAV+ becomes true. EXT.INSTRAV+, while generating EXT.RNICYC-, also makes the Count=0. At Count=0, EXT.RNICYC- is true, which advances the count to Count=1 (CNTR0 product term 1), then to Count=2 (CNTR1 product term 1), then finally to Count=3 (CNTR0 and CNTR1 product term 2). This simple state machine simply cycles from 3 to 0 to 1 to 2 to 3. At power up, the sequence is entered beginning at Count=0.

IO.INSTRAV- causes EXT.CPUTURNFF- to go low at the same time the counter resets to Count=0. The 2nd and 3rd product terms for the CPUTURN flip-flop allow EXT.CPUTURNFF- to remain low until the count advances to Count=3. Therefore, XP.CPUTURN- remains low on the extender backplane for at least three cycles after the rising edge of XP.RNI-.

The companion signal, IO.ACKIOHS-, used with IO.INSTRAV- to indicate a special handshake environment, is also received by a buffer [EX0904@11C] which creates the inverted signal EXT.ACKIOHS+. The D-type flip-flop [EX1204@13C] is also used to sample EXT.ACKIOHS+ at the start of the short half cycle to satisfy the cable timing requirements. Along with EXT.INSTRAVS+, the synchronized version of EXT.INSTRAV+, EXT.ACKIOHSS+ is used by the I/O handshaking PAL EXTP2 [EX1005@15C]. The operation of this PAL is explained in the following sections.

Going back to the instruction broadcast, EXT.INSTRAV+ is also used by the EXTP1 PAL to assert EXT.MEMREGEN- for one cycle.

```
--- PAL equation for EXT.MEMREGEN-
MEMREGEN := /WRITE*/DMACYC*BUSYL*DATAAV*BPONL
                                       ;copy VALIDFF for a DMA read
          + INSTRAV*BPONL              ;copy RNICYC for instruction
                                       ; broadcast
                                       ;MEMREGEN enables PE driver
                                       ; and DATA-OUT register for
                                       ; DMA read and instruction
                                       ; broadcast cycles
```

Note that the second product term is identical to the PAL equation for generating EXT.RNICYC-, so the timing of EXT.MEMREGEN- is the same as for EXT.RNICYC- in this situation. EXT.MEMREGEN- is used to enable the data-out data bus buffer onto the extender backplane during an instruction broadcast or during a DMA read completion. EXT.MEMREGEN- is synchronized to the start of the long half cycle by a flip-flop [EX0805@32E], creating EXT.PEREN+. In addition to enabling the parity error flag buffer, EXT.PEREN+ is combined with the EXT.OTACYC+ component (enables data bus drivers for OT* handshakes) at a NOR gate [EX0105@35E] to generate EXT.DOREN-. When EXT.DOREN- goes low, the data bus data-out buffers [EX0506@35C and EX0505@35C] are enabled to drive the backplane data bus with the information in the data-out registers. This path for data is used for instruction broadcasting, DMA reads from memory, and OT* I/O handshaking.

Non-Handshaking Instructions - STF/CLF and STC/CLC Case

Once the STF/CLF and STC/CLC instructions have been broadcast to the I/O cards in the extender box, the backplane protocols do not provide for any further action from the I/O card. The EXT card keeps XP.CPUTURN- asserted for three cycles after the assertion of XP.RNI-. When XP.CPUTURN- times out, DMA is allowed to resume.

I/O Handshaking Instructions

The balance of the I/O instructions contain I/O handshaking protocols for their execution. I/O handshaking is the method of communication whereby the CPU is placed into slave mode and is told what to do next by the I/O cards. This communication is always sequenced through with a command handshake followed by a data transfer handshake.

XP.IORQ- and XP.IOGO- are the two extender backplane signals utilized for the handshaking. The I/O cards generate XP.IORQ- to request a handshake, and the EXT card acknowledges the request with XP.IOGO-. A similar handshake is reciprocated on the CPU backplane due to the role reversal of the IOC card. The IOC card, acting as a virtual I/O card, makes a BP.IORQ- request, and the CPU responds with BP.IOGO-.

The XP.IORQ- False Assertions

The I/O Master timing specification for XP.IORQ- permits it to glitch during instruction decoding. The period of uncertainty begins during XP.RNI- and lasts about one clock cycle beyond. The I/O extender avoids any problems caused by the false assertions of XP.IORQ- with the use of a masking circuit that voids any assertion of XP.IORQ- within the clock cycle following the deassertion of XP.RNI-.

The masking circuit is based on the delayed versions of EXT.RNICYC-. A D-type flip-flop [EX0805@34E] uses EXT.RNICYC- to create the time-shifted EXT.MASKIORQL+ that is synchronized to the start of the long half cycle. EXT.MASKIORQL+ is time shifted by another half cycle using a D-type flip-flop [EX1104@13D], generating EXT.MASKIORQLS+. EXT.MASKIORQL+ and the shifted EXT.MASKIORQLS+ are combined together with a NOR gate [EX0206@35E] to produce the 1.6-cycle mask signal EXT.MASKIORQC-.

The I/O handshake request masking takes place at a AND gate [EX0804@11A], where EXT.MASKIORQC- meets EXT.IORQB+, the buffered version of XP.IORQ- after it has been processed by an inverter [EX1107@11A]. The first opportunity for the EXT card to detect an I/O handshake request comes on the second falling edge of XP.SCLK- following the deassertion of XP.RNI-. The qualified version of the I/O handshake request, EXT.QIORQ+, is sent to two destinations. It is received by a flip-flop [EX1106@12A] to generate EXT.IORQL+, a signal synchronized to the start of the long half cycle. EXT.QIORQ+ also travels to the EXTP2 PAL which handles all of the EXT's I/O handshaking chores.

The XP.IORQ- masking circuit is in effect only for the first clock cycle after an instruction broadcast, and therefore has no effect at the start of the second handshake in double handshake protocols.

The SFS/SFC Case

If an SFS/SFC instruction is broadcast on the I/O extender backplane, and the addressed I/O card determines that the skip condition is false, that I/O card simply does nothing. The EXT card merely times out for XP.CPUTURN- and releases the backplane for DMA usage. The CPU also times out, treating the instruction as a NOP, and proceeds to the next instruction.

When the skip condition is true, the addressed I/O card makes an I/O handshake request by asserting XP.IORQ-. After processing through the IORQ-masking circuit, a qualified EXT.QIORQ+ is generated, followed shortly with the assertion of EXT.IORQL+. EXT.IORQL+ is time shifted by a half cycle using a D-type flip-flop [EX1104@13D], creating EXT.IORQLS+. This signal is time shifted again by a half cycle at another D-type flip-flop [EX1106@12A] to generate EXT.IORQLL+. Since EXT.IORQLL+ is a full cycle behind EXT.IORQL+, the difference between these two signals can be used to create a one cycle pulse whenever EXT.IORQL+ goes high. This is accomplished by using a NAND gate [EX0905@15A] to invert EXT.IORQLL+, and the result is combined with EXT.IORQL+ using an AND gate [EX0804@15A]. This transition detection circuit creates a single cycle pulse, EXT.REQIOHS+, that is sent to an inverting buffer [EX0704@19D] to generate the EX.REQIOHS- cable message sent to the IOC board.

While EX.REQIOHS+ is being generated, the I/O handshaking PAL is trying to respond to the I/O handshake request. The EXT card asserts XP.IOGO- immediately to extract the control word from the handshaking I/O card. An I/O handshake for an SFS/SFC instruction always contains the "increment PC (program counter)" control word. This control word must be passed to the IOC card for it to tell the CPU to increment its program counter during the I/O handshake performed on the CPU backplane.

XP.IOGO- is derived from EXT.IOGOFF-, an output of the EXTP2 I/O handshaking PAL [EX1005@15C].

```
--- PAL equation for EXT.IOGOFF-
IOGOFF := /IOGOFF*QIORQ*FIRSTHS*/SPECIO*/SLVWAIT*BPONL
                                     ;1st HS SF*/OT*[not 0/2]/LI*
        + /IOGOFF*QIORQ*SPECIO*ACKIOHSS*/INSTRAVS*BPONL
                                     ;1st/2nd HS OT*[0/2]
        + /IOGOFF*QIORQ*FIRSTHS*SLVWAIT*ACKIOHSS*/INSTRAVS*BPONL
                                     ;5th HS HLT/break
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*IOCWOT*OTAFLAG*ACKIOHSS*
          /INSTRAVS*BPONL            ;2nd HS OT*[not 0/2]
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*/IOCWOT*ACKIOHSS*
          /INSTRAVS*BPONL
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*/IOCWOT*IOGOPD*BPONL
                                     ;2nd HS LI*
        +  IOGOFF*/IOGO3*BPONL       ;IOGO until IOGO3 set
        +  IOGOFF*IOGO3*IORQL*BPONL  ;IOGO for extended IORQ
```

The 1st product term, used in responding to an I/O handshake request, contains three qualifiers: EXT.FIRSTHS- must be true, and both EXT.SPECIO- and EXT.SLVWAIT- must be false. EXT.FIRSTHS- is used to enable this product term only for the first handshake in a double handshake situation. EXT.SPECIO- disables this product term when executing an OT* 0 or OT* 2 instruction. EXT.SLVWAIT- also eliminates involvement of this product term when synchronizing a multiple handshake operation, as is the case for HLT/break slave mode transfers.

When executing an SFS/SFC instruction, the three qualifiers always make the product term true, so EXT.IOGOFF- is asserted on the long half cycle of the clock after EXT.QIORQ+ becomes true. Once EXT.IOGOFF- is set, it remains set for at least three clock cycles. A simple two stage counter, consisting of EXT.IOGO23- and EXT.IOGO3-, is used to account for this passage in time.

```
--- PAL equation for EXT.IOGO23- and EXT.IOGO3-
IOGO23 := /IOGO23*IOGOFF*EXTGOS*BPONL  ;start IOGO cycle 2
        +  IOGO23*/IOGO3*BPONL         ;IOGO23 until IOGO3 set
        +  IOGO23*IOGO3*IORQL*BPONL    ;IOGO23 for extended IORQ

IOGO3 := /IOGO3*IOGO23*BPONL           ;start IOGO cycle 3
       +  IOGO3*IORQL*BPONL            ;IOGO3 for extended IORQ
```

EXT.IOGO23- becomes true one clock cycle after EXT.IOGOFF- goes low when EXT.EXTGOS+ is true, as indicated by product term 1. EXT.EXTGOS+ is always true for SFS/SFC instructions, because it is only used to arbitrate between concurrent DMA and a slave mode transfer caused by a system console (VCP) break operation. Product term 2 keeps EXT.IOGO23- going until EXT.IOGO3- is set. EXT.IOGO3- is set one cycle after EXT.IOGO23- becomes true, and is self-extinguishing unless EXT.IORQL+ is still true during the third cycle of EXT.IOGOFF-. This situation is the "extended IORQ" case which never applies to I/O interface cards designed with the I/O Master backplane interface. The handshaking protocol requires XP.IOGO- to go high exactly two clock cycles after XP.IORQ- goes high. The 8th product term for EXT.IOGOFF-, the 3rd product term for EXT.IOGO23-, and the 2nd product term for EXT.IOGO3-, all include hooks for the "extended IORQ" operation by freezing the I/O handshake in the third cycle until EXT.IORQL+ goes low. However, due to timing of the EXT-to-IOC data passage during an I/O handshake, XP.IORQ- cannot be extended without causing incorrect data (or control word) to be passed to the CPU. (The IOC does not handle the "extended IORQ" case because not all A-Series CPUs support the feature.) After the third cycle of EXT.IOGO-, all three flip-flops (EXT.IOGOFF-, EXT.IOGO23-, EXT.IOGO3-) return to their inactive states.

The control word from the I/O interface is retrieved during the last cycle of XP.IOGO- low. A gate [EX0206@31D] qualifies EXT.IOGO3- with EXT.IORQL+ to make EXT.QIOGO3+, a signal which goes high for one clock cycle during the last cycle of EXT.IOGO-. EXT.QIOGO3+ is time shifted by half a clock cycle, to the start of the short half cycle, after going through a D-type flip-flop [EX0106@32D] that is clocked with EXT.BCLK-. The output of the flip-flop, EXT.IODBCLK-, and the memory access data-in bus clock, EXT.MEMCLK50-, are merged together at a gate [EX0404@34D] to produce the completed data-in register clock signal, EXT.DICLK+. EXT.DICLK+ is used by the data-in registers [EX0606@34A and EX0605@34B] as the clock input, causing them to sample the data bus. For an I/O handshake, the data bus is always sampled at the start of the short half cycle during the last clock cycle of XP.IOGO- low.

EXT.DICLK+ is also received into a D-type flip-flop [EX0504@17D] to generate the time-shifted EXT.ABDBAV+, a signal that is synchronized to the start of the long half cycle. EXT.ABDBAV+ serves two purposes. First, it is inverted and buffered by a gate [EX0604@19D] to create the EX.ABDBAV- cable message for the IOC, indicating that new data is available on the cable's data-in bus. EXT.ABDBAV+ is also used by the data-in resynchronizing registers [EX0702@35A and EX0502@35B] to time shift the entire data-in bus to the start of the long half cycle. Recall that EXT.DICLK+ causes the first set of data-in registers to be loaded at the start of the short half cycle (for I/O handshaking) to satisfy the extender's I/O backplane timing. The second set of data-in registers time shifts the data to the start of the long half cycle to satisfy the timing requirements on the cable. All cable signals are permitted to change at the start of the long half cycle in order to reduce crosstalk effects.

The data-in bus is buffered by a set of drivers [EX0802@36A and EX0602@36B] to provide cable driving capability for the resynchronizing registers.

After the EXT completes an SFS/SFC I/O handshake, the IOC is still in the middle of its I/O handshake with the CPU. The IOC's involvement with the I/O handshake begins with the receipt of an asserted IOC.REQIOHS+, an inverted version of EX.REQIOHS- that has been processed through a buffer [IO0904@12C]. IOC.REQIOHS+ triggers the IOCP2 I/O handshaking PAL [IO1004@15B] to perform an I/O handshake with the CPU.

```
--- PAL equation for IOC.IORQFF-
IORQFF := /IORQFF*REQIOHS*/SPECIORQFF*IOPON
                                        ;EXT requests IO handshake
        + /IORQFF*IORQPD*IOPON          ; queued up request
        +  IORQFF*/IOGOS*IOPON          ;keep until IOGO 1st cycle
        +  IORQFF*IOGOSL*IOPON          ; (IOGOS true, IOGOSL false)
```

IOC.IORQFF- goes low when IOC.REQIOHS+ is received if the current instruction is not related to OT* 0 and OT* 2 (IOC.SPECIORQFF- false). Outside of the PAL, IOC.IORQFF- is inverted by a gate [I01007@18C], creating IOC.IORQFF+. IOC.IORQFF+ is used by both inputs to an open-collector NAND gate [I00907@19C] which drives BP.IORQ-. Since BP.IORQ- always goes low when IOC.IORQFF- goes low, any discussion about the state of IOC.IORQFF- also applies to BP.IORQ-.

Product term 1 initiates an I/O handshake request when there are no handshakes in the queue. Product term 2 is used in multiple handshake situations when there is a handshake queue. Product terms 3 and 4 keep IOC.IORQFF- set until the first cycle of BP.IOGO-, which is detected by a difference between the states of IOC.IOGOS+ and IOC.IOGOSL-. IOC.IOGOS+ is generated by sampling the inverted/buffered version of BP.IOGO- [I01104@10C] with a D-type flip-flop [I01006@11C]. IOC.IOGOS+ is a half cycle, time-shifted version of BP.IOGO-; BP.IOGO- changes at the start of the long half cycle while IOC.IOGOS+ changes at the start of the next short half cycle.

Inside the IOCP2 I/O handshaking PAL, IOC.IOGOS+ is used to generate IOC.IOGOSL-, a time-shifted version of IOC.IOGOS+. IOC.IOGOSL- changes at the start of the long half cycle. One cycle after the assertion of BP.IOGO-, IOC.IOGOS+ is true, but IOC.IOGOSL- is not yet true. This difference is detected by the IOC.IORQFF- logic as indicating the first cycle of BP.IOGO-, so IOC.IORQFF- is deasserted.

While the BP.IORQ- and BP.IOGO- handshaking protocols are being negotiated, the IOC is also obtaining the control word from the EXT card. The EXT's EX.ABDBAV- message is received with a buffer [I00904@12C] to create the inverted IOC.ABDBAV+. The responsibility for generating the register clocks to sample the data-in and address registers resides with the IOCP1 DMA state machine PAL [I00905@15A]. A signal called IOC.ABDBCLK+ is derived from IOC.ABDBAV+.

```
--- PAL equation for IOC.ABDBCLK+
/ABDBCLK := /IOPON                        ;ABDBCLK+=0 at power up
          + /MEMGOPD*/ABDBAV              ;if MEMGOPD not set, stay
                                          ; at 0 until ABDBAV true
          +  MEMGOPD*MRQFF*/MEMGOFF       ;if MEMGOQUEUE and MEMGOPD
          +  MEMGOPD*MRQFF* BUSYL         ; are set, pulse ABDBCLK hi
          + /MEMGOQUEUE*MEMGOPD*MRQFF*/ABDBAV
                                          ; immediately after current
                                          ; good MEMGO
                                          ;if only MEMGOPD set, pulse
                                          ; ABDBCLK after current good
                                          ; MEMGO only if new data is
                                          ; on the cable (ABDBAV)
                                          ;if current MEMGO is no
                                          ; good (A900 extended BUSY)
                                          ; when MEMGOQUEUE is set,
                                          ; forcing data pickup
                                          ; off the cable at the end
                                          ; of a good MEMGO.
```

The Boolean equations for IOC.ABDBCLK+ are written in their inverted form because the PAL has normally low-true outputs, but a high-true output is desired. De Morgan's theorems are used to adapt the Boolean equations to the device, but one can also treat the equations such that each product term generates a condition for IOC.ABDBCLK+ to be false.

During the execution of an I/O handshake, the IOC card is not performing any DMA activity, so the DMA queue is empty. Product term 1 initializes IOC.ABDBCLK+ to a low state. Product term 2 keeps IOC.ABDBCLK+ in the low state unless IOC.ABDBAV+ is true and the DMA queue is empty (IOC.MEMGOPD-false). When IOC.ABDBAV+ arrives during an I/O handshake, IOC.ABDBCLK+ pulses. IOC.ABDBCLK+ lasts only one clock cycle, allowing product term 2 to bring the output low again. Product terms 3, 4, and 5 are used during high speed DMA write operations to coordinate the passing of parameters when the DMA queue is full or partially full. IOC.ABDBCLK+ is used by the data-in registers [I00505@32C and I00305@32D] to sample the data on the cable data-in bus.

The data-in registers are enabled to drive the CPU backplane data bus with the assertion of IOC.DIREN-, an output of an AND-OR-INVERT gate [I01206@32E]. One half of that gate relates to DMA write operations into memory, and the other half is used for I/O read operations by the CPU. An IOC.IORDEN+ (I/O read enable) signal is framed by IOC.IOGOB+ so that the data bus is driven by the IOC only while BP.IOGO- is actually asserted. IOC.IORDEN+ is the inverted [I00404@18C] form of IOC.IORDEN-, an output of the IOCP2 I/O handshaking PAL.

```
--- PAL equation for IOC.IORDEN-
IORDEN := /IORDEN*IORQFF*IOGOS*/IOGOSL*/OTAFLAG*IOPON
                                          ;DATA-IN enable:extender>CPU
       +  IORDEN*IOGOS*IOPON              ;keep until IOGO ends
```

According to product term 1, IOC.IORDEN- is asserted after IOC.IORQFF- is set, the first cycle of BP.IOGO- is encountered, and the IOC.OTAFLAG+ message from the EXT is false. IOC.OTAFLAG+ is always false for the first I/O handshake (or odd-numbered handshake in a multi-handshake case) because the default direction for data flow is from the extender backplane to the CPU backplane. IOC.OTAFLAG+ is true during the second handshake of an OTA* instruction to indicate that there is a bus reversal. With IOC.IORDEN- becoming true at the end of the first cycle of BP.IOGO-, the IOC starts to drive the data bus at the beginning of the second cycle. Product term 2 keeps IOC.IORDEN- set until it detects the deassertion of BP.IOGO-. Since IOC.IORDEN- is not reset until one cycle after BP.IOGO- goes away, IOC.IOGOB+ is used outside of the PAL to limit the IOC's use of the data bus at the end of the I/O handshake.

The handshaking for the LI*, MI*, and OT* uses a double handshake protocol. During a SFS/SFC handshake, the CPU is merely told to increment the program counter to skip over the next instruction. The LI*, MI*, and OT* instructions require both a command message to the CPU followed by a data transfer cycle between the CPU and the I/O card.

The LI* and MI* Cases

The LI*/MI* instructions require a double handshake consisting of a control word transfer followed by a data transfer from the I/O card to the CPU. The first handshake is enacted as described above for the SFS/SFC handshake except that the control word passed to the CPU is of the form "at the end of the next handshake, take the data bus contents (as driven by the I/O card), and load it in (or merge with) the A (or B) register". Since there are four possible combinations with two variables (load/merge and A/B register), there should be four different control words that may be sent to the CPU during the control word handshaking for the LI* and MI* instructions. Actually, there are only three: load A-register, load B-register, and merge register. For the merge instruction, the CPU uses the register specified in the I/O instruction (MIA or MIB).

To distinguish between the first and second handshakes (or odd-numbered versus even-numbered handshakes for HLT), the EXT maintains an EXT.FIRSTHS- flag. EXT.FIRSTHS- is true (low) during the first (odd) handshake, and false (high) during the second (even) handshake.

```
--- PAL equation for EXT.FIRSTHS-
FIRSTHS := INSTRAVS*BPONL                ;get ready for I/O handshake
         + /SCHOD*BPONL                  ; for instruction or slave
         +  FIRSTHS*/IOGOFF*BPONL        ;keep FIRSTHS until IOGO HS
         +  FIRSTHS*IOGOFF*/IOGO3*BPONL ; through IOGO1 and IOGO2
         +  FIRSTHS*IOGOFF*IOGO3*IORQL*BPONL
                                         ; hold if IOGO3 extended
         + /FIRSTHS*IOGOFF*IOGO3*/IORQL*BPONL
                                         ;ready for third handshake
```

The first two product terms set the EXT.FIRSTHS- flag prior to the start of the first I/O handshake. Product term 1 applies to broadcast I/O instructions and product term 2 is used for slave handshake operations (break). Product terms 3, 4, and 5 provide the means for keeping EXT.FIRSTHS- asserted. For the LI* and MI* instructions, EXT.FIRSTHS- goes true during the instruction broadcast on the extender backplane and remains true until EXT.IOGOFF- is generated by the EXT card (product term 3). Product term 4 keeps EXT.FIRSTHS- low until the third cycle of EXT.IOGOFF-, while product term 5 extends EXT.FIRSTHS- by the same number of cycles that IOC.IOGOFF- is extended. In either case, EXT.FIRSTHS- goes high at the same time EXT.IOGOFF- goes high at the end of the first handshake. If the broadcast instruction does not apply to a card in the extender box, XP.IORQ- is never generated, which leads to EXT.IOGOFF- never getting set, then EXT.FIRSTHS- remains true until the next instruction broadcast. Product term 6 is used for HLT/break mode slave transfers.

The control word handshake for an LI*/MI* instruction is performed in the same way as the SFS/SFC handshake, differing only in the contents of the control word sent to the CPU. As with the first handshake, the second handshake is also initiated by an I/O card's assertion of XP.IORQ-. The second XP.IORQ- starts one cycle after the deassertion of the first XP.IORQ-. The timing diagram, Figure 12-3, shows that the second handshake request is made while the first handshake is in progress (XP.IOGO- low).

For LI*/MI* instructions, the data bus is always directed from the I/O card to the CPU during both the control word and data handshakes. This means that the second handshake can begin on the extender backplane (to retrieve the data from the I/O card) once the EXT card receives an acknowledgement from the IOC card that an I/O handshake has begun on the CPU backplane. The second handshake is not allowed to start unless there is a guarantee that the control word, temporarily stored in an IOC register, has been utilized before the incoming data from the second handshake overwrites the IOC register. If the CPU delays the first handshake on the CPU backplane, the EXT card delays the second handshake on the extender backplane.

The IOC card synchronizes the handshaking between both backplanes by generating a signal called IO.ACKIOHS-. The origin of this signal comes from the IOC.ACKIOHSFF+ output in the IOCP2 I/O handshaking PAL [IO1004@15B].

```
--- PAL equation for IOC.ACKIOHSFF+
/ACKIOHSFF := /IOPON                    ;ACKIOHSFF+= 0 at power up
          + /ACKIOHSFF*/IOGOS           ;ACKIOHSFF+= 1 if 1st cycle
          + /ACKIOHSFF*IOGOSL           ; of IOGO and
          + /ACKIOHSFF*/IORQFF*/SPECIORQFF
                                        ; IORQFF/SPECIORQFF set
          +  ACKIOHSFF                  ;one cycle pulse only
```

Product term 1 initializes IOC.ACKIOHSFF+ to a logic zero. Product terms 2, 3, and 4 keep IOC.ACKIOHSFF+ at logic zero until the first cycle of BP.IOGO- when either IOC.IORQFF- or IOC.SPECIORQFF- is true. This filter mechanism prevents IOC.ACKIOHSFF+ from becoming true when the CPU is handshaking with another I/O card on the CPU backplane. Product term 5 provides the path for bringing IOC.ACKIOHSFF+ low after it has been high for one cycle. IOC.ACKIOHSFF+ is sent to the EXT as the IO.ACKIOHS- cable message.

As with the first handshake, the second handshake request is also acknowledged with the assertion of XP.IOGO-. In the fastest case (first handshake proceeds without delay on the CPU backplane), XP.IOGO- goes low one cycle after having gone high for the first handshake.

```
--- PAL equation for EXT.IOGOFF-
IOGOFF := /IOGOFF*QIORQ*FIRSTHS*/SPECIO*/SLVWAIT*BPONL
                                   ;1st HS SF*/OT*[not 0/2]/LI*
        + /IOGOFF*QIORQ*SPECIO*ACKIOHSS*/INSTRAVS*BPONL
                                   ;1st/2nd HS OT*[0/2]
        + /IOGOFF*QIORQ*FIRSTHS*SLVWAIT*ACKIOHSS*/INSTRAVS*BPONL
                                   ;5th HS HLT/break
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*IOCWOT*OTAFLAG*ACKIOHSS*
          /INSTRAVS*BPONL          ;2nd HS OT*[not 0/2]
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*/IOCWOT*ACKIOHSS*
          /INSTRAVS*BPONL
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*/IOCWOT*IOGOPD*BPONL
                                   ;2nd HS LI*
        +  IOGOFF*/IOGO3*BPONL     ;IOGO until IOGO3 set
        +  IOGOFF*IOGO3*IORQL*BPONL    ;IOGO for extended IORQ
```

Product term 5 starts the second handshake during an LI*/MI* instruction because EXT.FIRSTHS- is false, it is not an EXT.SPECIO- operation, and the control word does not cause EXT.IOCWOT- to be set. The EXT card sets EXT.IOGOFF- for the second handshake upon receipt of an EXT.ACKIOHSS+ (qualified by EXT.INSTRAV+ to omit the special case for EXT.ACKIOHSS+ assertion), indicating that the IOC card will be using the control word within the next clock cycle. As with the first (control word) handshake, the second (data transfer to CPU) handshake completes after three cycles of EXT.IOGOFF- low. As before, the data retrieval logic implicitly clocks the data off of the extender backplane midway through the last cycle of EX.IOGOFF-.

From the IOC side of the extender, the second handshake is performed in an identical fashion to the first handshake. The same mechanisms used for obtaining and driving the control word onto the CPU backplane apply to the data transfer. The IOC views both handshakes for an LI*/MI* instruction as being the same operation.

The OT* (Except Select Codes 0 and 2) Case

The handshaking of common OT* instructions (excluding select codes 0 and 2) is initiated by the I/O cards using the methods described for the first handshake of the LI*/MI* instructions. As before, only one I/O card responds to this instruction: making a handshake request, having the request acknowledged, and finally passing a control word to the CPU by way of the I/O extender.

The control word for a generic OT* instruction tells the CPU to place the contents of the A or B register onto the data bus during the second handshake. Since a bus reversal is expected, the I/O extender also changes it mode of operation in order to return data to the requesting I/O card. Because two-way transfers cannot take advantage of the inherent pipelining in the extender, the double handshake OT* instruction requires the receiving I/O card to wait while the IOC performs two back-to-back handshakes on the CPU backplane. For the LI*/MI* instructions, the control word and data transfer handshakes can overlap, with the second handshake starting on the extender backplane after the first handshake on the CPU backplane has been initiated. Due to the bus reversal for the OT* instruction, the second handshake on the extender backplane is not allowed to start until the second handshake is completed on the CPU backplane and with the data retrieved from the CPU backplane.

A status flag, EXT.OTAFLAG-, is maintained on the EXT card to facilitate bus reversals. The signal goes low during the second (data transfer) handshake for OT* instructions. EXT.OTAFLAG- is generated by the EXTP2 I/O Handshaking PAL (EX1005@15C) for the data-out register data bus enables, and in its buffered/inverted form [IO0704@19D] is EX.OTAFLG+ for use by the IOC card.

```
--- PAL equation for EXT.OTAFLAG-
OTAFLAG := /OTAFLAG*IOGOFF*FIRSTHS*/SPECIO*IOCWOT*ACKIOHSS
           */INSTRAVS*BPONL
         + /OTAFLAG*/IOGOFF*IORQL*/FIRSTHS*/SPECIO*IOCWOT
           *ACKIOHSS*/INSTRAVS*BPONL    ;for OT*(not 0/2)
                                        ; 1st ACKIOHSS sets OTAFLAG
                                        ; 2nd ACKIOHSS sets IOGOFF
         + /OTAFLAG*IOGOFF*IOGO3*/IORQL*FIRSTHS*SPECIO*BPONL
                                        ;OT*(0/2) means auto OTAFLAG
         +  OTAFLAG*/IOGO3*BPONL        ;OTAFLAG until IOGO3 set
         +  OTAFLAG*IOGO3*IORQL*BPONL   ;OTAFLAG for extended IORQ
```

Product terms 1 and 2 perform identical functions except that the first term is in effect during EXT.IOGOFF- true, and the second term handles the case when EXT.IOGOFF- is false but a second handshake request is pending. Both product terms assume that the EXT.SPECIO- mode is off and the first handshake yielded an OTA/B-type control word (EXT.IOCWOT+ = true). When the EXT.ACKIOHSS+ message is received from the IOC card to signal that a BP.IOGO- has acknowledged the IOC's BP.IORQ-, EXT.OTAFLAG- sets. As a reminder, the first EXT.ACKIOHSS+ for an LI*/MI* handshake triggered the second handshake EXT.IOGOFF-. In an OT* handshake, the first EXT.ACKIOHSS+ merely sets the EXT.OTAFLAG- condition; the EXT.IOGOFF- for the second handshake is triggered by the second EXT.ACKIOHSS+. Once EXT.OTAFLAG- is set, it remains set with product terms 4 and 5, using the same conditions which keep the second EXT.IOGOFF- set. Therefore, EXT.OTAFLAG- is valid before the start of the second I/O handshake (on both CPU and extender backplanes), and stays valid until EXT.IOGOFF- deasserts. (See Figure 12-4.)

Several product terms in the EXTP2 I/O Handshake Control PAL utilize the EXT.IOCWOT+ signal, an output of an AND gate [EX0804@38B] which decodes all incoming data on the EXT's data-in register. Since the two-bit decoder examines only data bus bits DB7V+ and DB5V+ at all times, EXT.IOCWOT+ is significant only in the context of the control word I/O handshake. All PAL equations use EXT.IOCWOT+ with several I/O handshaking qualifiers. Of the 16 possible combinations for control words, only four control words affect bus reversals and their bit patterns require only two bits to decode.

Data Bus Bit :	DB7	DB6	DB5	DB4	
Control Word Bit :	CW3	CW2	CW1	CW0	Function
	0	0	0	0	NOP
	0	0	0	1	Load PC
	0	0	1	0	Load A-Register
	0	0	1	1	Load B-Register
	0	1	0	0	Set O-bit (overflow)
	0	1	0	1	Clear O-bit
	0	1	1	0	Merge with A(orB)-Register
	0	1	1	1	Increment PC
	1	0	0	0	Output Status register
	1	0	0	1	Enable ROM mode
IOCWOT+=H	1	0	1	0	Output A-register
IOCWOT+=H	1	0	1	1	Output B-register
	1	1	0	0	Clear E-bit (extend)
	1	1	0	1	Set E-bit
IOCWOT+=H	1	1	1	0	Output PC
IOCWOT+=H	1	1	1	1	Output, then increment PC

According to the control word table, CW3 and CW1, corresponding to DB7 and DB5, need to be at logic 1 in order to perform an output operation during the data transfer handshake. Therefore EXT.IOCWOT+ is high for all A/B/PC-register output operations.

After the EXT.OTAFLAG- status flip-flop is set, EXT.IOGOFF- is enabled to acknowledge the second handshake request on the extender backplane immediately upon the receipt of the second EXT.ACKIOHSS+, which indicates that the CPU has begun to place information on the CPU backplane. IOC.ACKIOHSFF+, originating on the IOC card, is a single cycle signal that is high during the second cycle of BP.IOGO-. The IOC card latches the backplane data midway through the third cycle of BP.IOGO-, using the data-out latch enable IOC.DOLCH+. IOC.DOLCH+ is the output of a gate [IO0705@33E] which combines IOC.MEMLCH- (for memory accessing) and IOC.IOLCH- (for I/O operations). IOC.IOLCH- itself is generated by combining IOC.IOHS3- and its time-shifted version, IOC.IOHS3S+ at a gate (IO0604@16E).

IOC.IOHS3- and IOC.IOHS2- are outputs of the IOCP2 I/O handshaking PAL [IO1004@15B].

```
--- PAL Equations for IOC.IOHS2- and IOC.IOHS3-
IOHS2 := IORQFF*IOGOS*/IOGOSL*IOPON     ;start IOGO counter with IOGO
       + SPECIORQFF*IOGOS*/IOGOSL*IOPON ; for IORQFF or SPECIORQFF

IOHS3 := IOHS2*BPONL                    ;count 3rd I/O Handshake cycle
```

According to product term 1, IOC.IOHS2- pulses for one clock cycle during the second cycle of BP.IOGO-; it has the same timing as IOC.ACKIOHSFF+. IOC.IOHS3- merely follows IOC.IOHS2- one cycle later, pulsing for one cycle during the third (last) cycle of BP.IOGO-. A flip-flop [IO0804@15E] time shifts IOC.IOHS3- by a half clock cycle to create IOC.IOHS3S+. IOC.IOLCH- is formed when both IOC.IOHS3- and IOC.IOHS3S+ are low, which is true during the first half cycle (long half cycle) of the third cycle of BP.IOGO-. When IOC.IOLCH- is used to generate the all-purpose IOC.DOLCH+, the data-out latches are transparent at the beginning of the third cycle of BP.IOGO-, then close a half clock cycle later.

Since the CPU is expected to drive the data bus during the second handshake for an OT* instruction, the IOC card must guarantee that it does not drive the bus at the same time. This is an application of the bus reversal message, EX.OTAFLG+. The IOC card receives EX.OTAFLG+ into an inverting buffer [IO0904@12D], creating IOC.OTAFLAG-, which is used by the IOCP2 I/O handshaking PAL. It affects IOC.IORDEN-.

```
--- PAL equation for IOC.IORDEN-
IORDEN := /IORDEN*IORQFF*IOGOS*/IOGOSL*/OTAFLAG*IOPON
                                    ;DATA-IN enable:extender>CPU
        +  IORDEN*IOGOS*IOPON       ;keep until IOGO ends
```

IOC.OTAFLAG- is valid before the second handshake begins on the CPU backplane. Therefore, when the EXT card signals that an output operation is to be performed, IOC.IORDEN- is false during the second handshake and keeps the IOC data-in register buffers off of the CPU backplane.

As data is captured by the IOC, an IO.DATAAV- message is sent to the EXT board to indicate that valid data is on the cable data-out bus. IO.DATAAV- is generated by running IOC.IOHS3- into a gate [IO0705@18C] to combine it with other DATAAV message sources, and inverting the result with a buffer [IO0704@19C].

From the EXT board's point of view, the second handshake begins after it receives IO.ACKIOHS-, buffers that to make EXT.ACKIOHS+, then time shifts the result to generate EXT.ACKIOHSS+. The EXTP2 I/O handshake PAL has been waiting with EXT.OTAFLAG- set.

```
--- PAL equation for EXT.IOGOFF-
IOGOFF := /IOGOFF*QIORQ*FIRSTHS*/SPECIO*/SLVWAIT*BPONL
                                        ;1st HS SF*/OT*[not 0/2]/LI*
        + /IOGOFF*QIORQ*SPECIO*ACKIOHSS*/INSTRAVS*BPONL
                                        ;1st/2nd HS OT*[0/2]
        + /IOGOFF*QIORQ*FIRSTHS*SLVWAIT*ACKIOHSS*/INSTRAVS*BPONL
                                        ;5th HS HLT/break
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*IOCWOT*OTAFLAG*ACKIOHSS*
          /INSTRAVS*BPONL               ;2nd HS OT*[not 0/2]
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*/IOCWOT*ACKIOHSS*
          /INSTRAVS*BPONL
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*/IOCWOT*IOGOPD*BPONL
                                        ;2nd HS LI*
        +  IOGOFF*/IOGO3*BPONL          ;IOGO until IOGO3 set
        +  IOGOFF*IOGO3*IORQL*BPONL     ;IOGO for extended IORQ
```

The fourth product term handles the second handshake for an OT* instruction. According to that product term, IOC.OTAFLAG- must be set (which "absorbs" the first IOC.ACKIOHSS+) before the second IOC.ACKIOHSS+ causes EXT.IOGOFF- to set. As always, once EXT.IOGOFF- is set, it lasts three cycles (or more) before it is cleared.

While the XP.IOGO- handshaking is in progress, the EXT card also receives the IO.DATAAV- message. As discussed before, this causes the data-out latches [EX0406@34C and EX0405@34C] to be loaded from the cable data-out bus. The contents of these latches are driven onto the extender backplane data bus when the buffers [EX0506@35C and EX0505@35C] are enabled by EXT.DOREN-. One of the inputs to the gate [EX0105@35E] that generates EXT.DOREN- is EXT.OTACYC+, which is generated by gating together EXT.IOGO23- and EXT.OTAFLAG- with a NOR gate [EX1205@19C]. This results in the EXT card driving the extender backplane data bus with the contents of the A (or B) register during the last two cycles of XP.IOGO-. Once the I/O card has received the data, the OT* instruction has completely executed, and DMA activity in the extender may resume one cycle later.

The OT* 0 and OT* 2 Cases

The OT* 0 And OT* 2 I/O instructions require all I/O cards to receive, interpret, and execute the broadcast instruction. The simultaneous execution does not present a problem if all I/O cards request and perform the I/O handshake in unison. Since BP.IORQ- is defined as a wired-OR open-collector backplane signal, all I/O cards in the CPU backplane perform the handshaking as if each card alone is handshaking with the CPU. However, the IOC card does not receive an EX.REQIOHS- from the EXT card until several clock cycles after the instruction is broadcast on the CPU backplane. Therefore if the IOC generates a BP.IORQ-, its handshaking is out of synchronization with the rest of the I/O system.

Although the OT* 0 and OT* 2 instructions use the same handshaking protocol as the generic OT* handshakes, a control word is not passed to the CPU at the end of the first (control word) handshake. The data bus is an actively driven bus, and unlike the open-collector BP.IORQ-, it is not a good policy for up to 18 totem-pole output buffers to be driving the same bus at the same time. For OT* 0 and OT* 2, the CPU "knows" what the control word should be, which allows the data bus to remain at a tri-stated level throughout the first (control word) handshake.

The I/O extender handles OT* 0 and OT* 2 instructions in a very special manner. When the IOC's instruction decoder detects an I/O instruction (IOC.IOINSTR+), it sorts out the OT* 0 and OT* 2 instructions as well, by generating IOC.OT02+. The IOCP2 I/O handshaking PAL [IO1004@15B] receives the synchronized version, IOC.OT02L+, which directly sets the IOC.SPECIORQFF- status flag. (See Figure 12-5.)

```
--- PAL equation for IOC.SPECIORQFF-
SPECIORQFF := /SPECIORQFF*OT02L*IOPON  ;OT* 0 and OT* 2
           +  SPECIORQFF*/QRNIL*SCHIDS*IOPON
                                       ;stay in SPECIORQ mode
           +  SPECIORQFF*/QRNIL*/SLAVEFF*IOPON
                                       ; until next fetch/slave
```

According to product term 1, IOC.SPECIORQFF- sets one cycle after IOC.OT02L+ is received. IOC.SPECIORQFF- remains set until the next I/O instruction broadcast (product term 2) or until the extender is called upon to perform a slave mode (break) I/O handshake (product term 2 and 3).

IOC.OT02L+ is also used to generate the IO.ACKIOHS- cable message that is sent simultaneously with IO.INSTRAV+ (synchronized IOC.IOINSTR+). An IO.ACKIOHS- message is meaningless before the instruction is broadcast on the extender backplane. Therefore, when IO.ACKIOHS- is used in conjunction with IO.INSTRAV+, this coded message to the EXT card indicates that the instruction broadcast contains an OT* 0 or OT* 2 instruction. When the EXTP2 I/O handshaking PAL [EX1005@15C] receives EXT.INSTRAVS+ with EXT.ACKIOHSS+, the EXT.SPECIO- flag is set immediately (product term 1).

```
--- PAL equation for EXT.SPECIO-
SPECIO := INSTRAVS*ACKIOHSS*BPONL      ;SPECIO-=0 if OT* 0 or OT* 2
        + SPECIO*/INSTRAVS*SCHOD*BPONL ;till next instr/slv ack
```

As with a similar flag on the IOC card, EXT.SPECIO- remains in effect from before the instruction broadcast on the extender backplane until the next EXT.INSTRAVS+ (product term 1 and 2), or until a slave mode handshake acknowledge (product term 2). Slave mode transfers are based on regular I/O handshakes, not on the OT* 0/2 variety.

On the CPU backplane after the broadcast instruction, all I/O cards have initiated the control word handshake by asserting BP.IORQ-. One cycle after BP.IOGO- is asserted, all I/O cards stop driving BP.IORQ- to allow the signal to be pulled up to a logic one. After a single inactive cycle, another assertion of BP.IORQ- is made by all I/O cards. Meanwhile, the CPU keeps BP.IOGO- low for three cycles, deasserts BP.IOGO-, then drives BP.IOGO- low again for three more cycles. During the second IOGO, the CPU places information on the backplane data bus for retrieval by the I/O cards.

While the I/O extender is in the special I/O handshake mode, the IOC card is prevented from generating BP.IORQ-, but is still required to monitor all activity on the CPU backplane. The IOC still issues an IO.ACKIOHS- message to indicate the passage of the first and second handshakes as they occur on the CPU backplane. The IOC also stays off the data bus during both handshakes. The only significant activity required of the IOC is to retrieve the data transferred from the CPU. The EXT modifies its handshaking to account for the slightly different mode of operation on the IOC card. The IOC.SPECIORQFF- and EXT.SPECIO- status flags allow the IOC and EXT to handle these situations.

Within the IOCP2 I/O handshaking PAL [IO1004@15B], IOC.SPECIORQFF- disables some operations while it replaces IOC.IORQFF- in others. Specifically, IOC.SPECIORQFF- affects the operation of IOC.IORQPD-, IOC.IORQFF-, IOC.IORDEN-, IOC.IOHS2-, and IOC.ACKIOHSFF+.

The assertion of IOC.SPECIORQFF- disables the generation of IOC.IORQPD-, IOC.IORQFF-, and IOC.IORDEN-.

```
--- PAL equations for IOC.IORQPD-, IOC.IORQFF-, and IOC.IORDEN-
IORQPD := /IORQPD*IORQFF*REQIOHS*/SPECIORQFF*IOPON
        +  IORQPD*IORQFF*IOPON         ;IORQ queue

IORQFF := /IORQFF*REQIOHS*/SPECIORQFF*BPONL
                                       ;EXT requests IO handshake
        + /IORQFF*IORQPD*IOPON         ; queued up request
        +  IORQFF*/IOGOS*IOPON         ;keep until IOGO 1st cycle
        +  IORQFF*IOGOSL*IOPON         ; (IOGOS true, IOGOSL false)

IORDEN := /IORDEN*IORQFF*IOGOS*/IOGOSL*/OTAFLAG*IOPON
                                       ;DATA-IN enable:extender>CPU
        +  IORDEN*IOGOS*IOPON          ;keep until IOGO ends
```

Since IOC.IORQFF- leads to the generation of BP.IORQ- by way of an open-collector buffer, the absence of the IOC's BP.IORQ- is caused by IOC.SPECIORQFF- in the first product term. The pending IORQ flag, IOC.IORQPD- is also disabled by IOC.SPECIORQFF- (product term 1) because that may lead to the generation of IOC.IORQFF- (IOC.IORQFF- product term 2). The IOC's data-in register buffers are also disabled because IOC.IORDEN- is not generated as a result of shutting down IOC.IORQFF- (product term 1). For regular OT* instructions, IOC.IORDEN- is true during the control word handshake, causing the data bus to be driven with the control word.

The absence of the IOC in the CPU backplane I/O handshaking does not prevent it from informing the EXT about the status and timing of that handshake. IOC.IOHS2-, IOC.IOHS3-, and IOC.ACKIOHS+ are generated as if the IOC were performing the handshake.

```
--- PAL equations for IOC.IOHS2-, IOC.IOHS3-, and IOC.ACKIOHSFF+
IOHS2 := IORQFF*IOGOS*/IOGOSL*IOPON    ;start IOGO counter with IOGO
       + SPECIORQFF*IOGOS*/IOGOSL*IOPON ; for IORQFF or SPECIORQFF

IOHS3 := IOHS2*IOPON                   ;3rd IO Handshake cycle

/ACKIOHSFF := /IOPON                   ;ACKIOHSFF+=0 at power-up
           + /ACKIOHSFF*/IOGOS         ;ACKIOHSFF+=1 if 1st cycle
           + /ACKIOHSFF*IOGOSL         ; of IOGO and
           + /ACKIOHSFF*/IORQFF*/SPECIORQFF
                                       ; IORQFF/SPECIORQFF set
           +  ACKIOHSFF                ;one cycle pulse only
```

IOC.SPECIORQFF- replaces IOC.IORQFF- for generating IOC.IOHS2- (product term 2), which causes IOC.IOHS3- to be asserted. Since IOC.IOHS3- is used to generate IOC.DOLCH+ and IO.DATAAV-, the handshaking data and cable message are automatically obtained in a method identical to the normal OT* case. IOC.ACKIOHSFF+ is also generated with IOC.SPECIORQFF- replacing the nonexistent IOC.IORQFF-.

The IOC card generates IOC.ACKIOHS- and IOC.DATAAV-, and obtains CPU backplane data at the end of both handshakes. The EXT must sort out these control signals and perform an extender backplane handshake somewhat different than the regular OT* handshake.

While the IOC card is synchronizing itself to a handshake that it did not create, the EXT card is broadcasting the OT* 0/2 instruction and waiting for an I/O handshake request to come back. All I/O cards on the extender backplane are expected to participate in this handshake. The EXT card does not acknowledge the request until it receives the IO.ACKIOHS- message from the IOC. This allows the subsequent handshakes to operate in synchronization.

```
--- PAL equation for EXT.IOGOFF-
IOGOFF := /IOGOFF*QIORQ*FIRSTHS*/SPECIO*/SLVWAIT*BPONL
                                  ;1st HS SF*/OT*[not 0/2]/LI*
        + /IOGOFF*QIORQ*SPECIO*ACKIOHSS*/INSTRAVS*BPONL
                                  ;1st/2nd HS OT*[0/2]
        + /IOGOFF*QIORQ*FIRSTHS*SLVWAIT*ACKIOHSS*/INSTRAVS*BPONL
                                  ;5th HS HLT/break
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*IOCWOT*OTAFLAG*ACKIOHSS
                 */INSTRAVS*BPONL  ;2nd HS OT*[not 0/2]
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*/IOCWOT*ACKIOHSS
                 *INSTRAVS*BPONL   ;2nd HS LI*
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*/IOCWOT*IOGOPD*BPONL
                                  ;2nd HS LI* queued
        +  IOGOFF*/IOGO3*BPONL     ;IOGO until IOGO3 set
        +  IOGOFF*IOGO3*IORQL*BPONL  ;IOGO for extended IORQ
```

The EXT card waits until EXT.ACKIOHSS+ is received (product term 2). Product terms 1, 4, 5, and 6 are disabled by the assertion of EXT.SPECIO-, while product term 3 is disabled by the deassertion of EXT.SLVWAIT-. EXT.SLVWAIT- is only true between certain even-numbered and odd-numbered handshakes in a multiple I/O handshake situation, such as break/HLT slave mode transfers. Therefore, EXT.SPECIO- and EXT.SLVWAIT- are mutually exclusive events.

Once EXT.IOGOFF- has been generated, the extender backplane handshake is properly synchronized with the CPU backplane handshake, the former tracking the latter. An invalid control word is obtained by the EXT at the end of the first handshake. The IOC eventually gets that control word, but it is discarded. Data similarly obtained during the second handshake is also discarded. Since OT* 0/2 instructions are data output instructions, data obtained by the IOC during the second handshake on the CPU backplane is placed on the extender backplane during the second handshake there.

The mechanism for performing the bus reversal is again based on the state of EXT.OTAFLAG-.

```
--- PAL equation for EXT.OTAFLAG-
OTAFLAG := /OTAFLAG*IOGOFF*FIRSTHS*/SPECIO*IOCWOT*ACKIOHSS
           */INSTRAVS*BPONL
         + /OTAFLAG*/IOGOFF*IORQL*/FIRSTHS*/SPECIO*IOCWOT:ACKIOHSS*
           /INSTRAVS*BPONL            ;for OT*[not 0/2]
                                      ; 1st ACKIOHSS sets OTAFLAG
                                      ; 2nd ACKIOHSS sets IOGOFF
         + /OTAFLAG*IOGOFF*IOGO3*/IORQL*FIRSTHS*SPECIO*BPONL
                                      ;OT*[0/2] means auto OTAFLAG
         +  OTAFLAG*/IOGO3*BPONL      ;OTAFLAG until IOGO3 set
         +  OTAFLAG*IOGO3*IORQL*BPONL  ;OTAFLAG for extended IORQ
```

Unlike the normal OT* case, bus reversal cannot depend on the two bits of the control word. The I/O cards are not supplying the control word because the operation is implicit in the instruction. Product terms 1 and 2 are disqualified with the presence of EXT.SPECIO-. EXT.SPECIO- performs this implicit control word function, causing EXT.OTAFLAG- to be generated automatically at the end of the first (control word) handshake.

The HLT Case

A HLT instruction forces the CPU to stop processing and go into a front panel mode. Since the front panel for the A-Series processors is virtual rather than physical CPU hardware, a branch to Virtual Control Panel software must occur. This software communicates with a designated I/O interface card and gives it the capability to alter CPU registers, memory locations, I/O card registers, and to invoke standard boot loaders. This software resides in Read Only Memory (ROM), which is in a different address space than the operating system and application software. Therefore a direct jump into the Virtual Control Panel program is not possible.

The A-Series processors support two mutually exclusive address spaces distinguished by the status of the ROM mode bit in the processor. When in ROM mode, execution and memory referencing occurs out of "boot" memory ROM and "boot" scratch RAM. Cross mapping instructions allow access to normal system memory RAM. "Boot" memory ROM contains the processor, memory controller, memory array, and I/O pretests in addition to the standard boot loaders and the Virtual Control Panel (VCP) program. Execution in non-ROM mode disallows references to the ROM mode address space.

The HLT instruction provides the mechanism to transfer execution from non-ROM mode addressing to ROM mode addressing. The instruction provides the means for saving the current value of the Program Counter (PC), loading a new value for the PC, and forcing execution in the ROM mode address space. As with other handshaking I/O instructions, HLT places the CPU into slave mode to send it control words and to transfer data. The I/O card performing the HLT instruction requests five handshakes that are grouped into three operations :

#1 (1a).	Control word handshake	: output current PC value
#2 (1b).	Data transfer handshake	: PC value stored in IOP
#3 (2a).	Control word handshake	: load new PC value
#4 (2b).	Data transfer handshake	: new PC value loaded into CPU
#5 (3).	Control word handshake	: enable ROM mode

The three operations are very similar to chaining OT* type handshakes with LI* type handshakes and finally terminated with an SFS type handshake.

In fact, the I/O extender treats multiple handshaking cases as a series of chained handshakes. Note that in all odd-numbered handshakes a control word is transferred to the CPU, and that in all even-numbered handshakes data is transferred in the direction indicated by the previous control word. Because the I/O extender is a pipelined-based design, chaining several handshakes together requires that proper synchronization needs to be maintained at all times. The synchronization between the control word handshake and the data transfer handshake has been discussed. What remains is the linkage synchronization between the even-numbered handshakes (data transfer) and the odd-numbered handshakes (control word).

Only one card in the entire system has the capability to execute the HLT instruction -- it is the interface card whose break mode switch (VCP) is set. Normally, that would be the system console terminal interface, but may also be a distributed systems networking interface to a remote computer. It is common practice for this card to reside in the CPU box, but the I/O extender allows the use of the VCP-enabled interface card on the extender backplane. This is an unsupported configuration because an extender box power failure destroys the RomP value previously loaded into register [3,C] of the VCP-enabled interface.

When a VCP-enabled interface card resides in the extender box and a HLT instruction is broadcast, that card performs the requisite five handshakes to pass into ROM mode and begin execution of the VCP routine in ROM. The first two handshakes are handled by the I/O extender as a regular OT* type double handshake. Since the data transfer handshake on the extender backplane is delayed until the corresponding handshake on the CPU backplane has finished, the next control word handshake (3rd HS) starts immediately following the completion of the data transfer handshake. The second pair of handshakes (3rd & 4th HS) are treated as an LI* type of handshake. Because the data transfer during the fourth handshake is for data input, the fifth handshake on the extender backplane needs to be delayed until the EXT receives assurance from the IOC that it is safe to do another input transfer (control word to the CPU). As before, this assurance comes in the form of EXT.ACKIOHSS+.

A flag named EXT.SLVWAIT- forces EXT.IOGOFF- to wait for the next EXT.ACKIOHSS+. It is generated within the EXTP2 I/O handshaking PAL [EX1005@15C].

```
--- PAL equation for EXT.SLVWAIT-
SLVWAIT := /SLVWAIT*IOGOFF*IOGO3*/IORQL*/FIRSTHS*/OTAFLAG*BPONL
                                            ;HLT/break
                                            ; multiple handshake sync
         + SLVWAIT*/IOGOFF*/INSTRAVS*SCHOD*BPONL ;till IOGOFF set
                                            ; or next I/O instruction
                                            ; or next slave acknowledge
```

Product term 1 causes an EXT.SLVWAIT- assertion at the end of the even-numbered (EXT.FIRSTHS- false) handshake (EXT.IOGOFF- true, EXT.IOGO3- true, and EXT.IORQL+ false) that passed data to the CPU (EXT.OTAFLAG- false). The second product term forces EXT.SLVWAIT- to clear after the delayed EXT.IOGOFF- is set. In addition, to guarantee a proper environment for the start of new I/O handshakes following another I/O instruction or break/slave mode processing, EXT.SLVWAIT- is also cleared by EXT.INSTRAVS+ or XP.SCHOD-.

EXT.SLVWAIT- alters the operation of EXT.IOGOFF-.

```
--- PAL equation for EXT.IOGOFF-
IOGOFF := /IOGOFF*QIORQ*FIRSTHS*/SPECIO*/SLVWAIT*BPONL
                                          ;1st HS SF*/OT*(not 0/2)/LI*
        + /IOGOFF*QIORQ*SPECIO*ACKIOHSS*/INSTRAVS*BPONL
                                          ;1st/2nd HS OT*[0/2]
        + /IOGOFF*QIORQ*FIRSTHS*SLVWAIT*ACKIOHSS*/INSTRAVS*BPONL
                                          ;5th HS HLT/break
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*IOCWOT*OTAFLAG*ACKIOHSS*
          /INSTRAVS*BPONL                 ;2nd HS OT*[not 0/2]
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*/IOCWOT*ACKIOHSS*
          /INSTRAVS*BPONL
        + /IOGOFF*IORQL*/FIRSTHS*/SPECIO*/IOCWOT*IOGOPD*BPONL
                                          ;2nd HS LI*
        +  IOGOFF*/IOGO3*BPONL            ;IOGO until IOGO3 set
        +  IOGOFF*IOGO3*IORQL*BPONL       ;IOGO for extended IORQ
```

When EXT.SLVWAIT- is set, only the third product term allows EXT.IOGOFF- to be asserted (product term 2 deals with OT* 0/2 instructions while product term 3 deals with HLT instructions). For an odd-numbered handshake (control word handshake, EXT.FIRSTHS- true) following an input data transfer handshake (EXT.SLVWAIT- true), product term 3 forces the next EXT.IOGOFF- to wait for the next EXT.ACKIOHSS+ from the IOC.

The delayed handshake is the control word handshake for passing the enable ROM mode command. Once that handshake completes, the HLT instruction is finished, and the CPU finds itself with a new PC value pointing to the start of the VCP routine in ROM mode memory. Its original PC value is safely contained in the Psave register of the I/O processor chip of the VCP-enabled interface card.

The User-Defined I/O Instruction Case

It is possible for an experienced designer to invent a custom-designed interface which responds to regular I/O instructions in an unconventional manner. For instance, an OTA 40 instruction may be a command for that interface to obtain the processor registers, perform some form of computation based on those values, then place the results back into the CPU registers. This is possible with I/O handshake chaining, similar to the method used for HLT instruction processing. The I/O extender is designed to allow handshake chaining because of the pipeline synchronization performed between handshakes.

DMA Contention with I/O Handshaking

The discussion of I/O handshaking assumed the absence of DMA contention. DMA activity on the extender backplane is shut down prior to the instruction broadcast using XP.CPUTURN-. XP.CPUTURN- is also low during the handshaking process, and if the handshake is extended, so is XP.CPUTURN-.

XP.CPUTURN- is generated by summing one CPUTURN component from the EXT and another CPUTURN component from the IOC. The EXT contribution comes from the EXT.CPUTURNFF- flip-flop of the EXTP1 PAL [EX1004@15B]. The IOC portion is the combination of IOC.CPUTURNFLG- from the IOCP3 PAL [IO1105@15C], and IOC.IORQFF- from the IOCP2 PAL [IO1004@15B].

The EXT.CNTR1- and EXT.CNTR0- counters of the EXTP1 PAL provide for four cycles of XP.CPUTURN- with every XP.RNI- assertion. If the EXTP1 PAL does not receive EXT.IORQL+ by the end of the fourth count, XP.CPUTURN- goes high and DMA is allowed to resume.

```
--- PAL equation for EXT.CPUTURNFF-
CPUTURNFF := INSTRAV*BPONL             ;set CPUTURNFF to prevent
                                       ; interference from DMA
                                       ; during an instr. broadcast
           + CPUTURNFF*/CNTR0*BPONL    ;keep CPUTURN for at least
           + CPUTURNFF*/CNTR1*BPONL    ; three cycles after RNICYC
           + CPUTURNFF*IORQL*BPONL     ;or when IORQL comes around
           + DMAWAIT*BPONL             ;set when DMAWAIT is true
                                       ; (INTRQLL+=1 or IOGOFF-=0)
```

Assume that an I/O handshake is desired and that EXT.IORQL+ is received while EXT.CPUTURNFF- is still set (product term 4). As long as the I/O card is requesting an I/O handshake, XP.CPUTURNFF- remains set. This scheme is incomplete because XP.CPUTURNFF- goes high at the end of the first I/O handshake, but product term 4 is unable to assert EXT.CPUTURNFF- again. That job is left up to product term 5, which is able to force EXT.CPUTURNFF- to set.

EXT.DMAWAIT- is generated by the EXTP3 PAL [EX1105@15D].

```
--- PAL equation for EXT.DMAWAIT-
DMAWAIT = IOGOFF*BPONL                 ;lengthen CPUTURN for IOGO
        + INTRQL*BPONL                 ;CPUTURN during INTRQLL
```

The presence of either EXT.IOGOFF- or EXT.INTRQL+ makes EXT.DMAWAIT- true. Note that EXT.DMAWAIT- is a combinatorial output; it changes immediately with changes on the input pins.

Since EXT.IOGOFF- is low during the one cycle that IOC.IORQL+ is low between I/O handshake requests, and EXT.IOGOFF- also remains low for a few cycles after the last I/O handshake request goes away, XP.CPUTURNFF- is set prior to the start of the initial I/O handshake request to the completion of the last I/O handshake acknowledge.

Because of the several clock-cycle skew (introduced by the extender's two-stage pipeline) between the I/O handshakes on the extender backplane and the corresponding handshakes on the CPU backplane, it is possible for DMA to start up again after the last handshake completes on the extender, but before the corresponding handshake request is acknowledged on the CPU backplane. If the DMA cycle proceeds, the IOC card's data-in register is overwritten with the DMA transfer's data. Since the DMA state machine runs without considering the state of the I/O handshake state machine on the IOC card, the IOC card makes the DMA request. This compounds the problem because DMA activity has priority over I/O handshake acknowledgements (BP.IOGO- assertion). When the IOC's I/O handshake request is finally honored and acknowledged, the data-in register no longer contains the original data required to complete the handshake. This situation is prevented by having the IOC contribute to the EXT's assertion of XP.CPUTURN-. If the IOC card is making an I/O handshake request, or if an I/O handshake is currently in progress, the IOC makes its IO.CTURNF- cable message true (low). A gate [IO0705@17D] combines IOC.IORQFF- with IOC.CPUTURNFLG- to generate IOC.CTURNF+.

```
--- PAL equation for IOC.CPUTURNFLG-
CPUTURNFLG = /A900MODE*/S2*S1          ;A600+: CPUTURNFLG-=0
           + /A900MODE*/S2*S0          ; at State=1..3
           +  A900MODE*/S2             :A900 : CPUTURNFLG-=0
           +  A900MODE*/S1             : at State=2..7
           +  A900MODE*/S0             :
           +  IOGOSL                   ;both: tracks IOGOSL and
                                       ; IORQFF (externally)
```

Only the sixth product term is of interest for keeping XP.CPUTURN- asserted during I/O handshaking. IOC.IORQFF-, together with IOC.IOGOSL-, makes IO.CTURNF- cover the entire time span from the initial I/O handshake request on the CPU backplane to the completion of the last I/O handshake.

Using the IORQ and IOGO signals on both the IOC and EXT cards, XP.CPUTURN- is kept low until all I/O handshaking has completed on both backplanes. This arbitration method prevents the registers on the IOC card from being accidentally overwritten. Therefore, all DMA activity on the extender backplane remains suspended until all I/O handshaking is complete.

Another source of contention is DMA activity on the CPU backplane while an I/O handshake between the CPU and IOC is in progress. This type of conflict is resolved using the regular backplane arbitration rules. While BP.IOGO- is low, DMA is suspended. If BP.IOGO- is asserted by the CPU as a DMA request is initiated, the CPU withdraws BP.IOGO- to yield the backplane for DMA use. While the IOC card is making an I/O handshake request with the assertion of BP.IORQ-, DMA activity on the CPU backplane may delay the CPU's assertion of BP.IOGO-. The I/O extender protocols account for this delay by delaying the IOC's message for the EXT, IO.ACKIOHS+. A lot of DMA activity on the CPU backplane merely lengthens the execution of the I/O handshakes for the I/O extender. If the handshakes are delayed on the CPU backplane, that usually leads to delayed handshakes on the extender backplane. An obvious exception is the single handshake SFS/SFC instruction. The extender backplane I/O handshake completes quickly regardless of the delay experienced by the IOC.

Computer Museum

Performing Direct Memory Access

Another major I/O interface card activity is direct memory access (DMA). Programmed I/O instructions may be used to transfer data between the CPU and an I/O interface, but this technique requires full utilization of the CPU resources to perform the transfer. With DMA techniques, the CPU only uses several I/O instructions to set up the DMA transfer, then passes control for the actual data movement to hardware on the I/O interface card.

The DMA Priority Chains

Unlike programmed I/O instructions for which the CPU always maintains control, and only one I/O interface is addressed, all I/O interface cards can perform DMA simultaneously. A centralized priority arbitrator does not exist which can resolve conflicts between two or more I/O cards that desire DMA transfers at the same time. A decentralized technique is used for resolving priority conflicts on the A-Series I/O backplane.

A high priority card currently requesting a DMA transfer disables all lower priority I/O cards from starting a new memory access at the start of the next clock cycle. Once the highest priority request is processed without another consecutive request from that interface, the next lower priority requesting interface assumes the role of the "highest priority requesting interface" for the next clock cycle. Every I/O card receives the BP.MCHID- DMA priority chain disable signal sent to it from the next higher priority interface card. Each I/O card generates the BP.MCHOD- DMA priority chain disable that is sent to the next lower priority interface card. The BP.MCHOD- output of the higher priority card is the same signal as the BP.MCHID- input of the next lower priority card.

The highest priority interface card in the CPU backplane is always enabled to perform DMA. If it wishes to do DMA, it asserts BP.MCHOD- to the card below. The second highest priority card is obligated to pass along this disable condition even if it does not desire to perform DMA itself. The I/O extender's IOC card conforms to this protocol, using an AND gate [I00805@19A] to generate and pass priority disabling conditions. If BP.MCHID- is low coming onto the IOC, BP.MCHOD- is also low leaving the IOC. If BP.MCHID- is high entering the IOC card, BP.MCHOD- is also high unless the IOC has its IOC.MRQFF- set, indicating that it wishes to perform DMA.

Borrowing from advanced processor design methods, the priority chains, and I/O architecture as well, are entirely pipelined when I/O extenders are used. Each I/O extender's EXT card plays the role of CPU/Memory, and is responsible for up to 19 I/O cards. DMA (as well as interrupt and slave mode) priority among the 19 I/O cards in the extender box is resolved every backplane clock cycle. One DMA transfer from the highest priority requesting card is performed and then its operands (address/data) are transferred to the IOC card. At the start of the next CPU backplane clock cycle, the IOC card fights for priority on the CPU backplane. Therefore, every DMA access by an I/O interface card in the extender backplane undergoes two levels of priority arbitration, initially among all I/O cards in the extender box, then again with all I/O cards on the CPU backplane. With this I/O extension scheme, many more I/O cards may be accommodated without affecting the basic CPU microcycle time, thereby preserving the CPU's computational performance and minimally affecting its I/O performance.

DMA Activity

All DMA activity is initiated by the I/O card, communicating directly with the memory subsystem without CPU intervention. Once a DMA request has been initiated, any bus contention with the CPU results in the processor deferring to the DMA transfer. There are three types of DMA transfers: DMA reads from memory, DMA writes to memory, and DMA self-configuration (a special version of the DMA read type).

All DMA transfers are initiated by the I/O card with the assertion of XP.MRQ-. During the first cycle of XP.MRQ- low, DMA priority is resolved, and the highest priority DMA requestor asserts XP.MEMGO- during the next extender backplane clock cycle. A memory address is also sent to the EXT card during XP.MEMGO-. If the DMA transfer involves a write into memory, data is also passed to the EXT along with the address. For memory reads, the extender backplane is locked into a busy state with the assertion of XP.BUSY- until the requested data has been obtained by the IOC and sent back to the EXT.

The IOC is responsible for recreating the DMA transfer on the CPU backplane using the address (and data, if applicable) sent to it from the EXT. For memory writes, once the information has been accepted by the memory controller, the IOC's job is done. In a memory read request, the IOC waits until data is returned by the memory controller, then passes that data to the EXT to complete the DMA read handshake on the extender backplane.

A DMA self-configuration is handshaken as a series of DMA reads. Instead of passing the returned data to the I/O card's interface logic, the data is used to load the IOP (I/O processor) DMA control registers which describe the current DMA configuration: DMA control word, interface control word (optional parameter), DMA address, and DMA word/byte count. This operation is completely transparent to the I/O extender and is handled as a series of DMA reads.

DMA Reads from Memory - Normal Case

A DMA read cycle is initiated with the assertion of XP.MRQ-, which is received by an inverting buffer [EX1107@11A] on the EXT board to create the buffered version, EXT.MRQB+. XP.MRQ- is terminated with a resistor network [EX1007@10B] to provide the necessary collector resistance for the open-collector XP.MRQ- drivers on the I/O cards.

EXT.EXTGO+ is generated by a NOR gate [EX1205@13A] combining EXT.MRQB+ and EXT.BUSYL+. EXT.EXTGO+ qualifies EXT.IOGOFF+ and EXT.IAKFF+ at two NAND buffers [both at EX0906@19C]. EXT.EXTGO+ is normally true (high) prior to the assertion of both EXT.IOGOFF+ and EXT.IAKFF+. EXT.EXTGO+ is only used to handle a DMA request which is initiated at the same time that the EXT wants to generate the first XP.IOGO- in response to the first XP.IORQ- following a slave acknowledgement. The DMA transfer is processed first. Shortly after EXT.IOGOFF- is set, EXT.CPUTURNFF- is also set to disable further DMA activity during the I/O handshaking.

A time-shifted version of EXT.MRQB+ is generated by a D-type flip-flop [EX1204@13C]. EXT.MRQS+ is synchronized to the start of the short half cycle of XP.SCLK-. As with EXT.EXTGO+, EXT.EXTGOS+ is generated by a NOR gate [EX1205@13A]. EXT.EXTGOS+ informs the EXTP2 and EXTP3 PALs about a delayed XP.IOGO-. EXT.MRQS+ serves two functions. First, it postpones the issuance of XP.CRS- on the extender backplane when XP.MRQ- is low. If XP.CRS- resets an active I/O interface while it is performing DMA, the address and/or data bus may be altered, thus writing incorrect data into a memory location. Secondly, EXT.MRQS+ is further time shifted to the next clock edge by a D-type flip-flop [EX0504@17D], creating EXT.MRQSL+, which is immediately buffered with an inverter [EX0604@19D] onto the cable as EX.REQDMA-.

The EX.REQDMA- message to the IOC card tracks XP.MRQ- with a delay of one clock cycle. Unlike most of the other cable messages, EX.REQDMA- is not a pulse with a duration of one clock cycle. (See Figure 12-6.) If XP.MRQ- is kept low between two DMA transfers on the extender backplane, the intent is to keep BP.MRQ- low between two IOC DMA transfers as well.

As with the other cable messages received by the IOC, EX.REQDMA- is terminated with a resistor network [IO0702@10D] to provide default conditions if the cable is inadvertently removed. An inverting buffer [IO0904@12D] conditions the signal and generates IOC.REQDMA+. The IOCP1 DMA Control PAL [IO0905@15A] uses IOC.REQDMA+ to generate IOC.MRQPD- (pending MRQ) and IOC.MRQFF- (current MRQ).

```
--- PAL equation for IOC.MRQFF-
MRQFF := REQDMA*/IAKB*/IOGOS*IOPON      ;REQDMA : start now or use
       + MRQPD*/IAKB*/IOGOS*IOPON       ; MRQPD when OK to start
       + MRQFF*/MEMGOFF*/MEMGOPD*REQDMA*IOPON
                                        ;remain asserted until MEMGOFF
                                        ; or MEMGOPD set (normal case),
                                        ; or until REQDMA goes away
                                        ; (aborted DMA case [PE])
       + MRQFF*/MEMGOFF*MEMGOPD*IOPON   ;extend MRQFF when MEMGOPD
                                        ; & set until MEMGO issued
       + MRQFF*MEMGOFF*MEMGOPD*MEMGOQUEUE*IOPON
                                        ;extend MRQFF beyond MEMGO
                                        ; if MEMGOQUEUE is set
       + MRQFF*MEMGOFF*BUSYL*IOPON      ;A600+ CPU preemptive memory
                                        ; access during first cycle
                                        ; of MRQ- (recovery mode)
                                        ;A900 CPU early VALID
                                        ; handshake recovery
                                        ; (recovery mode)
```

In the simplest case, when the CPU backplane is not preoccupied with an interrupt or I/O handshake acknowledgements, product term 1 allows IOC.MRQFF- to be asserted immediately upon the receipt of IOC.REQDMA+. Product term 3 keeps IOC.MRQFF- set until the IOC.MEMGOFF- is asserted, indicating that the DMA transfer has been attempted. Alternately, if IOC.REQDMA+ goes away when IOC.MEMGOPD- (MEMGO pending) has not yet set (IOC is not holding a pending DMA transfer), IOC.MRQFF- also deasserts. This unusual situation only occurs if a parity error is encountered during one of the control register transfers of a DMA self-configuration. This situation is described in detail later, but for now it suffices to know that IOC.REQDMA+ starts off IOC.MRQFF-, which is normally self-terminating, or in one particular situation, the end of IOC.REQDMA+ also terminates IOC.MRQFF-.

In a typical one-word DMA transfer, XP.MRQ- is low for only two clock cycles. Likewise, IOC.REQDMA+ lasts two cycles also. Using the scheme above, if IOC.MRQFF- cannot be generated within those two cycles due to overriding backplane activities (interrupt acknowledgement and I/O handshake acknowledgement), it is conceivable that the action intended by IOC.REQDMA+ may be forever lost. To prevent that from happening, another flag flip-flop is used to log pending memory requests which are preempted. IOC.MRQPD- is programmed to set whenever IOC.REQDMA+ is received during an interrupt acknowledgement or during an I/O handshake acknowledgement.

```
--- PAL equation for IOC.MRQPD-
MRQPD := REQDMA*IAKB*IOPON            ;REQDMA : hold DMA if
       + REQDMA*IOGOS*IOPON           ; IAK and IOGO in progress
       + MRQPD*IAKB*IOPON             ;remain held up until
       + MRQPD*IOGOS*IOPON            ; IAK and IOGO go away
```

Product terms 1 and 2 activate IOC.MRQPD- while product terms 3 and 4 serve to clear the pending MRQ flag after the preemptive situations go away. Going back to the PAL equation for IOC.MRQFF-, product term 2 is used to remind the IOC.MRQFF- that it should set now because IOC.REQDMA+ was previously received during a time when IOC.MRQFF- could not have been set.

When IOC.MRQFF- goes low, BP.MRQ- is generated and the DMA chain is disabled for all I/O cards with a lower backplane priority. IOC.MRQFF- is combined with two other MRQ-producing signals at a gate [IO0707@18A] to generate IOC.MRQ+, which drives the NAND gate open-collector BP.MRQ- buffer [IO1107@19A]. IOC.MRQFF- causes BP.MRQ- to go low for a legitimate DMA transfer, whereas IOC.EXTNRDY- and IOC.PROCHLDL- cause BP.MRQ- to go low to keep the CPU off of the backplane for arbitration. IOC.MRQFF- is also received by an AND gate [IO0805@19A] which generates the BP.MCHOD- DMA priority chain signal for the card immediately below the IOC on the CPU backplane. In addition, a DMA priority chain look-ahead signal is also generated using a NAND gate open-collector buffer [IO1107@19A]. Its input is IOC.MRQFF+, generated by running IOC.MRQFF- through an inverter [IO1007@18A]. With the assertion of BP.MRQ- and the simultaneous disabling of the DMA chain for lower priority DMA channels, the extender is ready to perform the actual DMA transfer.

Meanwhile, there is also activity on the extender backplane. One cycle after XP.MRQ- went low, the highest priority DMA channel asserts XP.MEMGO- and simultaneously places a DMA read address on the extender backplane. As with XP.MRQ-, XP.MEMGO- is also terminated with a resistor network and received into an inverting buffer [EX1107@11A] to create EXT.MEMGOB+. There are two users of EXT.MEMGOB+: the BUSY flip-flop [EX1006@11B] and the EXTP1 DMA Control PAL [EX1004@15B]. The BUSY flip-flop is always set when an EXT.MEMGOB+ pulse is received, and only resets when EXT.VALIDFF- is set within the EXTP1 PAL. For a DMA read, EXT.BUSYFF+ remains true for at least four clock cycles before resetting. An EXT.DMACYC- flip-flop is also set in the EXTP1 PAL (product term 1) with EXT.MEMGOB+, and lasts until the IOC has acknowledged that the corresponding DMA request has been successfully launched on the CPU backplane (product term 2).

```
--- PAL equation for EXT.DMACYC-
DMACYC := MRQB*MEMGOB*BPONL           ;starts DMA cycle
        + DMACYC*/ACKDMA*BPONL        ;stay DMA mode until ACKDMA
        + DMAQUEUE*BPONL              ;DMA queue is two deep
```

The rising edge of EXT.BUSYFF+ is used by a set of octal registers [EX0107@24A, EX0307@24B, EX0207@24C] to store the contents of the address bus (XP.AB+[0..14]), the address extension bus (XP.AE+[0..4]), the write enable bit (XP.WE-), and the DMA self-configuration bit (XP.SELFC-). According to the backplane specifications, the addressing information must be sampled at the start of the short half cycle during XP.MEMGO-. The protocols for the extender cable permit signal transitions for incoming address and data only at the start of the long half cycle. As mentioned previously, another set of registers [EX0102@25A, EX0303@25B, and EX0103@25C] clocked with EXT.ABDBAV+ is used to shift the transition of the cable's addressing information to the start of the long half cycle. Note that the rising edge of EXT.ABDBAV+ occurs one half cycle after the rising edge of EXT.BUSYFF+, because the former is derived from the latter by way of EXT.DICLK+. The tri-stated data bus is also sampled, but the IOC board knows that the incoming data is invalid during a DMA read.

As the addressing information is sampled by the EXT board, XP.WE- is singled out for additional processing. It is inverted by a buffer [EX0807@23E] to form EXT.WEB+. EXT.WEB+, and its clocked version, EXT.WRITE+, are sent to inform the EXTP1 DMA control PAL about the directionality of the DMA transfer. Since EXT.WRITE+ is an output of the WRITE flip-flop [EX0706@24C] that is clocked with EXT.BUSYFF+, its state is valid until the next DMA transfer (or interrupt trap cell fetch). EXT.WEB+ and EXT.WRITE+ are both false for DMA read operations from memory. This indicates that EXT.VALIDFF- must delay its assertion (see product term 1 below) until EXT.DATAAV+ is true, a sign that the IOC has performed its duty and valid DMA read data is available on the data-out bus of the cable.

```
--- PAL equation for EXT.VALIDFF-
VALIDFF := /WRITE*/DMACYC*BUSYL*DATAAV*BPONL
                                        ;rd or int: DATAAV
                                        ; received when DMACYC
                                        ; flag is clear
        + WEB*MRQB*MEMGOB*/DMACYC*BPONL
                                        ;wr: valid immediately if
                                        ; no DMACYC in progress
        + WEB*MRQB*MEMGOB*DMACYC*ACKDMA*BPONL
                                        ;wr: valid immediately if
                                        ; DMACYC but ACKDMA received
                                        ; for first DMA write
        + WRITE*DMACYC*BUSYL*ACKDMA*BPONL
                                        ;wr: valid when ACKDMA
                                        ; received for last request
                                        ; (when DMACYC FF is set)
```

With EX.REQDMA- sent off earlier, EX.ABDBAV- and the addressing information being sent off in the current cycle, the EXT board goes into a wait mode (with XP.BUSY- asserted on the extender backplane to signal that "memory" is busy) until the IOC has obtained the DMA read data. The IOC converts EX.ABDBAV- into IOC.ABDBAV+, which is immediately used by the IOCP1 DMA Control PAL to create the IOC.ABDBCLK+ clock signal for the addressing and data-in registers.

```
--- PAL equation for IOC.ABDBCLK+
/ABDBCLK := /IOPON                          ;ABDBCLK+=0 at power up
          + /MEMGOPD*/ABDBAV                ;if MEMGOPD not set, stay
                                            ; at 0 until ABDBAV true
          +  MEMGOPD*MRQFF*/MEMGOFF         ;if MEMGOQUEUE and MEMGOPD
          +  MEMGOPD*MRQFF* BUSYL           ; are set, pulse ABDBCLK hi
          + /MEMGOQUEUE*MEMGOPD*MRQFF*/ABDBAV
                                            ; immediately after current
                                            ; good MEMGO
                                            ;if only MEMGOPD set, pulse
                                            ; ABDBCLK after current good
                                            ; MEMGO only if new data is
                                            ; on the cable (ABDBAV)
                                            ;if current MEMGO is no
                                            ; good (A900 extended BUSY)
                                            ; then MEMGOQUEUE is set,
                                            ; forcing data pickup
                                            ; off the cable at the end
                                            ; of a good MEMGO.
```

IOC.ABDBCLK+ pulses high for one clock cycle immediately upon the receipt of IOC.ABDBAV+ (product term 2). Like the I/O handshaking case, the data-in queue is empty, meaning that the addressing and data-in registers may load the new information off the cable immediately. The addressing registers [IO0106@24A, IO0107@24B, IO0307@24C, and IO0207@24C] are clocked at the same time as the data-in registers.

IOC.MEMGOFF- is also generated from IOC.ABDBAV+.

```
--- PAL equation for IOC.MEMGOFF-
MEMGOFF := /MEMGOFF*MRQFF*/MCHID*/BUSYL*ABDBAV*IOPON
                                            ;ABDBAV received while memory
                                            ; is not busy
        + /MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*ABDBAV*IOPON
                                            ;ABDBAV received and OK
                                            ; to MEMGO on next cycle
                                            ; since this one is finishing
        + /MEMGOFF*MRQFF*/MCHID*/BUSYL*MEMGOPD*IOPON
                                            ;pending MEMGO, mem. not busy
        + /MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*MEMGOPD*IOPON
                                            ;pending MEMGO, OK next cycle
        + /MEMGOFF*INTRQFF*IAKB*/ICHID*ABDBAV*IOPON
                                            ;trap cell fetch immediately
        +  MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*IOPON
                                            ;recovery from A600+ CPU
                                            ; preemptive memory access
                                            ; occurring during first
                                            ; cycle of MRQ-
```

In the simplest case, IOC.MEMGOFF- is generated immediately after receiving IOC.ABDBAV+. Product term 1 performs this if IOC.MRQFF- is set for a DMA cycle (priority for lower cards disabled), BP.MCHID- is high (not disabled by a higher priority DMA channel), and IOC.BUSYL+ has been low (memory not busy). Product term 2 allows for a special case. If IOC.BUSYL+ and IOC.VALIDL+ are both true, the current memory cycle must be finishing, so the IOC predicts that the backplane will be available for use at the beginning of the next clock cycle.

IOC.ABDBAV+ only lasts one clock cycle, and if IOC.MEMGOFF- cannot be generated during that cycle, the DMA transfer could be lost. Similar in function to the IOC.MRQPD- flag, IOC.MEMGOPD- is set whenever IOC.ABDBAV+ is received and IOC.MEMGOFF- is not allowed to be set.

```
--- PAL equation for IOC.MEMGOPD-
MEMGOPD := MRQPD*ABDBAV*IOPON           ;if MRQFF not yet set
        + MRQFF*MCHID*ABDBAV*IOPON      ; or not yet EXT priority
        + MRQFF*BUSYL*/VALIDL*ABDBAV*IOPON
                                        ; memory cycle in progress
        + MRQFF*MEMGOFF*BUSYL*IOPON     ;A600+ CPU steals cycle
        + MEMGOPD*/MRQFF*IOPON          ;hold until MRQFF set
        + MEMGOPD*MRQFF*/MEMGOFF*IOPON ; and until MEMGO issued
        + MEMGOPD*MRQFF*MEMGOFF*/BUSYL*ABDBAV*IOPON
                                        ;queue up second DMA write
                                        ; as soon as first one has
                                        ; gone to memory
        + MEMGOQUEUE*IOPON              ;queue up second DMA write
```

In addition, there are several other reasons for IOC.MEMGOPD- to stand in for the absent IOC.MEMGOFF-. Product term 1 causes IOC.MEMGOPD- to set if IOC.MRQPD- is true. The assertion of IOC.MRQFF- must precede IOC.MEMGOFF- in order to establish the IOC as the highest priority DMA channel. Once IOC.MRQFF- is generated from IOC.MRQPD-, IOC.MEMGOFF- sets one cycle later as a result of IOC.MEMGOPD-, thus ensuring the proper protocol sequencing. Product term 2 sets IOC.MEMGOPD- in the event that IOC.ABDBAV+ is received when a higher priority DMA channel has disabled the IOC from performing DMA during the next clock cycle. Note that IOC.MRQFF- must be set before examining the incoming priority chain information. Product term 3 takes care of the situation in which IOC.ABDBAV+ is received while memory is busy (unless it is the last cycle of the memory access). Regardless of how IOC.MEMGOPD- is set, the flag flip-flop remains set until product terms 5 and 6 are no longer satisfied. Product term 5 allows IOC.MEMGOPD- to continue until IOC.MRQFF- is set. When that happens, product term 6 takes over and remains in effect until IOC.MEMGOFF- is set.

IOC.MEMGOFF- contains two product terms which employ the IOC.MEMGOPD- pending MEMGO status flip-flop.

```
--- PAL equation for IOC.MEMGOFF-
MEMGOFF := /MEMGOFF*MRQFF*/MCHID*/BUSYL*ABDBAV*IOPON
                                            ;ABDBAV received while memory
                                            ; is not busy
        + /MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*ABDBAV*IOPON
                                            ;ABDBAV received and OK
                                            ; to MEMGO on next cycle
                                            ; since this one is finishing
        + /MEMGOFF*MRQFF*/MCHID*/BUSYL*MEMGOPD*IOPON
                                            ;pending MEMGO, mem. not busy
        + /MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*MEMGOPD*IOPON
                                            ;pending MEMGO, OK next cycle
        + /MEMGOFF*INTRQFF*IAKB*/ICHID*ABDBAV*IOPON
                                            ;trap cell fetch immediately
        +  MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*IOPON
                                            ;recovery from A600+ CPU
                                            ; preemptive memory access
                                            ; occurring during first
                                            ; cycle of MRQ-
```

Product terms 3 and 4 are very similar to product terms 1 and 2, respectively. IOC.MEMGOPD- serves to remind IOC.MEMGOFF- that IOC.ABDBAV+ has occurred sometime in the past, that the addressing registers contain good information, and that the DMA transfer has not yet taken place. Although delayed, IOC.MEMGOFF- must be generated in order to proceed with the DMA transfer.

Once asserted, IOC.MEMGOFF- is further qualified with IOC.BUSYL+ at a gate [I00306@18A]. This is necessary because the IOCP1 PAL reacts to changes of BP.BUSY- 1.4 clock cycles after they occur on the CPU backplane. The qualified MEMGO, QMEMGO+ is true if IOC.MEMGOFF- is asserted while BP.BUSY- is false (high). If this is not the case, QMEMGO+ remains low even though IOC.MEMGOFF- is asserted. IOC.QMEMGO+ is the final step, since it is used by a NAND gate open-collector buffer [I00907@19B] to generate BP.MEMGO-. A memory cycle is started when BP.MEMGO- is brought low for one clock cycle.

There are two situations which can force the predictive logic for IOC.MEMGOFF- to go astray. Both cases involve processor modifications to the standard backplane protocols for CPU/DMA arbitration, or the handling of extended memory cycles. Normally, the first cycle of BP.MRQ- low is used for DMA priority resolution, but the CPU is not permitted to use the backplane due to the presence of BP.MRQ-. The A600+ CPU detects this priority resolution cycle and sneaks in an instruction fetch or operand read/write cycle. The IOC expects to assert BP.MEMGO- during the second cycle of BP.MRQ-, but memory became busy in this case. The second case involves the A900 CPU DMA write cycle that is extended because of concurrent memory refreshing. Normal protocol calls for a lengthened BP.BUSY-, terminated with both BP.BUSY- and BP.VALID- true during the last cycle. This protocol is inverted for the A900, with BP.BUSY- and BP.VALID- true during the first cycle, and only BP.BUSY- true during the remaining cycles.

Taking the preemptive memory access case first, the IOC expects to generate BP.MEMGO- during the second cycle of BP.MRQ-, but is not permitted due to the unexpected presence of BP.BUSY-. The DMA handshaking PAL handles this by using the fourth product term of IOC.MEMGOPD- to place the DMA state machine into a pending state to await the next possible occasion for a DMA transfer attempt.

```
--- PAL equation for IOC.MEMGOPD-
MEMGOPD := MRQPD*ABDBAV*IOPON            ;if MRQFF not yet set
         + MRQFF*MCHID*ABDBAV*IOPON      ; or not yet EXT priority
         + MRQFF*BUSYL*/VALIDL*ABDBAV*IOPON
                                         ; memory cycle in progress
         + MRQFF*MEMGOFF*BUSYL*IOPON     ;A600+ CPU steals cycle
         + MEMGOPD*/MRQFF*BPONL          ;hold until MRQFF set
         + MEMGOPD*MRQFF*/MEMGOFF*IOPON  ; and until MEMGO issued
         + MEMGOPD*MRQFF*MEMGOFF*/BUSYL*ABDBAV*IOPON
                                         ;queue up second DMA write
                                         ; as soon as first one has
                                         ; gone to memory
         + MEMGOQUEUE*IOPON              ;queue up second DMA write
```

When the A600+ preemptive memory access is extended to three or four cycles because of a concurrent memory refresh, the IOC.MEMGOPD- DMA queue allows IOC.MEMGOFF- to make another attempt at the completion of the preemptive memory cycle. Since IOC.MEMGOFF- has been delayed, backplane DMA priorities may have been reassigned. When allowed to proceed, IOC.MEMGOFF- is generated and IOC.MEMGOPD- is cleared one cycle later. If DMA priority is taken by a higher priority DMA channel, IOC.MEMGOFF- is inhibited from setting, causing IOC.MEMGOPD- to wait for another opportunity.

Usually, the A600+ CPU preemptive memory access causes only a one-cycle postponement of the DMA transfer because of the two-cycle memory access time. This may lead to the situation where the first IOC.MEMGOFF- attempt is made when IOC.BUSYL+ and IOC.VALIDL+ are both true, meaning that the CPU memory cycle is finishing. IOC.QMEMGO+ remains low because of the IOC.BUSYL+ mask. Since it takes two cycles to generate another IOC.MEMGOFF- by way of IOC.MEMGOPD-, the clock cycle immediately after the end of the preemptive memory access is wasted. IOC.MEMGOPD- is set in the usual fashion, but IOC.MEMGOFF- attempts an automatic retry to avoid the extra cycle of overhead, and therefore bypasses the normal route through IOC.MEMGOPD-.

```
--- PAL equation for IOC.MEMGOFF-
MEMGOFF := /MEMGOFF*MRQFF*/MCHID*/BUSYL*ABDBAV*IOPON
                                             ;ABDBAV received while memory
                                             ; is not busy
         + /MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*ABDBAV*IOPON
                                             ;ABDBAV received and OK
                                             ; to MEMGO on next cycle
                                             ; since this one is finishing
         + /MEMGOFF*MRQFF*/MCHID*/BUSYL*MEMGOPD*IOPON
                                             ;pending MEMGO, mem. not busy
         + /MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*MEMGOPD*IOPON
                                             ;pending MEMGO, OK next cycle
         + /MEMGOFF*INTRQFF*IAKB*/ICHID*ABDBAV*IOPON
                                             ;trap cell fetch immediately
         +  MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*IOPON
                                             ;recovery from A600+ CPU
                                             ; preemptive memory access
                                             ; occurring during first
                                             ; cycle of MRQ-
```

Product term 6 is the IOC.MEMGOFF- auto retry. If the IOC card still has priority, IOC.MEMGOFF- remains asserted during the next clock cycle. Since IOC.BUSYL+ is no longer true, BP.MEMGO- is generated. IOC.MEMGOPD- is also cleared when IOC.MEMGOFF- is issued in the absence of IOC.BUSYL+.

The second non-standard case involves the A900 processor's interpretation of the memory handshaking protocols during DMA write operations concurrent with memory refreshing. It handshakes with both BP.BUSY- and BP.VALID- asserted first, then keeps BP.BUSY- low for two or three more cycles after the deassertion of BP.VALID-. Suppose that IOC.MEMGOFF- is generated while IOC.BUSYL+ and IOC.VALIDL+ are both true. According to the automatic retry mechanism (product term 6), IOC.MEMGOFF- is automatically extended by one clock cycle. Assuming that the IOC still has DMA priority, the second cycle of IOC.MEMGOFF- still doesn't generate a BP.MEMGO- because of the extended IOC.BUSYL+. IOC.MEMGOPD- comes to the rescue again. If the IOC loses DMA priority after the first attempt, IOC.MEMGOPD- remembers that the DMA cycle has not been successfully executed.

Assuming that all of these detours are resolved, the IOC card eventually asserts BP.MEMGO-. When this happens, a message is sent to the EXT indicating that the DMA request has been accepted by the memory controller. IOC.QMEMGO+, which generates BP.MEMGO-, is inverted with a gate [IO1007@18B] to generate IOC.QMEMGO-. A gate [IO1106@18C] qualifies IOC.QMEMGO- with IOC.MRQFF- to produce IOC.ACKDMA+, which is inverted by a buffer [IO1204@19C] to create the IO.ACKDMA- cable message for the EXT. The IOC.MRQFF- component is necessary to restrict IOC.ACKDMA+ to DMA related MEMGOs, and not to include interrupt trap cell fetch MEMGOs.

The memory system immediately responds to BP.MEMGO- with BP.BUSY- to prevent new DMA cycles from being started by other DMA channels. In most cases, the memory system also simultaneously asserts BP.VALID- to indicate that valid data will appear on the backplane data bus on the rising edge of that control signal. Due to possible memory refreshing conflicts, it is also possible for several clock cycles to elapse between the assertion of BP.BUSY- and the eventual assertion of BP.VALID-.

The IOC card does not care how long it takes the memory system to complete the memory cycle because it enters a wait state until the assertion of BP.VALID-. A flag called IOC.EXTMEMCYC-, located in the IOCP1 DMA control PAL, is used to indicate that an IOC-initiated memory cycle is in progress.

```
--- PAL equation for IOC.EXTMEMCYC-
EXTMEMCYC := MRQFF*MEMGOFF*/BUSYL*IOPON ;extender's DMA cycle
           + /MEMGOFF*INTRQFF*IAKB*/ICHID*ABDBAV*IOPON
                                        ;extender's Int. Vector cycle
                                        ; (copy of MEMGO generator)
           + EXTMEMCYC*/VALIDL*IOPON    ;hold until VALID received
```

Product term 1 sets the IOC.EXTMEMCYC- with the internal version of IOC.QMEMGO- for DMA. Product term 3 keeps the flip-flop set until one cycle after IOC.VALIDL+ goes high. External to the PAL, IOC.EXTMEMCYC- is inverted with a gate [IO1007@18A] to produce IOC.EXTMEMCYC+.

IOC.EXTMEMCYC+ and IOC.WRITE- (updated with the addressing registers) are AND-ed together at an AND-OR-INVERT gate [IO1206@31E] which produces a low-true output during DMA read memory accesses. The output of the AOI gate qualifies a series of signals and forms the latch enable for the data-out latch and the instruction register. The data-out latch is updated with the data returned by the memory system at the end of the DMA cycle (rising edge of BP.VALID-). Although the instruction register gets updated, its contents are currently ignored.

IOC.EXTMEMCYC+ also runs through a NOR gate [IO1106@17A] as one of two signals which gets qualified with IOC.VALIDL- at a gate [IO0604@17C]. The result is passed through a gate [IO0705@18C] as one of two IOC.DATAAV+ sources. IOC.DATAAV+ is inverted by a buffer [IO0704@19C] to finally create the IO.DATAAV- message for the EXT board.

With the IO.DATAAV- message and the data-out latch containing the DMA read data, the IOC's involvement in the transfer is complete. As before, the EXT uses EXT.DATAAV+ to create the EXT.DOLCH+ latch signal to latch the data off the cable data-out bus. EXT.DATAAV+ also finds use in causing the assertion of EXT.MEMREGEN-.

```
--- PAL equation for EXT.MEMREGEN- and EXT.VALIDFF-
MEMREGEN := /WRITE*/DMACYC*BUSYL*DATAAV*BPONL
                                       ;copy VALIDFF for a DMA read
          + INSTRAV*BPONL              ;copy RNICYC for instruction
                                       ; broadcast
                                       ;MEMREGEN enables PE driver
                                       ; and DATA-OUT register for
                                       ; DMA read and instruction
                                       ; broadcast cycles

VALIDFF := /WRITE*/DMACYC*BUSYL*DATAAV*BPONL
                                       ;rd or int: DATAAV
                                       ; received when DMACYC
                                       ; flag is clear
         + WEB*MRQB*MEMGOB*/DMACYC*BPONL
                                       ;wr: valid immediately if
                                       ; no DMACYC in progress
         + WEB*MRQB*MEMGOB*DMACYC*ACKDMA*BPONL
                                       ;wr: valid immediately if
                                       ; DMACYC but ACKDMA received
                                       ; for first DMA write
         + WRITE*DMACYC*BUSYL*ACKDMA*BPONL
                                       ;wr: valid when ACKDMA
                                       ; received for last request
                                       ; (when DMACYC FF is set)
```

The first product term of EXT.MEMREGEN- is a copy of the first product term for EXT.VALIDFF-. When XP.VALID- is asserted on the extender backplane to signify the end of a DMA read cycle, an output enable for the data-out latch is simultaneously created. EXT.MEMREGEN- is delayed by a half clock cycle with a D-type flip-flop [EX0805@32E]. Using the negative flip-flop output, EXT.PEREN+ enables the parity error open-collector buffer [EX0907@38B], and in its inverted form [EX0105@35E] as EXT.DOREN-, also enables the data-out buffer to drive the extender backplane data bus with the contents of the data-out latch. Data (and the associated parity error information) is driven on the data bus for a half clock cycle before and after the rising edge of XP.VALID-.

The I/O extender handles all DMA reads from memory in exactly the same manner. There are always two cycles of overhead from the assertion of XP.MRQ- and XP.MEMGO- to the IOC's generation of the corresponding signals on the CPU backplane. On the data return trip, there is a single cycle delay from the assertion of BP.VALID- to the assertion of XP.VALID-. If a DMA read is executed in two cycles on the CPU backplane, then an I/O card in the extender that initiates a DMA read operation handshakes the same transfer in five cycles (two normal access plus three overhead cycles). When the memory handshake is extended on the CPU backplane for memory refreshing, the corresponding handshake on the extender backplane is extended by the same number of clock cycles. The peak DMA read-only bandwidth on the extender backplane is a fraction of the peak DMA read-only bandwidth on the CPU backplane. That fractional value is found by the formula : Ext_BW_coeff := CPU_mread_cyc / (3 + CPU_mread_cyc). For CPU_mread_cyc = 2, Ext_BW_coeff = 0.4; and for CPU_mread_cyc = 3, Ext_BW_coeff = 0.5. Keep in mind that the achievable DMA read bandwidth on the extender backplane is also limited by the amount of DMA activity from higher priority I/O cards on the CPU backplane.

DMA Reads from Memory - Parity Errors

The I/O extender is also equipped to handle parity errors detected during DMA read operations initiated by an I/O card on the extender backplane.

Parity errors are detected by the CPU's memory controller logic at the end of a memory read access. The controller generates parity information based on the data it is driving onto the backplane data bus. This information is compared with the polarity of the parity information stored into memory during a previous memory write of that memory location. If the comparison fails, the memory controller alerts the receiving card by asserting BP.PE- to indicate that the data bus contains invalid data. BP.PE- is also used by error-correcting memory controllers when multiple-bit errors are detected and cannot be corrected.

The IOC card latches the state of BP.PE- at the end of its DMA read cycle. IOC.PELCH+ [IO1106@32E] is the latch enable signal for the Parity Error Latch [IO0206@34A]. The timing of IOC.PELCH+ differs from that of IOC.DOLCH+ because parity information is valid at the start of the long half cycle following the rising edge of BP.VALID-. This is necessary because the parity comparison is performed after the backplane data bus has become valid. The latched version, IOC.PE-, is buffered by a gate [IO0602@35A] to provide the necessary drive for the cable signal.

The EXT receives IO.PE- into its Parity Error Latch [EX0305@32C] using the EXT.DOLCH+ latch enable, which is also used to load the data-out latches. The latched signal, EXT.PEL-, is inverted by a gate [EX1206@37B] to provide a high-true signal, EXT.PEL+, for the open-collector NAND gate buffer [EX0907@38B] driving XP.PE-. The parity error buffer is an open-collector device to duplicate the same type of driver used by the memory controllers on the CPU backplane. EXT.PEL+ is qualified by EXT.PEREN+ to limit the assertion of XP.PE- to the half cycles preceding and following the rising edge of XP.VALID-.

When the requesting I/O card receives the DMA read data along with a parity error (XP.PE- low), it suspends further DMA activity and makes an interrupt request to the CPU. Software is then invoked to handle this type of exception.

DMA Writes Into Memory - Simple Case

As with the DMA read case, a DMA write transfer begins with the assertions of XP.MRQ- and XP.MEMGO-. During XP.MEMGO- low, an address, as well as the corresponding data, is passed from the I/O interface card to the EXT. Due to differences in the backplane timing between the addressing bus (which includes the 15-bit address bus, the 5-bit address extension/mapping bus, the self-configure bit, and the write-enable bit) and the data bus, the two busses are sampled at different times by the EXT. The address bus is valid before the start of the short half cycle of the clock which causes XP.BUSY- to go low. Therefore the addressing information is picked up when the EXT.BUSYFF+ flip-flop sets. Since the data bus has twice as much capacitive loading as the address bus, its contents are not valid until 10 nsec after the start of the short half cycle. This number applies to the BP.SCLK- frequency used by the A600+. If the clock frequency is increased, the validity of the data bus is pushed further into the short half cycle. For all clock frequencies, the data bus remains valid through the end of the short half cycle, which has a minimum pulse width of 90 nsec under normal operating conditions (A600+). The EXT card employs a delay line to sample the data bus about 65 nsec after the start of the short half cycle of XP.SCLK-.

The data-in register clock begins with the generation of a half cycle pulse by logically ANDing EXT.BUSYL- with EXT.BUSYFF+. EXT.BUSYL- is the inversion of EXT.BUSYL+ after processing through a NAND gate [EX0905@31E]. EXT.BUSYL- is logically ANDed with EXT.BUSYFF+, and the result inverted using another NAND gate [EX0905@31E] to produce EXT.MEMCLK-. A delay line [EX1203@32E] with a 50 nsec delay time-shifts EXT.MEMCLK-, by approximately half of a short half cycle, to generate EXT.MEMCLK50-. Since there are two conditions for reading data off of the extender backplane, a gate [EX0404@34D] combines EXT.MEMCLK50- and EXT.IODBCLK- (for I/O handshaking) to create the universal EXT.DICLK+ clock pulse for the data-in registers. Since these are clock edge-triggered devices, the leading (rising) edge of EXT.DICLK+ loads the data-in registers with the data bus information. (See Figure 12-7.)

The operation of the addressing bus registers, the synchronizing registers, and their associated buffers has been discussed in the previous sections. For DMA write operations, these components behave in exactly the same fashion.

Because DMA read operations require the requesting I/O interface card to wait for returning data, the extender backplane is placed into a busy state while the I/O extender is obtaining that information. DMA write operations can be considered complete once the addressing information and the data have passed from the I/O interface card to the EXT. It is not necessary to freeze the extender backplane until the data has actually been written into CPU memory. For the sake of simplicity, the discussion of DMA writes in this section assumes a completely quiescent extender and CPU backplanes. That means the DMA write request initiated on the extender backplane propagates through the EXT and IOC cards, then onto the CPU backplane in the minimum number of clock cycles. Once the basic concept of DMA writes using the extender hardware has been established, the more complex case of using the extender two-stage pipeline is discussed.

As with any DMA handshake, the EXT.BUSYFF+ flip-flop [EX1006@11B] is set with EXT.MEMGOB+. EXT.MEMGOB+ is also received by the EXTP1 DMA Control PAL [EX1004@15B] where EXT.VALIDFF- is also asserted.

```
--- PAL equation for EXT.VALIDFF-
VALIDFF := /WRITE*/DMACYC*BUSYL*DATAAV*BPONL
                                          ;rd or int: DATAAV
                                          ; received when DMACYC
                                          ; flag is clear
         + WEB*MRQB*MEMGOB*/DMACYC*BPONL
                                          ;wr: valid immediately if
                                          ; no DMACYC in progress
         + WEB*MRQB*MEMGOB*DMACYC*ACKDMA*BPONL
                                          ;wr: valid immediately if
                                          ; DMACYC but ACKDMA received
                                          ; for first DMA write
         + WRITE*DMACYC*BUSYL*ACKDMA*BPONL
                                          ;wr: valid when ACKDMA
                                          ; received for last request
                                          ; (when DMACYC FF is set)
```

Product term 2 provides for the immediate assertion of EXT.VALIDFF- if EXT.DMACYC- is not set, indicating that there are no previous DMA transfers still in the extender pipeline. As a reminder, EXT.DMACYC- is set with EXT.MEMGOB+, but is cleared after receiving the EXT.ACKDMA+ message from the IOC card, confirming that the DMA write data has been written into CPU memory. With the assertion of EXT.VALIDFF-, EXT.BUSYFF+ clears after being set for one cycle. EXT.VALIDFF- is self-extinguishing because there are no product terms leading to its extension. The DMA write handshake completes on the extender backplane before the IOC has even begun to use the data sent to it from the EXT.

As in the DMA read case, EX.ABDBAV- is used by the EXT to tell the IOC that valid addressing information and data are on the cable bus lines. The IOC converts EX.ABDBAV- into IOC.ABDBCLK+ to load the addressing registers and the data-in register with the information necessary to perform the DMA write operation. This time IOC.WRITE- [IO0106@24A] is low, which causes the assertion of IOC.DIREN- [IO1206@32E] to enable the data-in buffers onto the CPU backplane data bus during BP.MEMGO-. Since the IOCP1 DMA Control PAL [IO0905@15A] generates IOC.MEMGOFF- with the receipt of EX.ABDBAV-, the DMA transfer over the CPU backplane is started as soon as the data arrives from the EXT.

If EX.ABDBAV- is received while the backplane is busy, or if the IOC is disabled by a higher priority DMA channel, IOC.MEMGOPD- is used to place the current request in the wait state. Once the IOC is able to obtain use of the CPU backplane, it generates BP.MEMGO- and writes the given data into the specified address in CPU memory. The DMA transfer is complete even if the resulting handshake takes several cycles to finish. No data is returned at the end of the handshake, so the IOC card prevents the data-out latches from loading nonexistent information from the CPU backplane data bus. IOC.WRITE- prevents IOC.MEMCYC- [IO1206@31E] from going low during a DMA write cycle, preventing IOC.DOLCH+ from pulsing high at the end of the handshake.

DMA Writes Into Memory - Queued Case

Since the I/O extender is based on a two-stage pipelined design, this permits several operations to be in progress at the same time. All that is needed to implement this feature is a method for the IOC to tell the EXT that it is permitted to advance the pipeline, and for the EXT to inform the IOC that new data is being fed into the pipeline. These signals already exist, in the form of IO.ACKDMA- and EX.ABDBAV-. For the DMA transfers previously described, IO.ACKDMA- is not significant while EX.ABDBAV- causes new address and data to be loaded into the IOC registers and BP.MEMGO- to be generated.

The DMA write queue is employed for two reasons. In the first case, it allows full bandwidth DMA write operation on the extender backplane. Assuming that the CPU backplane is allowing the IOC to achieve full DMA bandwidth, the extender uses the pipeline to keep a continuous flow of data from the extender backplane to the CPU backplane. The key to this efficiency lies in the handling of DMA writes. Once data is passed from the I/O interface card to the EXT, the extender backplane is available for another DMA transfer, with the assumption that the EXT will get the data into memory somehow. The EXT also turns the data over to the IOC, again assuming that the receiver will take care of the actual memory write. Finally, the IOC passes the DMA write information to the memory controller, where the actual memory write occurs.

The second use for the DMA write queue is that of a very small 2-word buffer. Because the IOC registers can hold the information for one DMA write operation, and the EXT registers for a second DMA write transfer, the DMA queue is also used to absorb some of the transfer rate irregularities caused by frequent use of the CPU backplane by higher priority DMA channels. The extender may receive several slower transfers (1 Mbyte per second, for example) into the queue, only to write them into CPU memory at full memory bandwidth (4.27 Mbyte per second for A600+) when finally given access to the CPU backplane.

Both applications of the DMA write queue are handled in exactly the same way. In the first case, the DMA write queue is always full, while in the second case, the DMA write queue becomes full when the pipeline cannot advance because of some blockage on the CPU backplane. As a reminder, the DMA write queue only applies to consecutive DMA write operations. An imbedded DMA read operation between two series of DMA write transfers forces the DMA write queue to be flushed when the read operation is performed. When the second series of DMA write transfers begin, the queue is reactivated, if necessary.

Two flags are maintained on each of the extender logic boards. The IOC flags are IOC.MEMGOPD- and IOC.MEMGOQUEUE-. The corresponding flags on the EXT are EXT.DMACYC- and EXT.DMAQUEUE-. Almost identical flags are kept on each board to limit the number of cable signal lines. The first flag on each board (IOC.MEMGOPD- and EXT.DMACYC-) indicates that the IOC registers contain valid information for a DMA write transfer which has not yet made it out to the CPU backplane. This also means that the pipeline cannot be advanced so as to overwrite these registers; doing so would destroy the information for one DMA write transfer. The second flag (IOC.MEMGOQUEUE- and EXT.DMAQUEUE-) indicates that the EXT registers contain valid DMA write information which cannot be advanced to the IOC registers. The second flag is set only after the first flag has been previously set. When the second flag is in effect, the extender backplane is placed into a BUSY state, but DMA channels are still permitted to arbitrate the next (third) DMA transfer without starting it.

To illustrate the operation of the DMA write queue, assume that the IOC card is completely denied access to the CPU backplane by higher priority DMA activity. Also, assume a high-speed device on the extender backplane is acquiring data for storage in main memory. When the first DMA write transfer is initiated, the EXT sets the EXT.DMACYC- flag to track the progress of the first transfer (product term 1).

```
--- PAL equation for EXT.DMACYC-
DMACYC := MRQB*MEMGOB*BPONL              ;starts DMA cycle
        + DMACYC*/ACKDMA*BPONL           ;stay DMA mode until ACKDMA
        + DMAQUEUE*BPONL                 ;DMA queue is two deep
```

Because the IOC cannot launch the first DMA transfer, IO.ACKDMA- is not sent back to the EXT. Thus, EXT.DMACYC- remains set.

With information in its holding registers, the IOC sets the IOC.MEMGOPD- flag to remind itself that it is waiting for a chance to use the backplane.

```
--- PAL equation for IOC.MEMGOPD-
MEMGOPD := MRQPD*ABDBAV*IOPON            ;if MRQFF not yet set
         + MRQFF*MCHID*ABDBAV*IOPON      ; or not yet EXT priority
         + MRQFF*BUSYL*/VALIDL*ABDBAV*IOPON
                                         ; memory cycle in progress
         + MRQFF*MEMGOFF*BUSYL*IOPON     ;A600+ CPU steals cycle
         + MEMGOPD*/MRQFF*IOPON          ;hold until MRQFF set
         + MEMGOPD*MRQFF*/MEMGOFF*IOPON ; and until MEMGO issued
         + MEMGOPD*MRQFF*MEMGOFF*/BUSYL*ABDBAV*IOPON
                                         ;queue up second DMA write
                                         ; as soon as first one has
                                         ; gone to memory
         + MEMGOQUEUE*IOPON              ;queue up second DMA write
```

BP.MCHID- prevents IOC.MEMGOFF- from setting, so the second product term of IOC.MEMGOPD- will cause the pending flag to be activated.

With the first DMA transfer blocked at the IOC, and the extender backplane available for another DMA memory cycle, assume that a second DMA write transfer is initiated. Because the EXT.DMACYC- flag is set, and if IO.ACKDMA- is not received by the EXT when the information for the second transfer is stored into the EXT registers, the EXT.DMAQUEUE- flag is set to indicate a full queue status.

```
--- PAL equation for EXT.DMAQUEUE-
DMAQUEUE := MRQB*MEMGOB*DMACYC*/ACKDMA*BPONL
                                          ;queue up 2nd DMA req
          + DMAQUEUE*/ACKDMA*BPONL        ;stay set until ACKDMA clears
```

Product term 1 sets the EXT.DMAQUEUE- flag when EXT.DMACYC- is set if another transfer is started (EXT.MRQB+ and EXT.MEMGOB+ are both true) when the DMA write pipeline is unable to advance (EXT.ACKDMA+) beyond the IOC and onto the CPU backplane.

Since the IOC and EXT registers both contain the information for two DMA write transfers in limbo, the extender must prevent another DMA transfer from starting. This is accomplished by extending the duration of the second transfer using the XP.BUSY- signal. Unlike the empty queue case, EXT.VALIDFF- does not automatically respond to DMA MEMGOs.

```
--- PAL equation for EXT.VALIDFF-
VALIDFF := /WRITE*/DMACYC*BUSYL*DATAAV*BPONL
                                          ;rd or int: DATAAV
                                          ; received when DMACYC
                                          ; flag is clear
         + WEB*MRQB*MEMGOB*/DMACYC*BPONL
                                          ;wr: valid immediately if
                                          ; no DMACYC in progress
         + WEB*MRQB*MEMGOB*DMACYC*ACKDMA*BPONL
                                          ;wr: valid immediately if
                                          ; DMACYC but ACKDMA received
                                          ; for first DMA write
         + WRITE*DMACYC*BUSYL*ACKDMA*BPONL
                                          ;wr: valid when ACKDMA
                                          ; received for last request
                                          ; (when DMACYC FF is set)
```

The assertion of EXT.DMACYC- rules out product terms 1 and 2. In product term 3, EXT.VALIDFF- is set immediately after the DMA XP.MEMGO- (as in the empty queue case, product term 2) only if the IOC is simultaneously disposing of the information in its holding registers by successfully launching a DMA write cycle on the CPU backplane (EXT.ACKDMA+ true). Otherwise, a lengthened second transfer is terminated only after EXT.ACKDMA+ is received (product term 4). At that time EXT.VALIDFF- is asserted, and one cycle later XP.BUSY- is deasserted. (See Figure 12-8.)

When the second transfer is picked up by the EXT, another EX.ABDBAV- message is sent to the IOC. IOC.MEMGOPD- is already set, so another status flag called IOC.MEMGOQUEUE- is utilized to indicate that the DMA write queue is now two deep.

```
--- PAL equation for IOC.MEMGOQUEUE-
MEMGOQUEUE := /MEMGOQUEUE*MEMGOPD*/MEMGOFF*ABDBAV*IOPON
                                         ;A600+ CPU preemptive access
            +  MEMGOQUEUE*/MRQFF*IOPON ;hold until MRQFF set
            +  MEMGOQUEUE*MRQFF*/MEMGOFF*IOPON
                                         ;hold until MEMGO issued
            +  MEMGOQUEUE*MRQFF* MEMGOFF*BUSYL*IOPON
                                         ;hold for A900 early VALID
                                         ; case (MEMGO really didn't
                                         ; make it to the backplane)
            + /MEMGOQUEUE*MEMGOPD* MEMGOFF*BUSYL*ABDBAV*IOPON
                                         ;A900 early VALID recovery,
                                         ; queue up 2nd DMA write
                                         ; because current DMA MEMGO
                                         ; was fooled into assertion
```

The first product term provides the setting condition, while product terms 2 and 3 keep IOC.MEMGOQUEUE- true until the IOC is allowed to attempt the assertion of IOC.MEMGOFF-. If IOC.MEMGOFF- does not lead to the successful assertion of BP.MEMGO-, product term 4 causes the IOC.MEMGOFF- attempt to be ignored, and subsequently keep IOC.MEMGOQUEUE- set. In product term 5, IOC.MEMGOQUEUE- becomes set when IOC.ABDBAV+ is received while the IOC is making an unsuccessful IOC.MEMGOFF- attempt. The decision to make an IOC.MEMGOFF- attempt is made by predictive logic which permits IOC.MEMGOFF- to go low during the next clock cycle if IOC.BUSYL+ and IOC.VALIDL+ are both true in the current cycle. Normally, the backplane would be available for use on the next cycle, but this is not true of A900 DMA memory writes with concurrent memory refreshing. The A900 asserts BP.VALID- during the first cycle of BP.BUSY-, then keeps BP.BUSY- low for the next two or three cycles.

In addition to the IOC.MEMGOFF- generating function, EX.ABDBAV- also causes new information to be loaded off of the cable addressing and data busses. However, doing so when IOC.MEMGOPD- is true would permanently overwrite the yet unused information on the IOC registers. Therefore, the operation of IOC.ABDBCLK- must be modified to accommodate the DMA write queue feature.

```
--- PAL equation for IOC.ABDBCLK+
/ABDBCLK := /IOPON                          ;ABDBCLK+=0 at power-up
          + /MEMGOPD*/ABDBAV                ;if MEMGOPD not set, stay
                                            ; at 0 until ABDBAV true
          +  MEMGOPD*MRQFF*/MEMGOFF         ;if MEMGOQUEUE and MEMGOPD
          +  MEMGOPD*MRQFF* BUSYL           ; are set, pulse ABDBCLK hi
          + /MEMGOQUEUE*MEMGOPD*MRQFF*/ABDBAV
                                            ; immediately after current
                                            ; good MEMGO
                                            ;if only MEMGOPD set, pulse
                                            ; ABDBCLK after current good
                                            ; MEMGO only if new data is
                                            ; on the cable (ABDBAV)
                                            ;if current MEMGO is no
                                            ; good (A900 extended BUSY)
                                            ; then MEMGOQUEUE is set,
                                            ; forcing data pickup
                                            ; off the cable at the end
                                            ; of a good MEMGO.
```

Assume that the DMA write queue is only one deep (IOC.MEMGOQUEUE- is not set), IOC.MEMGOFF- is issued when IOC.BUSYL+ is false, and IOC.ABDBAV+ is received. This describes a situation for a (first) DMA write transfer which has been retained on the IOC (IOC.MEMGOPD- set), but is allowed to advance onto the CPU backplane at the same time that information for the second DMA write transfer becomes available on the cable (IOC.ABDBAV+ true). The entire pipeline is advanced. Since the IOC has written the first transfer into memory, IOC.ABDBCLK+ is generated immediately after IOC.ABDBAV+ is received. With IOC.IOPON+ and IOC.MEMGOPD- both true, product terms 1 and 2 are automatically false. Product terms 3, 4, and 5 become false for one cycle to generate IOC.ABDBCLK+ for this particular case.

The next obvious case involves IOC.ABDBAV+ received when the IOC cannot generate IOC.MEMGOFF-. Even with the information for a new DMA write transfer available on the cable, the IOC cannot generate IOC.ABDBCLK+. The EXT board goes into a wait state and keeps the information for the second DMA write transfer on the cable. If the IOC cannot generate IOC.MEMGOFF-, product term 3 senses this, and keeps IOC.ABDBCLK+ low. At this time, the pipeline becomes full and IOC.MEMGOQUEUE- is set to indicate that status.

When IOC.MEMGOQUEUE- is set, product term 5 is automatically false, leaving only product terms 3 and 4 to prevent the assertion of IOC.ABDBCLK+. As soon as the IOC can generate a good IOC.MEMGOFF- (when IOC.BUSYL+ is not simultaneously high, leading to the assertion of BP.MEMGO-), product terms 3 and 4 become false. Thus, IOC.ABDBCLK+ automatically pulses high for one cycle following a successful DMA write into memory when the pipeline is full. This automatic IOC.ABDBCLK+ is possible because the information for the next DMA write transfer is readily available on the cable.

The launching of the first queued-up DMA write transfer advances the pipeline, so IOC.MEMGOQUEUE- becomes false after the assertion of BP.MEMGO-. EXT.DMAQUEUE- also goes false after receiving IO.ACKDMA-, an indication of a successful BP.MEMGO-. By now, the second transfer in the queue has advanced to the IOC registers, so the frozen handshake on the extender backplane is allowed to complete. IO.ACKDMA- is also used to start off EXT.VALIDFF-, which leads to the clearing of EXT.BUSYFF+.

```
--- PAL equation for EXT.VALIDFF-
VALIDFF := /WRITE*/DMACYC*BUSYL*DATAAV*BPONL
                                         ;rd or int: DATAAV
                                         ; received when DMACYC
                                         ; flag is clear
         + WEB*MRQB*MEMGOB*/DMACYC*BPONL
                                         ;wr: valid immediately if
                                         ; no DMACYC in progress
         + WEB*MRQB*MEMGOB*DMACYC*ACKDMA*BPONL
                                         ;wr: valid immediately if
                                         ; DMACYC but ACKDMA received
                                         ; for first DMA write
         + WRITE*DMACYC*BUSYL*ACKDMA*BPONL
                                         ;wr: valid when ACKDMA
                                         ; received for last request
                                         ; (when DMACYC FF is set)
```

Product term 4 is responsible for generating EXT.VALIDFF- if EXT.ACKDMA+ is received when EXT.DMACYC- is set.

The queue status flags are cleared after BP.MEMGO- is generated. However, IOC.MEMGOPD- is kept in action by the previous assertion of IOC.MEMGOQUEUE-. Likewise, EXT.DMACYC- remains asserted because of EXT.DMAQUEUE-. This technique provides the linkage necessary to back out of the DMA write queue one step at a time, and at the same time provide for the possibility that the queue may become full again after becoming one deep.

As the second handshake is finishing up on the extender backplane, the IOC is ready to generate the second BP.MEMGO- as soon as the memory controller indicates that the first transfer is complete. Assuming that the second transfer is launched immediately, IOC.MEMGOPD- becomes false, and the entire DMA write queue is momentarily empty. Meanwhile on the extender backplane, the third consecutive DMA write transfer begins. This transfer arrives when EXT.DMACYC- is to be cleared by the incoming IO.ACKDMA-. Since IO.ACKDMA- clears EXT.DMACYC-, but XP.MEMGO- sets EXT.DMACYC- (if EXT.DMACYC- is currently false), the two actions nullify each other and EXT.DMACYC- remains set to indicate that a single DMA transfer is in the pipeline. The resulting handshake on the extender backplane is also brief and immediate, as if the pipeline were empty.

Assuming no additional backplane delays due to memory refreshing and DMA priority conflicts, the third transfer arrives at the IOC just as the second handshake is completing. Since IOC.MEMGOPD- and IOC.MEMGOQUEUE- are both false, the incoming information is immediately loaded into the IOC registers and BP.MEMGO- is generated. The accompanying IO.ACKDMA- goes back to the EXT and clears EXT.DMACYC-. The pipeline becomes empty again.

The preceding discussion dealt with a typical and straightforward application of the DMA write queue. In a real world application in which the IOC card is typically very low in the DMA priority chain on the CPU backplane, the IOC may be prevented from doing DMA for long periods of time. The DMA write queue may actually improve performance for the I/O cards in the extender because it can acquire data over a period of time when the CPU backplane is busy. When the IOC is eventually allowed access to the CPU backplane, it can unload the pipeline as fast as the memory controller can accept the information (full DMA bandwidth).

The maximum DMA efficiency for the I/O extender is achieved when the pipeline is kept full. If the I/O extender's IOC card is the highest priority DMA channel on the CPU backplane and if the I/O cards in the extender are performing enough DMA writes to achieve full DMA bandwidth (as measured on the CPU backplane), none of the other I/O cards on the CPU backplane are allowed to do DMA. On the other hand, the CPU is at its lowest efficiency, since all available memory bandwidth is servicing the I/O extender.

DMA Self-Configuration

Aside from DMA reads and DMA writes, there is another special category of DMA transfers. A DMA self-configuration is a procedure in which an interface card uses DMA techniques to obtain the DMA configuration words which describe the attributes of the following data transfer. The most obvious technique for loading the I/O interface DMA configuration registers uses the OTA/B I/O instructions. However, this method is slow because it requires the CPU to read the configuration words from memory, and then pass the information to the I/O card one word at a time. In a DMA self-configuration, the I/O card is given the starting address for a block of words in memory which contain the DMA configuration. The block may contain three or four words, and is known as a triplet or quad, respectively.

To the I/O extender, the DMA self-configuration operation appears as a series of DMA read requests. The self-configuration operation is rather slow, requiring over 17 clock cycles to obtain a quad if the I/O card was on the CPU backplane. For the same card on the extender backplane, this time is extended because of the increased number of cycles required to perform the DMA reads. The fact that the entire operation requires over 5 usec to execute is not a problem for the extender control logic. The I/O card performing the self-configuration denies priority to lower priority DMA channels even if it is using three cycles between each transfer. To do this, the self-configuring I/O card keeps XP.MRQ- low during the entire operation.

Even this is transparent to the I/O extender. However, there is one subtle side effect of the continuous XP.MRQ- which does affect the IOC. For normal DMA read operations, a parity error encountered during a memory read causes the affected I/O card to terminate DMA and promptly interrupt the CPU. A parity error during a DMA self-configuration is slightly different because it causes the current, ongoing request (XP.MRQ-) to abort. For regular DMA, XP.MRQ- terminates with XP.MEMGO-, and a parity error at the end of one transfer prevents the next request. But with a self-configuration, the request has been made, and then aborted after a parity error.

The IOC card handling the self-configuration must cope with this situation. It understands that activity started by IOC.REQDMA+ is normally terminated with IOC.ABDBAV+. For the normal case, IOC.MRQFF- remains set until a good IOC.MEMGOFF- is issued in the absence of further IOC.REQDMA+, IOC.ABDBAV+, and IOC.MEMGOPD-. The aborted self-configuration DMA appears to the IOC as the deassertion of IOC.REQDMA+ when IOC.ABDBAV+ has not been received and IOC.MEMGOPD- is not set. This indicates that no data has been sent to the IOC before the DMA request is withdrawn.

```
--- PAL equation for IOC.MRQFF-
MRQFF := REQDMA*/IAKB*/IOGOS*IOPON     ;REQDMA : start now or use
       + MRQPD*/IAKB*/IOGOS*IOPON      ; MRQPD when OK to start
       + MRQFF*/MEMGOFF*/MEMGOPD*REQDMA*IOPON
                                       ;remain asserted until
                                       ; MEMGOFF or MEMGOPD set
                                       ; (normal case), or
                                       ; until REQDMA goes away
                                       ; (aborted DMA case [PE])
       + MRQFF*/MEMGOFF*MEMGOPD*IOPON  ;extend MRQFF when MEMGOPD
                                       ; & set until MEMGO issued
       + MRQFF*MEMGOFF*MEMGOPD*MEMGOQUEUE*IOPON
                                       ;extend MRQFF beyond MEMGO
                                       ; if MEMGOQUEUE is set
       + MRQFF*MEMGOFF*BUSYL*IOPON     ;A600+ CPU preemptive memory
                                       ; access during first cycle
                                       ; of MRQ- (recovery mode)
                                       ;A900 CPU early VALID
                                       ; handshake recovery
                                       ; (recovery mode)
```

Since IOC.REQDMA+ is at least one cycle long, IOC.MRQFF- is set directly (product term 1) or indirectly through IOC.MRQPD- (product term 2). Product term 3 provides the condition for the deassertion of IOC.MRQFF- when IOC.REQDMA+ is removed before any DMA transfer information has been sent to the IOC from the EXT.

The DMA-Interrupt Conflict

The conflict between a DMA operation via the I/O extender and an I/O instruction broadcast by the CPU is an example of interlock resolution. The I/O extender has to guarantee that any DMA already in the extender's DMA pipeline can complete before the I/O instruction is broadcast. By the same token, it terminates further DMA activity on the extender backplane to allow the instruction broadcast to complete. This arbitration period requires two cycles (due to the use of a two-stage pipeline) beginning with the earliest hint of a potential collision between DMA and the instruction broadcast.

A similar conflict exists between DMA activity and interrupt processing. Unlike the instruction broadcasting operation, the CPU does not give advance warning that it is about to acknowledge an I/O interrupt request. Once an interrupt acknowledge is generated, the CPU assumes that the very next memory cycle is a trap cell fetch performed by the highest priority requesting interface card. Therefore, if a DMA transfer is propagating through the extender's DMA pipeline when the interrupt acknowledgement is made, the IOC must place the DMA transfer on hold. However, the I/O extender cannot pass the acknowledgement to the extender backplane because the subsequent trap cell fetch would cause the IOC addressing and data-in registers to be overwritten.

Prior to the actual interrupt acknowledgement, the only other warning that the I/O extender may use is the interrupt request itself. If any I/O card(s) on the extender backplane generates an interrupt request for any reason, the EXT asserts XP.CPUTURN- to prevent the generation of any new DMA requests. Since the DMA shutdown occurs prior to the IOC making the corresponding interrupt request on the CPU backplane, the I/O extender can always guarantee that there is no DMA activity prior to the interrupt acknowledgement and the subsequent trap cell fetch. DMA is allowed to resume immediately after the trap cell fetch unless another interrupt request is pending on the extender backplane.

Processing Interrupts

Interrupt processing on the extender backplane begins with an I/O interface card making an interrupt request. The I/O extender passes the request to the CPU backplane, and after a period of time, the CPU makes an acknowledgement. This action places the CPU backplane into an interrupt acknowledgement mode so that the next memory cycle is interpreted as a vectored interrupt trap cell fetch. The interrupting interface initiates the access to the memory location corresponding to its select code (hence, an auto addressed interrupt jump, or vectored interrupt). This memory address is passed to the CPU backplane by the I/O extender. The CPU executes the instruction fetched by the interrupting interface card (the trap cell fetch), which is usually a jump instruction to the start of the routine that handles I/O interrupts. (See Figure 12-9 for signal timing.)

The Interrupt Priority Chains

Interrupt priority is determined by a serial enable/disable chain similar in operation to that used for resolving DMA priority. An I/O interface card which makes an interrupt request disables all lower priority interface cards from responding to the next interrupt acknowledgement. Every I/O card receives the BP.ICHID- interrupt chain disable signal sent to it from the next higher priority interface card. Each I/O card generates the BP.ICHOD- interrupt chain disable that is sent to the next lower priority interface card. Therefore, the BP.ICHOD- output of the higher priority card is connected to the BP.ICHID- input of the next lower priority card.

The highest priority interface card on the CPU backplane is always enabled to respond to an interrupt acknowledgement if it is making an interrupt request. When that card makes an interrupt request, it also asserts BP.ICHOD- to disable all lower priority cards. The next lower priority card receives the disable and passes the disabling condition to the next lower priority card. This sequence is repeated for every I/O card installed on the CPU backplane. The IOC card handles this situation in exactly the same manner using a 3-input AND gate [IO1205@19B]. Whenever BP.ICHID- is received low (disabling the IOC from responding to an interrupt acknowledgement), BP.ICHOD- is likewise low when leaving the IOC for the next lower priority I/O card. If BP.ICHID- is received high, the IOC determines the level of BP.ICHOD-. There are two conditions which cause the IOC to disable lower priority I/O cards. For the first case, if the IOC is making an interrupt request (IOC.INTRQFF- low), BP.ICHOD- is brought low. The second case is a special application of the interrupt chain when the entire I/O system is placed into a special mode known as Diagnose Mode 1 or 2. Both topics are discussed in the following sections.

Like the DMA priority chain, a completely linear interrupt priority chain is not feasible without extending the backplane clock cycle time. The interrupt priority chain is partitioned, so that interrupt priority is resolved on the extender backplane disjointly yet simultaneously with the priority resolution on the CPU backplane. It does not matter how many I/O cards are making an interrupt request in the entire system, only one (the highest priority) interrupt request can be acknowledged at a time. Any interrupt request(s) on the extender backplane invokes the generation of an interrupt request on the CPU backplane by the IOC. If the IOC card is disabled from responding to the current interrupt acknowledgement, then no acknowledgement is made on the extender backplane. Eventually, the IOC card receives an interrupt acknowledgement when it has interrupt priority. When that occurs, an interrupt acknowledgement is performed on the extender backplane, with the highest priority interrupting interface performing the vectored trap cell fetch. Therefore, interrupt priority is a two step process for interrupts generated by I/O cards on the extender backplane.

Requesting an Interrupt

All interrupt requests are initiated by I/O interface cards on the extender backplane using the open-collector XP.INTRQ- signal. The EXT provides a resistor termination [EX1007@10A] for this signal to duplicate the same environment normally seen on the CPU backplane. XP.INTRQ- is received into an inverting buffer [EX1107@11A] to isolate the board version of the signal from the noisier backplane. EXT.INTRQ+ is sampled at the start of the long half cycle with a D-type flip-flop [EX1106@12A] to decouple the backplane timing requirements from that of the EXT. EXT.INTRQL+, the synchronized version, is used by the EXTP3 Interrupt and Slave Control PAL [EX1105@15D] to create EXT.INTRQLL+.

```
--- PAL equation for EXT.INTRQLL+
/INTRQLL := /BPONL                   ;init INTRQLL+=0
          + /INTRQL                  ;delayed version of INTRQL
```

EXT.INTRQLL+ is simply a delayed version of EXT.INTRQL+. Since XP.INTRQ- is considered an asynchronous signal, it needs to be doubly sampled before it is usable to a synchronous state machine that is running off the same clock used to sample the signal. Note that the EXT.INTRQLL+ Boolean equation requires the application of De Morgan's theorem to work within the constraints of a normally low-true device.

The EXT cable message, EX.INTFLG-, is derived by running EXT.INTRQLL+ into an inverting buffer [EX0604@19D] which provides cable line driving capability and signal isolation. The EXT card does very little in processing the original XP.INTRQ- signal, providing only the signal synchronizing service for the IOC card.

EX.INTFLG- is received by a resistor network and an inverting buffer [I00904@12D] on the IOC board. The inverted signal, IOC.INTFLG+, goes to two different PALs, IOCP1 and IOCP4. At IOCP1, IOC.INTFLG+ generates IOC.INTRQFF-, and at IOCP4, the signal is used to disqualify the serial chaining of the interrupt chain when the I/O system is not in Diagnose Mode 1 or 2.

The IOCP1 PAL's IOC.INTRQFF- output is the mechanism through which the IOC makes an interrupt request to the CPU on behalf of the I/O cards on the extender backplane.

```
--- PAL equation for IOC.INTRQFF-
INTRQFF := INTFLG*/IAKB*/REQDMA*/MRQFF*/MRQPD*IOPON
                                      ;INTFLG sets INTRQFF unless
                                      ; concurrent DMA cycle is
                                      ; coming, pending, or
                                      ; in progress
         + INTRQFF*INTFLG*ICHID*IOPON ;hold if chain disabled
                                      ;if INTFLG goes away before
                                      ; IAK true, let it go away
         + INTRQFF*/ICHID*IAKB*/ABDBAV*IOPON
                                      ;hold until ABDBAV, which
                                      ; generates MEMGO
```

Computer Museum

Product term 1 allows IOC.INTRQFF- to set when IOC.INTFLG+ is received unless another interrupt acknowledgement is already in progress or if the IOC is trying to finish up a concurrent DMA cycle. Keep in mind that IOC.INTFLG+ is not a single cycle pulse because the IOC must be able to detect the aborted interrupt request case (similar to the aborted DMA request case). Product term 2 keeps IOC.INTRQFF- asserted when the IOC does not have interrupt priority. Product term 1 is used if the IOC has interrupt priority but is waiting for the CPU's interrupt acknowledge. Finally, product term 3 takes care of the situation in which the IOC has priority, the CPU has acknowledged the interrupt, and IOC.ABDBAV+ has not yet been received. This product term keeps IOC.INTRQFF- asserted until the trap cell fetch occurs.

The outboard logic for IOC.INTRQFF- is quite simple. IOC.INTRQFF- is used by the 74F11 [I01205@19B] to generate the IOC.ICHOD- interrupt chain disable signal for the next lower priority interface card. When IOC.INTRQFF- is asserted, all lower priority cards are disabled from responding to the next interrupt acknowledgement; the IOC itself may also be disabled, but it is still permitted to reserve a future acknowledgement. The interrupt request, BP.INTRQ-, is generated by an open-collector NAND buffer [I00907@19B] whose inputs are IOC.INTRQFF+ qualified by BP.ICHID-. Although this qualification serves no practical function (since BP.INTRQ- is open-collector and the response to an interrupt acknowledgement is based on the interrupt priority chain), it is a duplicate of the I/O Master circuitry found on all I/O interface cards. The third, and last, use for IOC.INTRQFF- is a qualifier for the serial chaining function of Diagnose Modes 1 and 2 located in the IOCP4 PAL. The Diagnose Modes 1 and 2 features are described in a later section.

Acknowledging an Interrupt

Assume that several higher priority interface cards are requesting an interrupt. The CPU acknowledges those interrupt requests by asserting BP.IAK- and then waits for the trap cell fetch response. During this time, BP.ICHID- (as received by the IOC) is low. This keeps the IOC from generating BP.INTRQ- (immaterial, since the higher priority cards must be asserting this signal by virtue of BP.ICHID- low), and from responding to BP.IAK-.

BP.IAK- comes onto the IOC through an inverting buffer [IO0807@10A] to create IOC.IAKB+. The backplane protocols specify that BP.IAK- changes state at the beginning of the long half cycle, that BP.MRQ- can preempt BP.IAK- during the long half cycle, and that the state of BP.IAK- can be sampled at the start of the next short half cycle or at the start of the following long half cycle. With that intent, a D-type flip-flop samples IOC.IAKB+ at the start of the short half cycle to create IOC.IAKS+. However, the A700 CPU implements BP.IAK- differently. It asserts BP.IAK- at the start of the short half cycle to avoid the long half cycle arbitration period. Since the IOC's DMA state machine may be active when the CPU is acknowledging another I/O card's interrupt request (assume that IOC is not requesting an interrupt), the DMA state machine must know not to use the backplane in the event that BP.IAK- was asserted prior to the start of the long half cycle. Since IOC.IAKS- is valid half a cycle after the DMA state machine starts up the next transfer, IOC.IAKS- arrives too late to be used as a qualifier. Therefore, only IOC.IAKB+ changes state early enough to shut down DMA to take care of the A700 protocol difference. IOC.IAKB+ is employed by the IOCP1 DMA control PAL to place the new DMA transfer into the DMA queue and to wait until the end of the interrupt acknowledgement cycle.

Once the higher priority interrupt requests have been serviced, BP.ICHID- goes high, as seen by the IOC board. This indicates that the next BP.IAK- received should cause the corresponding assertion of XP.IAK- on the extender backplane. In the event that there are several interrupt requests queued up in the extender, it is possible for the IOC to keep sending acknowledgements to the EXT until all of the interrupts have been serviced. It is also possible for a higher priority interrupting interface to preempt the extender as it is sequencing through its own queue of interrupt requests.

The interrupt acknowledgement handler on the IOC resides in the IOCP4 Miscellaneous Functions PAL [IO1005@15D]. Two outputs are generated when IOC.IAKS+ is received.

```
--- PAL equations for IOC.IAKSL+ and IOC.ACKINT+
/IAKSL := /BPONL                          ;initialize IAKSL+=0
       + /IAKS                            ;delayed version of IAKS

/ACKINT := /BPONL                         ;initialize ACKINT+=0
         + /ACKINT * /IAKS                ;one cycle pulse to inform EXT
         + /ACKINT *  IAKSL               ; to process interrupt request
         + /ACKINT *  ICHID               ;if EXT did not request interrupt
         + /ACKINT * /INTRQFF             ; it does not get an ACKINT
         +  ACKINT                        ;reset ACKINT+=0
```

Note the use of De Morgan's theorems to implement high-true functions using normally low-true output devices. IOC.IAKSL+ is simply a half-cycle delayed version of IOC.IAKS+. Since IOC.ACKINT+ and IOC.IAKSL+ are both clocked at the start of the long half cycle, the difference between IOC.IAKS+ and IOC.IAKSL+ can be detected and be used to generate a single cycle pulse after the leading (falling) edge of BP.IAK- (product terms 2 and 3). Product terms 4 and 5 provide the necessary qualifications for generating IOC.ACKINT+. The EXT gets an acknowledgement only if the IOC is currently generating an interrupt request (product term 5) and it has interrupt priority (product term 4). Product term 6 is used to return IOC.ACKINT+ to its inactive state (low) after being active for only one clock cycle.

The IOC.ACKINT+ qualified interrupt acknowledge is inverted through a buffer [IO1204@19D] to inform the EXT that it may generate an XP.IAK- on the extender backplane to service its highest priority interrupt request.

Generating the Acknowledgement

The EXT handles the interrupt acknowledgement by generating XP.IAK- and waiting until the trap cell fetch.

IO.ACKINT- is terminated on the EXT end of the cable using pull-up/pull-down resistors, and is received into an inverter [IO0904@11C] to create the buffered EXT.ACKINT+ signal. Although EXT.ACKINT+ travels to the EXTP1 DMA Control PAL, the PAL equations reveal that it is not used there (a vestigial function). EXT.ACKINT+ is sampled by a D-type flip-flop [EX1204@13C] to satisfy the cable timing requirements. The time-shifted EXT.ACKINTS+ signal goes to the EXTP3 Interrupt and Slave Control PAL [EX1105@15D].

The EXTP3 PAL uses EXT.ACKINTS+ to set the EXT.IAKFF- flip-flop.

```
--- PAL equation for EXT.IAKFF-
IAKFF := ACKINTS*BPONL                    ;ACKINT
       + IAKFF*/EXTGOS*BPONL              ;remain through MRQ & BUSY
       + IAKFF* EXTGOS*/BUSYFF*BPONL      ;remain till trap cell fetch
```

The EXT.IAKFF- sets immediately upon the receipt of EXT.ACKINTS+ (product term 1). Product term 2 is redundant because all DMA has been suspended when the interrupt request is received. Also, the DMA queue is emptied before the IOC makes an interrupt request on the CPU backplane. The third product term keeps EXT.IAKFF- set until EXT.BUSYFF+ is set to indicate that the information for the trap cell fetch has been received by the EXT.

External to the EXTP3 PAL, EXT.IAKFF- is inverted by a gate [EX1206@18C] prior to its use by the XP.IAK- NAND buffer [EX0906@19C]. The EXT.EXTGO+ qualifier always tests true when EXT.IAKFF- is set. After the EXT asserts XP.IAK-, it waits for a response from the highest priority interrupting interface card, in the form of a memory cycle known as the vectored trap cell instruction fetch.

Handling the Vectored Trap Cell Instruction Fetch

The interrupting card's response to an interrupt acknowledge takes the form of a memory read using the card's select code as the memory address. Most of the lowest 64 (decimal) memory locations in the processor's physical memory space are designated the interrupt trap cells. There is a one-to-one correspondence between each trap cell and its associated I/O card; the correspondence mechanism is the card's select code. The trap cell contents are usually jump instructions to the software interrupt handling routines elsewhere in main memory.

XP.MEMGO- is asserted by the I/O card responding to the XP.IAK- interrupt acknowledgement. On the EXT board, the interrupt trap cell fetch is handled as if it were a DMA read operation with an empty DMA queue (no need to wait for earlier DMA writes to complete). EXT.BUSYFF+ sets, and the addressing information is read off of the extender backplane. EX.ABDBAV- is sent to the IOC with valid addressing information on the cable. EXT.IAKFF- is cleared with the assertion of EXT.BUSYFF+, indicating that the interrupt acknowledge has received its expected response (product term 3).

```
--- PAL equation for EXT.IAKFF-
IAKFF := ACKINTS*BPONL                    ;ACKINT
       + IAKFF*/EXTGOS*BPONL              ;remain through MRQ & BUSY
       + IAKFF* EXTGOS*/BUSYFF*BPONL      ;remain till trap cell fetch
```

The EXT keeps EXT.BUSYFF+ set (XP.BUSY- low on the extender backplane) until the end of the trap cell fetch on the CPU backplane, signalled by the receipt of IO.DATAAV- from the IOC.

When the IOC receives EX.ABDBAV-, it sets IOC.MEMGOFF- to initiate the trap cell fetch on the CPU backplane.

```
--- PAL equation for IOC.MEMGOFF-
MEMGOFF := /MEMGOFF*MRQFF*/MCHID*/BUSYL*ABDBAV*IOPON
                                        ;ABDBAV received while memory
                                        ; is not busy
        + /MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*ABDBAV*IOPON
                                        ;ABDBAV received and OK to
                                        ; MEMGO on next cycle
                                        ; since this one is finishing
        + /MEMGOFF*MRQFF*/MCHID*/BUSYL*MEMGOPD*IOPON
                                        ;pending MEMGO, mem. not busy
        + /MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*MEMGOPD*IOPON
                                        ;pending MEMGO, OK next cycle
        + /MEMGOFF*INTRQFF*IAKB*/ICHID*ABDBAV*IOPON
                                        ;trap cell fetch immediately
        +  MEMGOFF*MRQFF*/MCHID*BUSYL*VALIDL*IOPON
                                        ;recovery from A600+ CPU
                                        ; preemptive memory access
                                        ; occurring during first
                                        ; cycle of MRQ-
```

Only product term 5 applies to the IOC.MEMGOFF- generation for a trap cell fetch. Since the CPU backplane is quiescent due to the assertion of BP.IAK-, the trap cell fetch IOC.MEMGOFF- always translates into a valid BP.MEMGO- assertion. The other qualifiers eliminate this product term when the IOC is not in the interrupt trap cell handling mode (IOC.INTRQFF- true, IOC.IAKB+ true, and IOC.ICHID- false). After the IOC's assertion of BP.MEMGO-, the CPU deasserts BP.IAK- and the memory system performs the memory read.

At the end of the memory read cycle for the trap cell fetch, the IOC reads the returned data off of the backplane data bus, as it normally does for a DMA read access. The CPU also reads the data bus, but it interprets the data as an instruction. The IOC sends an IO.DATAAV- message to the EXT at the conclusion of the memory cycle, which causes the assertion of EXT.VALIDFF- and the subsequent completion of the memory read handshake. Data is returned to the I/O interface card, but it is ignored because it is not an I/O instruction.

Diagnose Modes 1 and 2 and the Interrupt Chain

The OTA/B 2 instruction is normally used to load the Global Register on the I/O interface cards to enable one I/O card for subsequent programmed-I/O handling. Once a specific I/O card has been addressed and enabled, programmed-I/O instructions may be used to read and write into a set of interface card registers (including control, configuration, and data transfer registers).

When OTA/B 2 is executed with data of 20B (octal) through 77B, the I/O card whose select code matches the value of the Global Register responds to I/O instructions if the Global Register mode is enabled (CLF 2). Global Register values of 10B through 17B are reserved, since the VCP program disallows the use of I/O interface select codes in that range. Performing OTA/B 2 with data values of 0B through 7B writes into a companion register known as the Diagnose Register. There are eight diagnose modes, one for each possible value. Refer to the HP 1000 A/L-Series I/O Interfacing Guide for more information about the operation of the Global and Diagnose Registers.

Diagnose Mode 0 (0 stored in the Diagnose Register) corresponds to the normal operating mode of the I/O system. Diagnose Modes 4 through 6 are not implemented, while Diagnose Modes 3 and 7 affect only the interface selected by the value written into the Global Register. Only Diagnose Modes 1 and 2 affect the entire I/O system. Both diagnose modes use the LIA/B 2 instruction to sequence through all I/O cards installed in the system; the interrupt chain is used to sort out which card is enabled to respond to the next LIA/B 2 command. This serial polling scheme requires the interrupt chain to assume a linear topology instead of the I/O extender's normal two-level hierarchical mode of operation.

The I/O extender accommodates the dual operational modes by dynamically switching out of the hierarchical mode and into the linear mode when there are no active interrupt requests pending on the I/O extender backplane. This technique is transparent to software and existing hardware because Diagnose Mode 1 and 2 operations are always performed in the absence of normal interrupt requests (since the same interrupt chain is used for both functions).

When there are no interrupt requests pending on the I/O extender backplane, the interrupt chain exiting the lowest priority I/O card is in the enabled state. If this information is fed back to the IOC's BP.ICHOD- generator, it does not affect BP.ICHOD- in any way. If there are pending interrupt requests on the extender backplane, IOC.INTRQFF- shuts down the interrupt chain locally. If the extender's bottom-of-interrupt chain data is now used by the IOC's BP.ICHOD- generator, it still has no effect because IOC.INTRQFF- has already made BP.ICHOD- low. Therefore, it is safe to merge the extender's bottom-of-interrupt chain status into the IOC's BP.ICHOD- generator.

At the other end of extender backplane's interrupt chain (top-of-chain), it is also acceptable to feed the IOC's BP.ICHID- into the extender. If the IOC is disabled from responding to a BP.IAK-, it does not tell the EXT to issue an XP.IAK-, so the state of the interrupt chain on the extender backplane is immaterial.

To accommodate Diagnose Modes 1 and 2, the incoming BP.ICHID- on the IOC card is split into three paths. BP.ICHID- directly affects BP.ICHOD- in the first path. With the second path, BP.ICHID- affects the IOC's response to BP.IAK-. The third path involves the passage of BP.ICHID- to the extender backplane.

The outgoing BP.ICHOD- also merges three interrupt chain disabling conditions. First, there is BP.ICHID- from a higher priority interface. Secondly, the assertion of IOC.INTRQFF- also affects BP.ICHOD-. Lastly, the extender's bottom-of-interrupt chain status is qualified locally on the IOC before its use in the generation of BP.ICHOD-.

The first LIA/B 2 executed during Diagnose Mode 1 or 2 involves the highest priority I/O card. Subsequent LIA/B 2 execution is determined by having the currently executing card enable the next card for the next LIA/B 2. After a period of time, the I/O card immediately above the IOC is enabled to handshake the LIA/B 2 instruction. After it starts the I/O handshake, it changes the state of the interrupt chain so that the IOC sees BP.ICHID- high.

BP.ICHID- is sampled by a D-type flip-flop [IO0806@12B] at the start of the long half cycle to satisfy both backplane and cable timing. The time-shifted version, IOC.ICHIDL-, is inverted first by a gate [IO0404@17D], and again with an inverting buffer [IO0704@19D] to drive the IO.IOCICHOD- cable message for the EXT board. The EXT receives IO.IOCICHOD- with a resistor network [EX1003@10D] and an inverting buffer [EX0904@11C] to produce EXT.IOCICHOD+. A D-type flip-flop [EX1104@13D] samples EXT.IOCICHOD+ at the start of the short half cycle to satisfy cable protocol timing. The EXT.IOCICHODS+ result is not consistent with the extender's backplane timing requirements, so it is sent through another D-type flip-flop [EX1106@12A] to delay the signal by a half clock cycle. EXT.ICHOD+ is finally inverted by a gate [EX1206@18A] to provide the correct polarity for the top-of-interrupt chain information on the extender backplane.

By the time the next LIA/B 2 instruction is executed (during Diagnose Mode 1 or 2), the highest priority interface card on the extender backplane is enabled to respond. The chain enable is propagated one card at a time after each successive LIA/B 2. Eventually, the very last I/O card on the extender backplane executes the LIA/B 2 instruction and drive its XP.ICHOD- high to enable the next lower priority card. This status information must be passed back to the CPU backplane to allow the I/O card below the IOC to continue the Diagnose Mode 1 or 2 sequence.

This feature is implemented with an Interrupt Chain Jumper (ICJ) card. Although there are two different ICJ cards (12025-60003 and 12025-60004), one for the 12025A Extender and another for the 12025B Extender, their interrupt chain jumpering capabilities are identical. The ICJ card receives XP.ICHOD- from the lowest priority I/O card and jumpers this signal line to an unused bussed signal line on the extender backplane for reception by the EXT card. The XP.ICHOD- status is sent back up the backplane as XP.INTCHDIS-, using the backplane line that is normally used as FETCH- on the CPU backplane. The extender does not use FETCH-, the CPU's instruction fetch status signal.

XP.INTCHDIS- has a pull-up resistor on the EXT to handle the situation when the optional ICJ is not used. In that case, the IOC is at the bottom of the CPU backplane's interrupt chain, so the bottom-of-interrupt chain on the extender backplane is also the bottom of the system's logical interrupt chain.

As with XP.INTRQ-, XP.INTCHDIS- is considered an asynchronous signal, and as such, it must pass through two levels of sampling with the system clock before it is usable. The first sampling stage occurs at a D-type flip-flop [EX0504@17D] and the second sampling stage is handled by the EXTP3 Interrupt and Slave Control PAL [EX1105@15D].

```
--- PAL equation for EXT.INTCHDISLL+
/INTCHDISLL := /BPONL                    ;initialize INTCHDISLL+=0
             + /INTCHDISL                ;delayed version of INTCHDISL
```

EXT.INTCHDISLL+ is buffered onto the cable with an inverter [EX0704@19D]. The EX.EXTICHOD- message passed to the IOC is the doubly clocked bottom-of-interrupt chain information. This double cycle shift corresponds to the shift treatment given XP.INTRQ- in generating EX.INTFLG-, so if both messages are active at the same time, the IOC ignores EX.EXTICHOD-.

The IOC receives EX.EXTICHOD- with a resistor network [IO0702@10D], and inverts the signal using a buffer [IO0904@12D] to generate IOC.EXTICHOD+. The IOCP4 PAL decides when IOC.EXTICHOD+ is used.

```
--- PAL equation for IOC.EXINTCH-
EXINTCH := EXTICHOD*/IAKS*/INTFLG*/INTRQFF*BPONL
                                  ;pass on extender INT priority
                                  ; chain except during IAK,
                                  ; INTFLG, and INTRQFF (used only
                                  ; for proper sequencing during
                                  ; Diagnose Mode 1 or 2 polling;
                                  ; IOC.INTRQFF- is used to generate
                                  ; IOC.ICHOD- for determining
                                  ; normal interrupt priority
```

The single product term prevents the use of IOC.EXTICHOD+ in generating IOC.EXINTCH- whenever normal interrupt processing is detected (BP.IAK- true, or EX.INTFLG- true, or IOC.INTRQFF- true). During Diagnose Mode 1 and 2 operations, IOC.EXTICHOD+ directly sets IOC.EXINTCH-, which is used by the IOC's BP.ICHOD- generator [IO1205@19B]. Until the lowest priority I/O card on the extender backplane has responded to an LIA/B 2, IOC.EXINTCH- is true, which makes BP.ICHOD- low. Three clock cycles after XP.INTCHDIS- goes high, BP.ICHOD- goes high to enable the I/O card directly below the IOC for the next LIA/B 2.

The Slave Mode (Break Mode) Feature

Slave mode is invoked by any I/O interface that is enabled for such operations to cause some change in the CPU's operating environment by direct control from the I/O interface. All I/O handshaking instructions (SFS/C, OT*, MI*, LI*, HLT) are really dedicated implementations of the slave mode feature. In complexity, HLT instructions require the most I/O handshake cycles. However, all of these instructions must reside in the CPU's instruction stream for execution. The A/L-Series I/O architecture allows for a non-programmed method of invoking such I/O handshakes.

Although the architecture could use priority chain lines to arbitrate among several slave mode requestors, only one I/O interface may ever be designated the "break enabled" device. The break enabled interface is the only card which has the capability to make a slave request and to perform the subsequent I/O handshaking. It is also the only interface which has the capability to respond to the HLT instruction. The setting of the Global Register has no effect on which interface is enabled to respond to the HLT instruction. The break enabled interface card has the sole responsibility for HLTs and slave requests.

The Slave Priority Chains

Only one I/O card is allowed to make a slave request, therefore it is somewhat redundant to have slave priority chains for resolving priority. The slave priority chains work the same way as the DMA and interrupt priority chains. An AND gate [IO0805@19B] generates the BP.SCHOD- output from the BP.SCHID- (output from higher priority I/O card) and the IOC.SLAVEFF- inputs.

As a matter of reference, keep in mind that the BP.SCHID- and BP.SCHOD- signals (also XP.SCHID- and XP.SCHOD- signals) are normally low except for the one clock cycle when the CPU acknowledges the slave request.

Requesting a Slave Mode Handshake

For a break-enabled interface located on the extender backplane, it makes a slave mode handshake request by asserting XP.SLAVE-. The EXT terminates the open-collector line with a resistor network [EX1007@10A] prior to receiving the signal with an inverting buffer [EX1107@11A]. The buffered signal is sampled at the start of the long half cycle using a D-type flip-flop [EX1106@12A] to generate EXT.SLAVEL+.

EXT.SLAVEL+ is time shifted by a full clock cycle and also inverted using the EXTP3 Interrupt and Slave Control PAL [EX1105@15D] to create EXT.SLAVELL-.

```
--- PAL equation for EXT.SLAVELL-
SLAVELL := SLAVEL * BPONL           ;delayed version of SLAVEL
```

An AND gate [EX0804@17D] combines the EXT.SLAVEL+ and EXT.SLAVELL- signals to form a single-cycle high-true pulse. EXT.REQSLV+ is inverted by a buffer [EX0704@19D] to send the IOC an EX.REQSLV- message. The EXT and the extender backplane now wait for an acknowledgement.

On the IOC board, EX.REQSLV- is terminated at a resistor network [IO0702@10D] before its reception at an inverting buffer [IO0904@12D]. IOC.REQSLV+ is used by the IOC to set the Slave Mode Flag, a J-K flip-flop [IO0504@14E]. The flip-flop is updated at the start of the long half cycle, so this satisfies both the cable and CPU backplane timing protocols. The low-true output of the flip-flop, IOC.SLAVEFF-, is used to generate the backplane request and to inform the IOCP2 and IOCP4 PALs of the current Slave Mode operation.

The assertion of IOC.SLAVEFF- automatically disables the slave priority chain leaving the IOC. This logic is performed at the AND gate [IO0805@19B] which generates BP.SCHOD-. IOC.SLAVEFF- is also inverted by a gate [IO1007@18B] to drive the open-collector backplane slave request signal, BP.SLAVE-.

Acknowledging the Slave Mode Request

The CPU checks the status of the slave request signal (and other interrupting conditions) at the completion of every instruction. When BP.SLAVE- is true, the processor is diverted from its current instruction processing and starts processing the slave request. The CPU makes an acknowledgement by pulsing BP.SCHOD- high for one clock cycle at the start of the short half cycle. Allowing for propagation through the priority chain's AND gates from higher priority interface cards, the IOC samples BP.SCHID- at the start of the next short half cycle, therefore allowing one full clock cycle for the signal to propagate down the chain. IOC.SCHOD- is low when it leaves the IOC because IOC.SLAVEFF- is set.

A D-type flip-flop [IO1006@11C] creates the synchronized versions IOC.SCHIDS- and IOC.SCHIDS+. IOC.SCHIDS- is used as the K-input to the Slave Request flag flip-flop to clear the flag. When the flag is cleared, BP.SLAVE- is removed from the backplane. Now that the CPU has acknowledged the slave request, it waits for the I/O handshake request.

IOC.SCHIDS+ is received by the IOCP2 PAL [IO1004@15B].

```
--- PAL equation for IOC.SPECIORQFF-
SPECIORQFF := /SPECIORQFF*OT02L*IOPON  ;OT* 0 and OT* 2
            +  SPECIORQFF*/QRNIL*SCHIDS*IOPON
                                       ;stay in SPECIORQ mode
            +  SPECIORQFF*/QRNIL*/SLAVEFF*IOPON
                                       ; until next fetch/slave
```

IOC.SCHIDS+, in conjunction with IOC.SLAVEFF-, is used by the IOCP2 PAL to clear the IOC.SPECIORQFF- flag (product terms 2 and 3). This action prepares the I/O Handshaking PAL to process normal I/O handshakes, not the OT* 0/OT* 2 variety.

IOC.SCHIDS+ is also received by the IOCP4 PAL [IO1005@15D]. The IOCP4 Misc Functions PAL is responsible for generating the acknowledgement message to the EXT.

```
--- PAL equations for IOC.SCHIDSL+ and IOC.ACKSLV+
/SCHIDSL := BPONL*/SCHIDS          ;initialize SCHIDSL+=1,
                                   ;delayed version of SCHIDS+

/ACKSLV := /BPONL                  ;initialize ACKSLV+=0
         + /ACKSLV* SCHIDS         ;one cycle pulse to inform EXT
         + /ACKSLV*/SCHIDSL        ; to process slave request
         + /ACKSLV*/SLAVEFF        ;no ACKSLV if no slave req from EXT
         +  ACKSLV                 ;reset ACKSLV+=0
```

IOC.SCHIDS+ is time-shifted by a half clock cycle to create IOC.SCHIDSL+. The difference between IOC.SCHIDS+ and IOC.SCHIDSL+ (when IOC.SLAVEFF- set) is detected and a one cycle IOC.ACKSLV+ pulse is created. Since this is a high-true output implemented in a normally low-true part, the equations satisfy the case when IOC.ACKSLV+ is not true. IOC.ACKSLV+ goes high when: IOC.SCHIDS+ is false (low), IOC.SCHIDSL+ is true (high), and IOC.SLAVEFF- is true (low). IOC.ACKSLV+ is inverted by a buffer [IO1204@19D] to create the IO.ACKSLV- cable message.

The EXT receives IO.ACKSLV- into a resistor network [EX1003@10C]. The signal is inverted by a buffer [EX0904@11C] to generate EXT.ACKSLV+, which is subsequently sampled at the start of the short half cycle using a D-type flip-flop [EX1204@13C]. The output of the flip-flop satisfies both the polarity and timing requirements for the slave acknowledge signal on the extender backplane, so XP.SCHOD- emerges directly from the flip-flop. The slave requesting interface receives XP.SCHOD- pulsing high for one cycle as a cue to start the slave mode transfers by making an I/O handshake request.

The EXT also uses XP.SCHOD- to prepare for the ensuing I/O handshakes, which is handled by the EXTP2 I/O Handshaking PAL [EX1005@15C].

```
--- PAL equations for EXT.FIRSTHS-, EXT.SPECIO-, and EXT.SLVWAIT-
FIRSTHS := INSTRAVS*BPONL              ;get ready for I/O handshake
         + /SCHOD*BPONL                ; for instruction or slave
         +  FIRSTHS*/IOGOFF*BPONL      ;keep FIRSTHS until IOGO HS
         +  FIRSTHS*IOGOFF*/IOGO3*BPONL ; through IOGO1 and IOGO2
         +  FIRSTHS*IOGOFF*IOGO3*IORQL*BPONL
                                       ; hold if IOGO3 extended
         + /FIRSTHS*IOGOFF*IOGO3*/IORQL*BPONL
                                       ;ready for third handshake

SPECIO := INSTRAVS*ACKIOHSS*BPONL      ;SPECIO+=1 if OT* 0 or OT* 2
        + SPECIO*/INSTRAVS*SCHOD*BPONL ;till next instr/slave ack

SLVWAIT := /SLVWAIT*IOGOFF*IOGO3*/IORQL*/FIRSTHS*/OTAFLAG*BPONL
                                       ;HLT/break
                                       ; multiple handshake sync
         + SLVWAIT*/IOGOFF*/INSTRAVS*SCHOD*BPONL ;till IOGOFF set
                                       ; or next I/O instruction
                                       ; or next slave acknowledge
```

Regardless of the previous state of EXT.FIRSTHS-, when XP.SCHOD- pulses high, the first (or odd-numbered) handshake flag is forced true. Reflecting a similar action performed on the IOC, XP.SCHOD- also forces the EXT.SPECIO- flag to clear, since none of the slave mode transfers involve special handshaking considerations. Finally, EXT.SLVWAIT- is initialized to false.

Handling the Slave Mode Transfers

As with the handling of I/O handshakes for I/O instructions, slave mode transfers also begin with the I/O card's assertion of XP.IORQ-. Unlike the I/O instruction case, EXT.MASKIORQC- is not in operation, so it is not necessary to wait several clock cycles before sampling XP.IORQ-. Aside from that difference, the balance of the slave mode transfer is identical to the handshaking used to execute the HLT instruction. Refer to that HLT discussion for more details.

Operation of Multiple Extenders

The I/O extender hardware architecture was developed to make multiple extender operation transparent to the CPU, to other I/O cards, and even to other I/O extenders. To accomplish this task, each I/O extender must mimic the functionality of the standard I/O cards, and be able to control the CPU using arbitration without getting in the way of other I/O cards or other I/O extenders. Once this "virtual I/O card" approach is implemented for one I/O extender, this transparency permits the use of multiple I/O extenders in order to accommodate the 48 select code limit of the architecture.

Instruction Processing

With either A600+/A700 or A900 processor mode, the I/O extenders must prevent the CPU from using the backplane to broadcast the next I/O instruction before all I/O extenders empty their DMA queue. Since all I/O extenders decode/detect the impending broadcast simultaneously, all IOCs assert BP.MRQ- for at least two cycles to give the DMA queue a chance to empty. Because the arbitration is performed in parallel, there is no additional performance penalty for having more than one I/O extender. If all of the I/O extenders have pending DMA cycles in the wait queue, those DMA transfers will all complete, one at a time according to DMA priority, before the CPU is allowed to broadcast the I/O instruction.

If the I/O instruction broadcast is an OT* 0 or OT* 2, all I/O extenders go into the special I/O handshaking mode and retrieve the outgoing data at the same time. None of the I/O extenders will make an I/O handshake request (BP.IORQ-).

DMA

DMA operations through one I/O Extender do not affect DMA operations through another I/O Extender except for the normal priority chain interactions.

Interrupts

There is one situation in which the normal protocols are modified to accommodate the DMA queue if two I/O extenders have interrupt requests pending when the CPU acknowledges the interrupt. This situation only arises when multiple I/O extenders are used with the A700 CPU, due to its non-standard timing for the assertion of BP.IAK-. Assume that the lower priority extender (of two extenders) has been making an interrupt request for some time. The higher priority extender is ready to make a DMA request when the CPU acknowledges the interrupt. If the CPU is an A600+ or an A900, the DMA cycle is allowed to preempt the interrupt acknowledge, the DMA queue allowed to empty, and the CPU tries again with BP.IAK-. At that time, the higher priority extender responds to the interrupt acknowledge, and the lower priority extender to the next interrupt acknowledge.

The situation is different for the A700 CPU. Because BP.IAK- is asserted prior to the long half cycle (arbitration period), the DMA transfer is prohibited from proceeding and making a backplane DMA request. However, the higher priority extender also cannot proceed with the interrupt acknowledge because the resulting trap cell fetch would overwrite the DMA transfer information. This deadlock situation is avoided because the IOC disallows interrupt requests when there is a DMA transfer waiting in the DMA pipeline. Therefore, the higher priority extender is forced to relinquish its claim to the interrupt priority and allows the lower priority extender to respond to the first interrupt acknowledgement. At a suitable time, the queued up DMA transfers are launched by the higher priority extender, emptying the DMA queue so that it is able to generate its own BP.INTRQ- and to respond to the next interrupt acknowledgement.

Slave Mode

Since only one I/O interface is enabled to make a slave request, it is not possible to have a contention between several I/O extenders making simultaneous slave requests.

Chapter 6
Removal and Replacement

Introduction

This chapter provides procedures for removal and replacement of assemblies in the HP 12025A/B I/O Extender.

Electrical Safety

Before proceeding with any maintenance or service on the extender which requires physical contact with electrical or electronic components, be sure that either power is removed or that safety precautions are followed to protect against shock. Heed all "CAUTION" and "WARNING" labels on the extender. All service work must be done by qualified personnel.

Required Tools

No tools other than ordinary hand tools are required.

Assembly Removal & Replacement (HP12025A)

WARNING

Hazardous voltages are present inside the system mainframe. Heed all WARNING - HAZARDOUS VOLTAGE labels.

The following paragraphs describe procedures for removing and replacing the various 12025A assemblies. (See Figures 9-3 and 9-4 for exploded views of the 12025A.) It is assumed that the I/O extender is a freestanding device. If it is rack mounted, read the entire assembly removal procedure and refer to Figures 9-3 and 9-4. When it is obvious that the procedure cannot be performed with the extender in the rack, do the following:

a. Set the Power switch to the OFF position and disconnect the power cord.

b. Disconnect all cables from the extender plug-in cards.

c. Remove all screws securing the extender in the rack.

d. Remove the extender from the rack.

Plug-In Cards (HP 12025A)

CAUTION

STATIC SENSITIVE DEVICE. Use anti-static handling procedures when removing or installing an extender plug-in card.

Removal

Remove a plug-in card from the extender card cage as follows:

a. Set the Power switch to OFF. Open the rear covers.

b. If applicable, remove the cable connector from the plug-in card, and disconnect the wire from the grounding strip. Remove the plug-in card by pulling outward on the card extractor levers.

Replacement

Replace a plug-in card by reversing the removal procedure.

NOTE

If a replacement card is being installed, be sure the card's switches are set properly.

Power Supply (HP 12025A)

Removal

To remove the extender power supply, proceed as follows:

a. Set the Power switch to OFF and disconnect the power cord from the rear of the extender.

b. Remove the front cover by grasping it at the sides and firmly pulling it away from the extender. Remove the cable connector cover from the front of the power supply. (See Figure 4-1.)

c. Disconnect the power cable and voltage configuration cable from the power supply.

d. Remove the four screws securing the front of the power supply to the mainframe. Grasp the power supply by its handle and carefully pull it out of the mainframe.

Replacement

CAUTION

When installing a replacement power supply, be sure to configure the power supply for 115- or 230-Vac operation as appropriate for the installation. (If the line voltage is 230 Vac, power supply connector J2 must be used.)

Replace the power supply by reversing the removal procedure.

Fans (HP 12025A)

Removal

To remove a fan, proceed as follows:

a. Set the Power switch to OFF and disconnect the power cord from the rear of the extender.

b. Remove the three screws securing the power distribution unit (PDU) at the rear of the extender.

c. Remove the extender front panel by grasping it at the sides and firmly pulling it away from the extender. Remove the cable connector cover from the front of the power supply.

d. Disconnect the power cable and voltage configuration cable from the front of the power supply.

e. Remove the two screws securing the front of the PDU. Grasp the PDU and carefully pull it out of the front of the extender.

f. Place the PDU on a workbench and disconnect the power cord from the defective fan. Remove the four screws securing the fan and remove the fan.

Replacement

CAUTION

When installing a fan, be sure to orient the fan so that the direction of air flow is out of the extender. (Air flow direction is indicated on the fan.)

Replace the fan by reversing the removal procedure.

Card Cage Backplane (HP 12025A)

Removal

To remove the card cage backplane, proceed as follows:

a. Set the Power switch to OFF and disconnect the power cord.

b. Remove all of the plug-in cards from the card cage by using the procedure given previously.

c. Remove the extender power supply by using the procedure given previously.

d. Remove the two screws securing the front of the Power Distribution Unit (PDU), and the three screws securing its rear.

e. Remove the screws securing the top and sides of the shroud, including the three screws inside the front chamber. Remove the shroud.

f. Remove the screws securing the RFI shield to the backplane and remove the shield.

g. Remove the screws securing the backplane and remove the backplane.

Replacement

Replace the backplane by reversing the removal procedure.

Assembly Removal & Replacement (HP12025B)

WARNING

Hazardous voltages are present inside the system mainframe. Heed all WARNING - HAZARDOUS VOLTAGE labels.

The following paragraphs describe procedures for removing and replacing the various 12025B assemblies. (See Figures 9-5 and 9-6 for exploded views of the 12025B.) It is assumed that the I/O extender is a freestanding device. If it is rack mounted, read the entire assembly removal procedure and refer to Figures 9-5 and 9-6. When it is obvious that the procedure cannot be performed with the extender in the rack, do the following:

a. Set the Power switch to the OFF position and disconnect the power cord.

b. Disconnect all cables from the extender plug-in cards.

c. Remove all screws securing the extender in the rack.

d. Remove the extender from the rack.

Plug-In Cards (HP 12025B)

CAUTION

STATIC SENSITIVE DEVICE. Use anti-static handling procedures when removing or installing an extender plug-in card.

Removal

Remove a plug-in card from the extender card cage as follows:

a. Set the Power switch to OFF. Open the rear covers.

b. If applicable, remove the cable connector from the plug-in card, and disconnect the wire from the grounding strip. Remove the plug-in card by pulling outward on the card extractor levers.

Replacement

Replace a plug-in card by reversing the removal procedure.

NOTE

If a replacement card is being installed, be sure the card's switches are set properly.

Front Panel (HP 12025B)

Removal

Remove the extender front panel as follows:

a. Grasp the front panel by the two indented handles at the sides of the panel.

b. Firmly pull the panel away from the extender chassis.

Replacement

Replace the front panel by reversing the removal procedure.

Fan Panel (HP 12025B)

Removal

Remove the fan panel as follows:

a. Set the LINE switch to OFF and disconnect the power cord.

b. Remove the front panel as described above.

c. Remove the four screws, four lock washers, and four flat washers securing the cover plate to the fan panel and remove the plate.

d. Remove the nine screws, nine lock washers, and nine flat washers, and remove the fan panel from the extender chassis.

e. Disconnect the line configuration/fan power connector from the power supply. Note whether the connector is in P7 or P8.

Replacement

CAUTION

When connecting the line configuration/fan power connector to the power supply, use connector P7 (see Figure 6-1) if ac line input voltage is 230 Vac.

Replace the fan panel by reversing the removal procedure.

Fans (HP 12025B)

Removal

Remove a fan as follows:

a. Set the LINE switch to OFF and disconnect the power cord.

b. Remove the front panel as previously described.

c. Remove the four screws, four lock washers, and four flat washers securing the cover plate to the fan panel and remove the plate.

d. Disconnect the individual fan power plug.

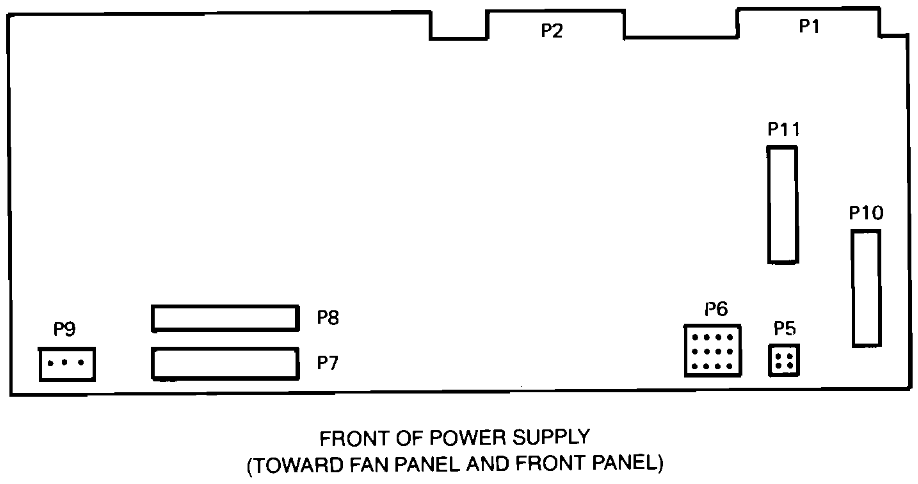
e. Remove three screws, three lock washers, and three flat washers, and remove the fan from the fan panel.

Replacement

CAUTION

When installing a fan, be sure to orient the fan so that the direction of air flow is into the extender. Air flow direction is indicated on the fan.

Replace the fan by reversing the removal procedure.



CONNECTOR	DESCRIPTION
P1	DC Output Connectors
P2	DC Output Connectors
P5	Battery Switch Connector
P6	Test Points and External Battery Connector
P7	230 Vac Line Configuration Connector
P8	115 Vac Line Configuration Connector
P9	AC Line Input Connector
P10	Battery Backup Connector (Not used)
P11	25 kHz Sine Wave Connector

8400-83

Figure 6-1. Power Supply Connectors, HP 12025B

Power Supply (HP 12025B)

Removal

CAUTION

Before removing the power supply, set the LINE switch to OFF and disconnect the power cord. Allow 90 seconds for the high voltages on the power supply to discharge.

Remove the power supply as follows:

a. Refer to Figures 6-1 and 6-2.

b. Remove the front panel and fan panel as previously described.

c. Disconnect the ac line input connector from P9.

d. Disconnect the 115-Vac/230-Vac line configuration connector from P8 or P7.

e. Disconnect the test point connector from P6.

f. Remove the two screws securing the bottom bracket of the power supply.

g. Grasp the power supply by the front edge and pull the power supply away from the backplane.

Replacement

CAUTION

All cables (ac line and test point) must be dressed for minimum length above the power supply to prevent interference with the fans.

When connecting the line configuration/fan power connector to the power supply, use connector P7 (see Figure 6-1) if the ac line input voltage is 230 Vac.

Replace the power supply by reversing the removal procedure.

25 kHz Sine Wave Module (HP 12025B)

Removal

Remove the HP 12158A 25 kHz Sine Wave Module as follows:

a. Refer to Figures 6-1 and 6-2.

b. Remove the front panel and the power supply as previously described.

c. Loosen the power supply upper plate, lift the plate, and lift the 25 kHz sine wave module from the power supply.

Replacement

Replace the 25 kHz sine wave module by reversing the removal procedure. If this is a new installation of the 25 kHz module, remove and discard the two screws in the power supply board.

Card Cage Backplane (HP 12025B)

Removal

Remove the backplane as follows:

a. Remove the plug-in cards and the power supply as previously described.

b. Remove the connectors on the right-hand (primary) side of the line filter.

c. Remove four screws and remove the line filter mounting bracket assembly.

d. Remove four screws and remove the left guide mounting bracket.

e. Remove the seven screws securing the RFI shield to the backplane and remove the shield.

f. Remove seven screws and remove the backplane.

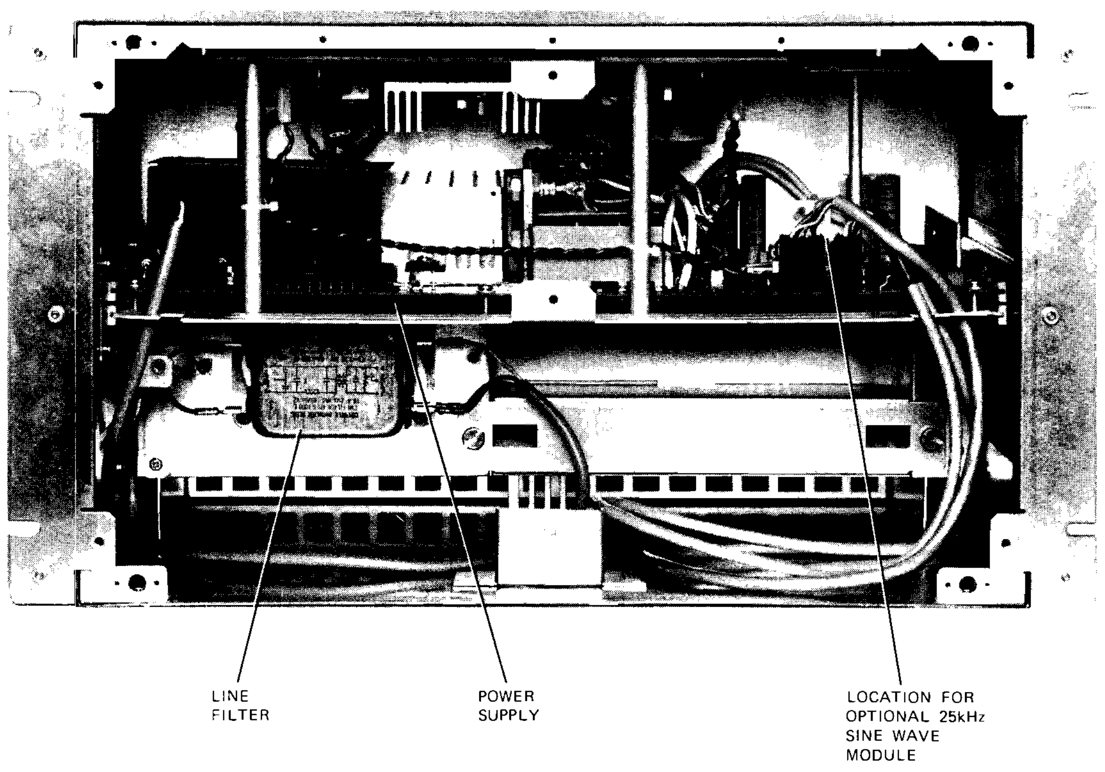


Figure 6-2. HP 12025B Front View, Cover and Fan Panel Removed

Replacement

CAUTION

When reconnecting the wires to the line filter, connect the white wire to the top lug. Connect the black wire and the green wire to the bottom lugs, with the green one closest to the backplane.

Replace the backplane by reversing the removal procedure.

Line Filter

Removal

Remove the line filter as follows:

a. Refer to Figure 6-2.

b. Remove the fan panel as previously described.

c. Remove four screws and four washers and remove the line filter.

Replacement

CAUTION

When reconnecting the wires to the line filter, connect the white wire to the top lug. Connect the black wire and the green wire to the bottom lugs, with the green one closest to the backplane.

Replace the line filter by reversing the removal procedure.

Chapter 7
Adjustments

There are no adjustments in the HP 12025A/B I/O Extender.

Chapter 8 Troubleshooting and Diagnostics

Introduction

This chapter provides troubleshooting information for the HP 12025A/B I/O Extender.

Electrical Safety

Before proceeding with any maintenance or service on the extender which requires physical contact with electrical or electronic components, be sure that either power is removed or that safety precautions are followed to protect against shock. Heed all "CAUTION" and "WARNING" labels on the extender. All service work must be done by qualified personnel.

Equipment Required

A voltmeter and the host computer system are required to troubleshoot the I/O extender.

NOTE

There are no diagnostics for the I/O extender. However, HP 24612A diagnostics can be used to test I/O interface cards installed in the extender (the HP-IB card diagnostic must have revision 2440 or later). The operating procedures for the 24612A diagnostics are the same for I/O cards in the extender as they are for I/O cards in the CPU card cage.

Initial Checkout Failure

The initial checkout procedure given in Chapter 3 only checks the dc voltages in the extender and in the host computer. If the checkout fails, do the following:

a. Using a voltmeter, check the extender dc voltages at the rear-panel test connector (12025B) or on the front edge of the Interrupt Chain Jumper (ICJ/12) card. See Table 8-1 or Figure 8-1. (Install the ICJ/12 card if it is not already installed.) If any dc voltage is out of tolerance, replace the extender power supply.

b. Check the computer dc voltages as instructed in the computer installation and service manual. If any dc voltage is out of tolerance, follow the instructions given in the computer manual.

c. If the dc voltages are good, place the EXT card in a different slot in the extender and determine whether the initial checkout procedure fails in the new slot.

   1. If the checkout fails in only one slot, inspect the backplane connector of that slot for possible damage. Replace the backplane if necessary and repeat the checkout procedure to verify the fix.

   2. If the checkout fails in both slots, go to step _d_.

d. Place the IOC card in a different I/O slot in the host CPU and determine whether the initial checkout procedure fails in the new slot.

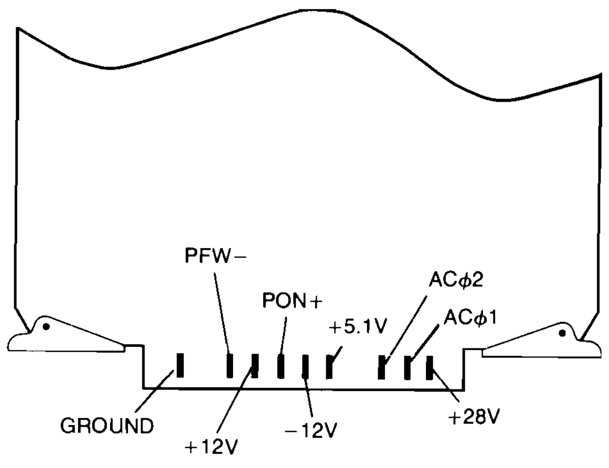
   1. If the checkout fails in only one slot, inspect the backplane connector of that slot for possible damage. Replace the backplane if necessary and repeat the checkout procedure to verify the fix.

   2. If the checkout fails in both slots, troubleshoot by swapping out, in the following order, the EXT card, the IOC card, and the extender cable. After each swap, perform the checkout procedure.

Table 8-1. Power Supply Voltages, HP 12025B

TEST CONN. PIN NO.	TEST POINT	NOMINAL VOLTAGE
1	+5	5.1V +/-0.10V
2	+12	12.0V +0.72V -0.36V
3	-12	-12.0V +/-0.72V
4	+5M	Not valid
5	PON+ (Power On)	5.1V +/-0.10V
6	PFW- (Power Fail Warning)	5.1V +/-0.10V
7	MLT- (Mem. Lost)	Not valid
8	ø1	27.0 Vrms +/-2.16V
9	ø2	27.0 Vrms +/-2.16V
12	Common	

TEST POINTS ARE ON ICJ/12 CARD



VOLTAGE	RANGE*
PFW-	4.9 — 5.2
+12V	11.6 — 12.7
PON+	4.9 — 5.2
−12V	−11.2 — −12.7
+5.1V	4.99 — 5.20
ACø1	18.0 — 21.1 VRMS
ACø2	18.0 — 21.1 VRMS
+28V	22.4 — 33.6

*UNDER NORMAL LOAD CONDITIONS

8300-80A

Figure 8-1. HP 12025A Voltage Test Points

Troubleshooting Procedures

There are only three operating symptoms of trouble in the I/O extender; they are:

a. Only one I/O interface card in the extender is failing while other I/O cards are working. (This is most probably a failure of the I/O card or its associated peripheral device.)

b. All I/O cards in the extender are failing but those in the CPU are working. (This is most probably a failure of the extender.)

c. All A-Series measurement and control cards in the extender are failing but other I/O cards are working.

Troubleshooting procedures for the above symptoms are given in the following paragraphs.

NOTE

Turning off power to the I/O extender causes a power fail shutdown of the computer that may appear to be a computer failure.

Only One I/O Card Failing

If only one I/O card is failing, proceed as follows:

a. Execute a diagnostic test of the failing I/O card by using the appropriate diagnostic and the instructions given in the manual for the diagnostic. Note that the HP-IB card diagnostic must have revision 2440 or later.

b. Place the I/O card in a different slot in the extender and determine whether the diagnostic test fails in the new slot.

   1. If the test fails in only one slot, inspect the backplane connector of that slot for possible damage. Replace the backplane if necessary and repeat diagnostic test to verify the fix.

   2. If the test fails in both slots, place the I/O card in the IOC slot in the host CPU backplane and repeat the diagnostic test of the card. If the test fails, replace the I/O card. If the test passes, go to step c.

c. Reinstall the I/O card in an extender slot and swap out the EXT card. Repeat the diagnostic test.

d. If the test passes, the replaced EXT card is defective. If the test fails, swap out the IOC card and repeat the diagnostic test.

e. If the test passes, the replaced IOC card is defective. If the test fails, swap out the extender cable and repeat the diagnostic test.

All I/O Cards Failing

If all the I/O cards in the extender are failing, proceed as follows:

NOTE

With the A700 computer, all the I/O cards in the extender will fail if the A700 Lower Processor card has "old" ROMs. Verify that U91 on the Lower Processor card has part number 12152-80053 or higher.

a. Inspect the extender cable for bent or broken contacts or other damage; replace cable if necessary.

b. Verify that the I/O extender is properly installed. (Refer to Chapter 3.) If more than one extender is installed, be sure that the ICJ card is installed as necessary.

c. Turn on power to both the host CPU and the I/O extender. Verify that the green LEDs on the extender EXT and IOC cards are lit. If either LED is not lit, perform the following:

   1. Using a voltmeter, check the dc voltages at the rear-panel test connector (12025B) or on the front edge of the ICJ/12 card. See Table 8-1 or Figure 8-1. (Install the ICJ/12 card if it is not already installed.) If any dc voltage is out of tolerance, replace the extender power supply.

   2. If the extender dc voltages are good, troubleshoot by swapping out, in the following order, the EXT card, the IOC card, and the extender cable. After each swap, perform a diagnostic test on an I/O card in the extender.

All Measurement and Control Cards Failing

If only the A-Series measurement and control (MC) cards are failing, the probable cause, and corrective measure, is:

a. The 25 kHz voltages are out of tolerance. Check these voltages at the rear-panel (12025B) or on the front edge of the ICJ/12 card. See Table 8-1 or Figure 8-1. (Install the ICJ/12 card if it is not already installed.) If these voltages are not within tolerance, turn off the power for 90 seconds. Then turn the power back on. If the fault recurs, replace the 25 kHz power module.

b. A defective MC card is shorting the 25 kHz voltages. Remove all MC cards and reinstall them one at a time, checking the 25 kHz voltages to see which MC card is causing the short. Replace the defective MC card.

c. A failure of the optional 25 kHz module. Replace the module.

d. A blown fuse on the 25 kHz module (HP 12025A only). Replace the fuse. Check for a defective MC card shorting the 25 kHz voltages. If the problem persists, replace the 25 kHz power module.

e. A short on the backplane. Replace the backplane.

Extender Cable Wiring

Table 8-2 gives the pin numbers for the extender cable wiring.

Power Distribution

Power distribution diagrams for the I/O extenders are in Chapter 12.

Table 8-2. Wiring for Cable 12025-60007

P1&P2 PIN*	SIGNAL	P1&P2 PIN*	SIGNAL
1A	Ground	1B	Ground
2A	EX.AE1+	2B	EX.EXTPFW+
3A	EX.AE3+	3B	EX.EXTPON-
4A	EX.SELFC-	4B	EX.AE0+
5A	EX.AB1+	5B	EX.AE2+
6A	EX.AB3+	6B	EX.AE4+
7A	EX.AB4+	7B	EX.AB0+
8A	EX.AB6+	8B	EX.AB2+
9A	EX.AB8+	9B	EX.AB5+
10A	EX.AB10+	10B	EX.AB7+
11A	EX.AB13+	11B	EX.AB9+
12A	EX.WE-	12B	EX.AB11+
13A	EX.DB1+	13B	EX.AB12+
14A	EX.DB3+	14B	EX.AB14+
15A	EX.DB5+	15B	EX.DB0+
16A	EX.DB6+	16B	EX.DB2+
17A	EX.DB8+	17B	EX.DB4+
18A	EX.DB10+	18B	EX.DB7+
19A	EX.DB12+	19B	EX.DB9+
20A	EX.DB15+	20B	EX.DB11+
21A	EX.ABDBAV-	21B	EX.DB13+
22A	EX.INTFLG-	22B	EX.DB14+
23A	EX.REQSLV-	23B	EX.REQDMA-
24A	EX.OTAFLG+	24B	EX.REQIOHS-
25A	IO.DB0+	25B	EX.EXTICHOD-
26A	IO.DB1+	26B	IO.DB2+
27A	IO.DB3+	27B	IO.DB4+
28A	IO.DB5+	28B	IO.DB6+
29A	IO.DB8+	29B	IO.DB7+
30A	IO.DB10+	30B	IO.DB9+
31A	IO.DB12+	31B	IO.DB11+
32A	IO.DB14+	32B	IO.DB13+
33A	IO.DB15+	33B	IO.PE-
34A	IO.ACKDMA-	34B	IO.DATAAV-
35A	IO.ACKINT-	35B	IO.INSTRAV-
36A	IO.ACKIOHS-	36B	IO.ACKSLV-
37A	IO.IOCICHOD-	37B	IO.CTURNF-
38A	IO.IOCPON-	38B	IO.CNTLRST-
39A	IO.CLOCK-	39B	not used
40A	Ground	40B	Ground

* Each pin at connector P1 is connected to the corresponding pin at P2; that is, pin P1-1A is connected to P2-1A, P1-2A to P2-2A, and so on.

Chapter 9
Replaceable Parts

Introduction

This chapter provides information on the replaceable parts of the HP 12025A/B I/O Extender.

NOTE

Many of the schematic diagrams and parts lists in this document contain the generic number of IC packs. These are for your convenience in referring to pack diagrams and general operating conditions in semiconductor manufacturer's catalogs. The generic number, however, should not be used to order replacement parts. Many parts used in Hewlett-Packard equipment are purchased with special specifications and tolerances, or may undergo special testing and treatment (such as burn-in). Replacement parts therefore must be ordered using the Hewlett-Packard part number to insure that the replacement part will restore the equipment to its proper operating condition.

PCA Date Codes

Figures 9-1 and 9-2 show the date codes of the PCAs (printed circuit assemblies) in the I/O extender.

Exchange Assemblies

Most defective I/O extender assemblies (for example, PCA card, power supply, etc.) can be exchanged for an operative assembly. For the cost and other details of the exchange program, contact your nearest HP Sales and Service Office.

Replaceable Parts

Tables 9-1 through 9-4 list the replaceable parts of the HP 12025A/B I/O Extender. Tables 9-1 and 9-2 include a figure index number for replaceable parts that are shown in the exploded view diagrams. Parts designated "OBD" (order by description) are commercially obtainable. The "MFR CODE" identifies the manufacturer of a part (see Table 9-5).

Ordering Information

To order replaceable parts, address the order to your nearest Hewlett-Packard Sales and Service Office. The following information should be included in the order for each replaceable part.

a. Complete model number and serial number.

b. Hewlett-Packard part number for each part.

c. Complete description of each part.

List of Abbreviations

Table 9-6 lists the abbreviations used in the tables of replaceable parts.

Illustrated Parts

Figures 9-1 and 9-2 show the location of the replaceable parts on the IOC and EXT printed circuit cards. Other replaceable parts are shown in the exploded view diagrams, Figures 9-3 through 9-6.

Table 9-1. Replaceable Parts (HP 12025A Only)

FIG. & INDEX#	HP PART NO.	DESCRIPTION	MFR. CODE	MFR. PART NO.
9-3-1	02430-40001	FRONT COVER	28480	02430-40001
2	5180-4242	NAMEPLATE	28480	5180-4242
3	0950-1646	POWER SUPPLY - 300W	28480	0950-1646
4	2360-0117	SCREW 6.32 x 0.375L	00000	OBD
5	12025-60003	ICJ/12 I/O JUMPER CARD	28480	12025-60003
6	02430-00012	PANEL - REAR, LEFT	28480	02430-00012
7	02430-00021	PANEL - REAR, RIGHT	28480	02430-00021
8	2360-0192	SCREW, 6-32 X 0.250L FH	00000	OBD
9	2360-0113	SCREW, 6-32 X 0.250 PH	00000	OBD
10	2360-0429	SCREW, 6-32	00000	OBD
11	02430-00028	COVER, PDU/FAN CONNECTOR	28480	02430-00028
12	2200-0103	SCREW, 4-40 X 0.250 W/LK	00000	OBD
13	02430-60005	PDU/FAN ASSEMBLY	28480	02430-60005
14	8120-1378	POWER CORD	28480	8120-1378
	02430-60015	BACKPLANE (not shown)	28480	02430-60015
	3150-0459	AIR FILTER (not shown)	28480	3150-0459
	12025-60007	CABLE ASSEMBLY (not shown)	28480	12025-60007
9-4-1	02430-00008	FAN PANEL	28480	02430-00008
2	3160-0420	FAN - 4.125 INCH DIAMETER	28480	3160-0420
3	3103-0113	THERMAL SWITCH (S3 & S4)	28480	3103-0113
4	3101-0402	SWITCH, RKR.DPST (S1 & S2)	86845	3101-0402
5	02430-00009	SWITCH PLATE	28480	02430-00009
6	2360-0194	SCREW #6-32 X 0.312L FH	00000	OBD
7	2360-0196	SCREW #6-32 X 0.375L FH	00000	OBD
8	02430-60006	CABLE, AC POWER	28480	02430-60006
9	2260-0002	NUT, #4	00000	OBD
10	0590-1516	CLIP	00000	OBD
11	1400-0249	CABLE TIE	00000	OBD
12	2190-0108	WASHER, LOCK #4	00000	OBD
13	3050-0229	WASHER, FLAT #4	00000	OBD
14	2200-0144	SCREW, 4-40 X 0.375 FH	00000	OBD
15	02430-60007	CABLE, AC SW TO POWER SUPPLY	28480	02430-60007
16	02430-60008	CABLE, PCA TO POWER SUPPLY	28480	02430-60008
17	2420-0001	NUT, 6-32 W/LOCK	00000	OBD
18	2190-0008	WASHER, LOCK EXT #6	00000	OBD
19	02430-80010	FAN RESISTOR PCA	28480	02430-80010
20	02430-60011	CABLE ASSY BATT.BKUP - SW.	28480	02430-60011
21	8120-4272	FAN CABLE - LONG	28480	8120-4272
22	0811-3643	RESISTOR 200 OHM, 1W (R1 & R2)	28480	0811-3643
23	2200-0145	SCREW, 4-40 X 0.438	00000	OBD
24	8120-4271	FAN CABLE - SHORT	28480	8120-4271
25	02430-00027	SWITCH BRACKET	28480	02430-00027
26	02430-00028	COVER, PDU/FAN CONNECTOR	28480	02430-00028
27	0400-0155	STRAIN RELIEF GROMMET	00000	OBD

Table 9-2. Replaceable Parts (HP 12025B Only)

FIG. & INDEX#	HP PART NO.	DESCRIPTION	MFR. CODE	MFR. PART NO.
9-5-1	12210-00025	SUPPORT, FRNT PNL (not shown)	28480	12210-00025
2	12151-00032	RACK EAR	28480	12151-00032
3	12210-00020	BATTERY SUPPORT	28480	12210-00020
4	12151-00034	BATTERY TRAY	28480	12151-00034
5	12210-00009	FRONT DEFLECTOR	28480	12210-00009
6	12210-00010	FAN PANEL	28480	12210-00010
7	12210-00022	REAR SHIELD	28480	12210-00022
8	12210-00013	GUIDE MOUNTING BRACKET	28480	12210-00013
9	12151-00039	RIGHT INNER RACK EAR	28480	12151-00039
10	12151-00038	LEFT INNER RACK EAR	28480	12151-00038
11	12151-60009	FAN CABLE	28480	12151-60009
12	12151-60006	LINE FILTER CABLE	28480	12151-60006
13	12151-60007	AC POWER CABLE	28480	12151-60007
14	9135-0172	LINE FILTER, POWER	49956	10PSIOR
15	3160-0315	FAN - TBAX	23936	4606X
16	3160-0092	FAN GUARD	28875	055012
17	0403-0436	PCA GUIDE	None	115-287
18	0400-0085	GROMMET - SNAP 0.875ID	28480	0400-0085
19	0360-1263	TERMINAL - BARR BLK	28480	0360-1263
20	0361-1072	RIVET (NOT REPLACEABLE)	None	
21	3050-0228	WASHER, FLAT #6 SS	00000	OBD
22	3050-0139	WASHER, FLAT #8	00000	OBD
23	2190-0851	WASHER, LOCK #6	00000	OBD
24	2360-0318	SCREW, PH #6-32 X 1.88	00000	OBD
25	2360-0359	SCREW, PH #6-32 X 0.375	00000	OBD
26	2360-0203	SCREW, PH #6-32 X 0.625	00000	OBD
27	2510-0103	SCREW, PH #8-32 X 0.375L	00000	OBD
28	2510-0099	SCREW, PH #8-32 X 0.250L	00000	OBD
29	2510-0195	SCREW, MACH #8-32 X 0.375	28480	2510-0195
30	2360-0333	SCREW, MACH 6-32 X 6.35	28480	2360-0333
31	12210-20001	PIN, LOCATION	28480	12210-20001
32	2190-0073	WASHER, LOCK #8	00000	OBD
	0950-1671	POWER SUPPLY (not shown)	28480	0950-1671
9-6-1	12025-60004	PCA-JUMPER CARD (not shown)	28480	12025-60004
2	12151-00023	REAR DOOR, RIGHT	28480	12151-00023
3	12151-00024	REAR DOOR, LEFT	28480	12151-00024
4	12210-60001	POWER PANEL	28480	12210-60001
5	12210-00031	GROUND BRACKET	28480	12210-00031
6	12210-60002	BACKPLANE	28480	12210-60002
7	12151-40001	FRONT PANEL	28480	12151-40001
8	1390-0607	PANEL FASTENER	55566	115-SS-0
9	3050-0228	WASHER, FLAT #6 SS	00000	OBD
10	2360-0359	SCREW, 6-32 X 0.375 W/SPLIT LK	00000	OBD
11	2510-0045	SCREW, 8-32 X 0.375L	00000	OBD
12	2360-0116	SCREW, 6-32 X 0.312 FH EXT	00000	OBD
	4208-0405	FOAM FILTER (not shown)	78112	4208-0405
	12151-60008	TEST POINT CABLE (not shown)	28480	12151-60008
	12025-00001	COVER PLATE (not shown)	28480	12025-00001

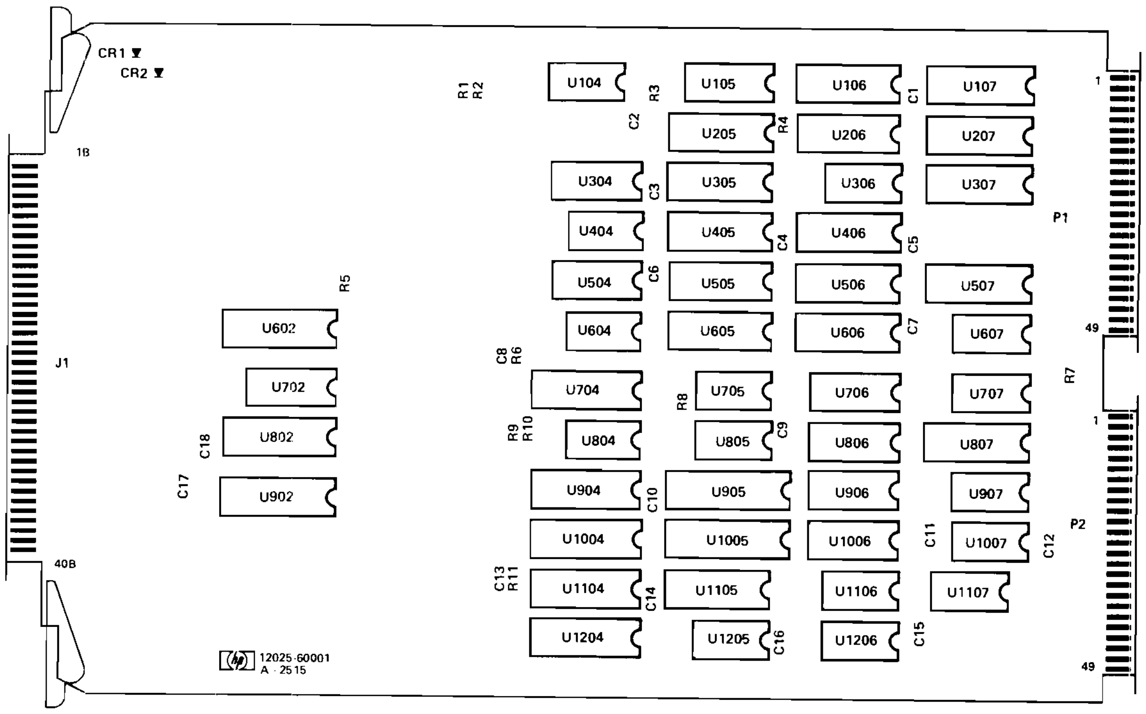


Figure 9-1. Parts Location Diagram for IOC Card (12025-60001)

Table 9-3. Replaceable Parts (IOC Card)

Reference Designation	HP Part Number	C D	Qty	Description	Mfr Code	Mfr Part Number
	12025-60001	7	1	PCA-I/O CONTROL	28480	12025-60001
C1	0160-4835	7	18	CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C2	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C3	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C4	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C5	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C6	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C7	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C8	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C9	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C10	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C11	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C12	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C13	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C14	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C15	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C16	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C17	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C18	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
CR1	1990-0486	6	1	LED-LAMP LUM-INT=1MCD IF=20MA-MAX BVR=5V	28480	5082-4684
CR2	1990-0485	5	1	LED-LAMP LUM-INT=800UCD IF=30MA-MAX	28480	5082-4984
R1	0683-3315	4	2	RESISTOR 330 5% .25W FC TC=-400/+600	01121	CB3315
R2	0683-3315	4		RESISTOR 330 5% .25W FC TC=-400/+600	01121	CB3315
R3	0683-1025	9	8	RESISTOR 1K 5% .25W FC TC=-400/+600	01121	CB1025
R4	0683-1025	9		RESISTOR 1K 5% .25W FC TC=-400/+600	01121	CB1025
R5	0683-1025	9		RESISTOR 1K 5% .25W FC TC=-400/+600	01121	CB1025
R6	0683-1025	9		RESISTOR 1K 5% .25W FC TC=-400/+600	01121	CB1025
R7	0683-1035	1	1	RESISTOR 10K 5% .25W FC TC=-400/+700	01121	CB1035
R8	0683-1025	9		RESISTOR 1K 5% .25W FC TC=-400/+600	01121	CB1025
R9	0683-1025	9		RESISTOR 1K 5% .25W FC TC=-400/+600	01121	CB1025
R10	0683-1025	9		RESISTOR 1K 5% .25W FC TC=-400/+600	01121	CB1025
R11	0683-1025	9		RESISTOR 1K 5% .25W FC TC=-400/+600	01121	CB1025
U1- U103				NOT ASSIGNED		
U104	1820-1451	8	3	IC GATE TTL S NAND QUAD 2-INP	01295	SN74S38N
U105	1820-2947	9	1	IC DIVR TTL LS BIN	28480	1820-2947
U106	1820-2701	3	4	IC FF TTL F D-TYPE POS-EDGE-TRIG COM	07263	74F374PC
U107	1820-2701	3		IC FF TTL F O-TYPE POS-EDGE-TRIG COM	07263	74F374PC
U108- U204				NOT ASSIGNED		
U205	1820-3145	1	5	IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U206	1820-3378	2	5	IC LCH TTL ALS D-TYPE NEG-EDGE-TRIG OCTL	28480	1820-3378
U207	1820-2701	3		IC FF TTL F D-TYPE POS-EDGE-TRIG COM	07263	74F374PC
U208- U303				NOT ASSIGNED		
U304	1820-1426	7	1	IC DCDR TTL LS BCD-TO-DEC 4-TO-10-LINE	01295	SN74LS145N
U305	1820-3294	1	2	IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U306	1820-2685	2	2	IC GATE TTL F NOR QUAD 2-INP	07263	74F02PC
U307	1820-2701	3		IC FF TTL F D-TYPE POS-EDGE-TRIG COM	07263	74F374PC
U308- U403				NOT ASSIGNED		
U404	1820-2506	6	2	IC INV TTL F HEX	07263	74F04PC
U405	1820-3145	1		IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U406	1820-3378	2		IC LCH TTL ALS D-TYPE NEG-EDGE-TRIG OCTL	28480	1820-3378
U407- U503				NOT ASSIGNED		
U504	1820-0629	0	1	IC FF TTL S J-K NEG-EDGE-TRIG	01295	SN74S112N
U505	1820-3294	1		IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U506	1820-3947	1	1	IC PRGMBL-LGC TTL S PLA	28480	1820-3947
U507	1820-3378	2		IC LCH TTL ALS D-TYPE NEG-EDGE-TRIG OCTL	28480	1820-3378
U508- U601				NOT ASSIGNED		

Table 9-3. Replaceable Parts (IOC Card) (Cont.)

Reference Designation	HP Part Number	C D	Qty	Description	Mfr Code	Mfr Part Number
U602	1820-3145	1		IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U603				NOT ASSIGNED		
U604	1820-2690	9	1	IC GATE TTL F OR QUAD 2-INP	07263	74F32PC
U605	1820-3378	2		IC LCH TTL ALS D-TYPE NEG-EDGE-TRIG OCTL	28480	1820-3378
U606	1820-3378	2		IC LCH TTL ALS D-TYPE NEG-EDGE-TRIG OCTL	28480	1820-3378
U607	1820-2691	0	2	IC FF TTL F D-TYPE POS-EDGE-TRIG	07263	74F74PC
U608- U701				NOT ASSIGNED		
U702	1810-0296	6	1	NETWORK-RES 16-DIP MULTI-VALUE	28480	1810-0296
U703				NOT ASSIGNED		
U704	1820-1633	8	5	IC BFR TTL S INV OCTL 1-INP	01295	SN74S240N
U705	1820-2684	1	1	IC GATE TTL F NAND QUAD 2-INP	07263	74F00PC
U706	1820-2769	3	1	IC MUXR/DATA-SEL TTL F 4-TO-1-LINE DUAL	07263	74F153PC
U707	1820-2687	4	1	IC GATE TTL F NAND TPL 3-INP	07263	74F10PC
U708- U801				NOT ASSIGNED		
U802	1820-3145	1		IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U803				NOT ASSIGNED		
U804	1820-2691	0		IC FF TTL F D-TYPE POS-EDGE-TRIG	07263	74F74PC
U805	1820-2686	3	1	IC GATE TTL F AND QUAD 2-INP	07263	74F08PC
U806	1820-3280	5	1	IC FF TTL F D-TYPE POS-EDGE-TRIG COM CLK	28480	1820-3280
U807	1820-1633	8		IC BFR TTL S INV OCTL 1-INP	01295	SN74S240N
U808- U901				NOT ASSIGNED		
U902	1820-3145	1		IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U903				NOT ASSIGNED		
U904	1820-1633	8		IC BFR TTL S INV OCTL 1-INP	01295	SN74S240N
U905	12025-80008	6	1	IC-IOCP1	28480	12025-80008
U906	1820-2696	5	2	IC FF TTL F D-TYPE POS-EDGE-TRIG COM CLK	07263	74F175PC
U907	1820-1451	8		IC GATE TTL S NAND QUAD 2-INP	01295	SN74S38N
U908- U1003				NOT ASSIGNED		
U1004	12025-80009	7	1	IC-IOCP2	28480	12025-80009
U1005	12025-80011	1	1	IC-IOCP4	28480	12025-80011
U1006	1820-2696	5		IC FF TTL F D-TYPE POS-EDGE-TRIG COM CLK	07263	74F175PC
U1007	1820-2506	6		IC INV TTL F HEX	07263	74F04PC
U1008- U1103				NOT ASSIGNED		
U1104	1820-1633	8		IC BFR TTL S INV OCTL 1-INP	01295	SN74S240N
U1105	12025-80010	0	1	IC-IOCP3	28480	12025-80010
U1106	1820-2685	2		IC GATE TTL F NOR QUAD 2-INP	07263	74F02PC
U1107	1820-1451	8		IC GATE TTL S NAND QUAD 2-INP	01295	SN74S38N
U1108- U1203				NOT ASSIGNED		
U1204	1820-1633	8		IC BFR TTL S INV OCTL 1-INP	01295	SN74S240N
U1205	1820-2688	5	1	IC GATE TTL F AND TPL 3-INP	07263	74F11PC
U1206	1820-1158	2	1	IC GATE TTL S AND-OR-INV DUAL 2-INP	01295	SN74S51N
	0403-0289	3	2	EXTR-PC BD RED POLYC .063-BD-THKNS	28480	0403-0289
	1480-0116	8	2	PIN-GRV .062-IN-DIA .25-IN-LG STL	28480	1480-0116

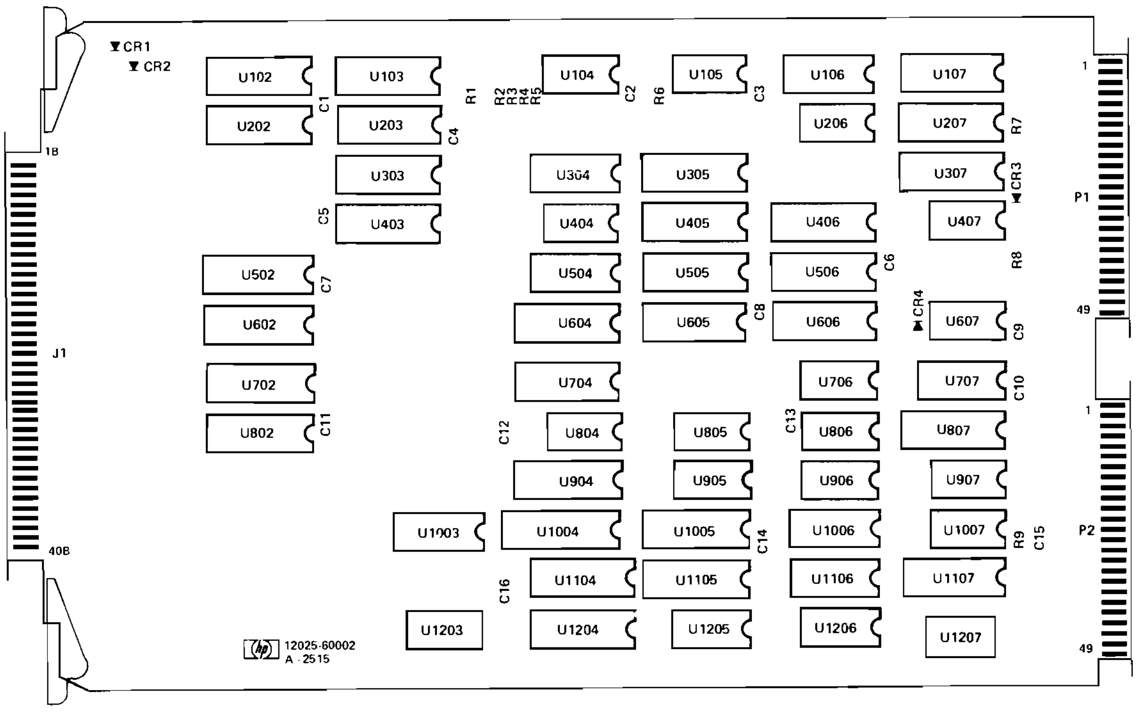


Figure 9-2. Parts Location Diagram for EXT Card (12025-60002)

Table 9-4. Replaceable Parts (EXT Card)

Reference Designation	HP Part Number	C D	Qty	Description	Mfr Code	Mfr Part Number
	12025-60002	8	1	PCA-EXT CONTROL	28480	12025-60002
C1	0160-4835	7	16	CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C2	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C3	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C4	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C5	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C6	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C7	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C8	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C9	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C10	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C11	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C12	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C13	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C14	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C15	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
C16	0160-4835	7		CAPACITOR-FXD .1UF +-10% 50VDC CER	28480	0160-4835
CR1	1990-0486	6	1	LED-LAMP LUM-INT=1MCD IF=20MA-MAX BVR=5V	28480	5082-4684
CR2	1990-0485	5	1	LED-LAMP LUM-INT=800UCD IF=30MA-MAX	28480	5082-4984
CR3	1901-0460	9	2	DIODE-STABISTOR 30V 150MA DO-7	28480	1901-0460
CR4	1901-0460	9		DIODE-STABISTOR 30V 150MA DO-7	28480	1901-0460
R1	0683-1025	9	3	RESISTOR 1K 5% .25W FC TC=-400/+600	01121	CB1025
R2	0683-3315	4	2	RESISTOR 330 5% .25W FC TC=-400/+600	01121	CB3315
R3	0683-3315	4		RESISTOR 330 5% .25W FC TC=-400/+600	01121	CB3315
R4	0683-1025	9		RESISTOR 1K 5% .25W FC TC=-400/+600	01121	CB1025
R5	0683-1035	1	1	RESISTOR 10K 5% .25W FC TC=-400/+700	01121	CB1035
R6	0683-1025	9		RESISTOR 1K 5% .25W FC TC=-400/+600	01121	CB1025
R7	0698-3447	4	1	RESISTOR 422 1% .125W F TC=0+-100	24546	C4-1/8-T0-422R-F
R8	0683-2215	1	2	RESISTOR 220 5% .25W FC TC=-400/+600	01121	CB2215
R9	0683-2215	1		RESISTOR 220 5% .25W FC TC=-400/+600	01121	CB2215
U1- U101				NOT ASSIGNED		
U102	1820-3294	1	10	IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U103	1820-3294	1		IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U104	1820-1451	8	2	IC GATE TTL S NAND QUAD 2-INP	01295	SN74S38N
U105	1820-2685	2	3	IC GATE TTL F NOR QUAD 2-INP	07263	74F02PC
U106	1820-2696	5	2	IC FF TTL F D-TYPE POS-EDGE-TRIG COM CLK	07263	74F175PC
U107	1820-3294	1		IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U108- U201				NOT ASSIGNED		
U202	1820-3145	1	7	IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U203	1820-3145	1		IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U204				NOT ASSIGNED		
U206	1820-2685	2		IC GATE TTL F NOR QUAD 2-INP	07263	74F02PC
U207	1820-3294	1		IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U208- U302				NOT ASSIGNED		
U303	1820-3294	1		IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U304	1820-0629	0	1	IC FF TTL S J-K NEG-EDGE-TRIG	01295	SN74S112N
U305	1820-3378	2	3	IC LCH TTL ALS D-TYPE NEG-EDGE-TRIG OCTL	28480	1820-3378
U306				NOT ASSIGNED		
U307	1820-3294	1		IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U308- U402				NOT ASSIGNED		
U403	1820-3145	1		IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U404	1820-2684	1	3	IC GATE TTL F NAND QUAD 2-INP	07263	74F00PC
U405	1820-3378	2		IC LCH TTL ALS D-TYPE NEG-EDGE-TRIG OCTL	28480	1820-3378
U406	1820-3378	2		IC LCH TTL ALS D-TYPE NEG-EDGE-TRIG OCTL	28480	1820-3378
U407	1858-0053	3	1	TRANSISTOR ARRAY 14-PIN PLSTC DIP	28480	1858-0053
U408- U501				NOT ASSIGNED		

Table 9-4. Replaceable Parts (EXT Card) (Cont.)

Reference Designation	HP Part Number	C D	Qty	Description	Mfr Code	Mfr Part Number
U502	1820-3294	1		IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U503				NOT ASSIGNED		
U504	1820-2696	5		IC FF TTL F D-TYPE POS-EDGE-TRIG COM CLK	07263	74F175PC
U505	1820-3145	1		IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U506	1820-3145	1		IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U507-U601				NOT ASSIGNED		
U602	1820-3145	1		IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U603				NOT ASSIGNED		
U604	1820-1633	8	5	IC BFR TTL S INV OCTL 1-INP	01295	SN74S240N
U605	1820-3294	1		IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U606	1820-3294	1		IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U607	1820-2506	6	2	IC INV TTL F HEX	07263	74F04PC
U608-U701				NOT ASSIGNED		
U702	1820-3294	1		IC FF TTL ALS D-TYPE POS-EDGE-TRIG COM	28480	1820-3294
U703				NOT ASSIGNED		
U704	1820-1633	8		IC BFR TTL S INV OCTL 1-INP	01295	SN74S240N
U706	1820-2691	0	2	IC FF TTL F D-TYPE POS-EDGE-TRIG	07263	74F74PC
U707	1810-0256	8	1	NETWORK-RES 16-DIP1.0K OHM X 15	01121	316A102
U708-U801				NOT ASSIGNED		
U802	1820-3145	1		IC DRVR TTL ALS BUS OCTL	28480	1820-3145
U803				NOT ASSIGNED		
U804	1820-2686	3	1	IC GATE TTL F AND QUAD 2-INP	07263	74F08PC
U805	1820-2691	0		IC FF TTL F D-TYPE POS-EDGE-TRIG	07263	74F74PC
U806	1820-2684	1		IC GATE TTL F NAND QUAD 2-INP	07263	74F00PC
U807	1820-1633	8		IC BFR TTL S INV OCTL 1-INP	01295	SN74S240N
U808-U903				NOT ASSIGNED		
U904	1820-1633	8		IC BFR TTL S INV OCTL 1-INP	01295	SN74S240N
U905	1820-2684	1		IC GATE TTL F NAND QUAD 2-INP	07263	74F00PC
U906	1820-1450	7	1	IC BFR TTL S NAND QUAD 2-INP	01295	SN74S37N
U907	1820-1451	8		IC GATE TTL S NAND QUAD 2-INP	01295	SN74S38N
U908-U1002				NOT ASSIGNED		
U1003	1810-0296	6	1	NETWORK-RES 16-DIP MULTI-VALUE	28480	1810-0296
U1004	12025-80012	2	1	IC-EXTP1	28480	12025-80012
U1005	12025-80013	3	1	IC-EXTP2	28480	12025-80013
U1006	1820-2693	2	1	IC FF TTL F J-K BAR POS-EDGE-TRIG	07263	74F109PC
U1007	1810-0182	9	1	NETWORK-RES 14-DIP MULTI-VALUE	28480	1810-0182
U1008-U1103				NOT ASSIGNED		
U1104	1820-3280	5	3	IC FF TTL F D-TYPE POS-EDGE-TRIG COM CLK	28480	1820-3280
U1105	12025-80014	4	1	IC-EXTP3	28480	12025-80014
U1106	1820-3280	5		IC FF TTL F D-TYPE POS-EDGE-TRIG COM CLK	28480	1820-3280
U1107	1820-1633	8		IC BFR TTL S INV OCTL 1-INP	01295	SN74S240N
U1108-U1202				NOT ASSIGNED		
U1203	1810-0763	2	1	DELAY LINE ACTIVE DELAY LINE; STTL	28480	1810-0763
U1204	1820-3280	5		IC FF TTL F D-TYPE POS-EDGE-TRIG COM CLK	28480	1820-3280
U1205	1820-2685	2		IC GATE TTL F NOR QUAD 2-INP	07263	74F02PC
U1206	1820-2506	6		IC INV TTL F HEX	07263	74F04PC
U1207	1813-0196	1	1	XTAL-CLOCK-OSCILLATOR 14.7456-MHZ	28480	1813-0196
	0403-0289	3	2	EXTR-PC BD RED POLYC .063-BD-THKNS	28480	0403-0289
	1480-0116	8	2	PIN-GRV .062-IN-DIA .25-IN-LG STL	28480	1480-0116

Table 9-5. Code List of Manufacturers

Mfg. Code	Manufacturer Name	Address		Zip Code
01121	Allen-Bradley Co.	Milwaukee,	WI	53204
01295	Texas Instrument Inc. Semiconductor Component Div.	Dallas,	TX	75222
07263	Fairchild Semiconductor Div.	Mountain View,	CA	94042
23936	Pamotor Div. William J. Purdy	Burlingame,	CA	94005
24546	Corning Glass Works (Bradford)	Bradford,	PA	16701
28480	Hewlett-Packard Co. Corporate HQ.	Palo Alto,	PA	94304
28875	IMC Magnetic Corp.	Rochester,	NH	03867
49956	Raytheon Co.	Lexington,	MA	02173
55566	R.A.F. Electronic Hardware Inc.	Stratford,	CT	06497
78112	Scott Paper Co.	Philadelphia,	PA	19113
86845	Marquart Co.	Van Nuys,	CA	91409

Table 9-6. Abbreviations

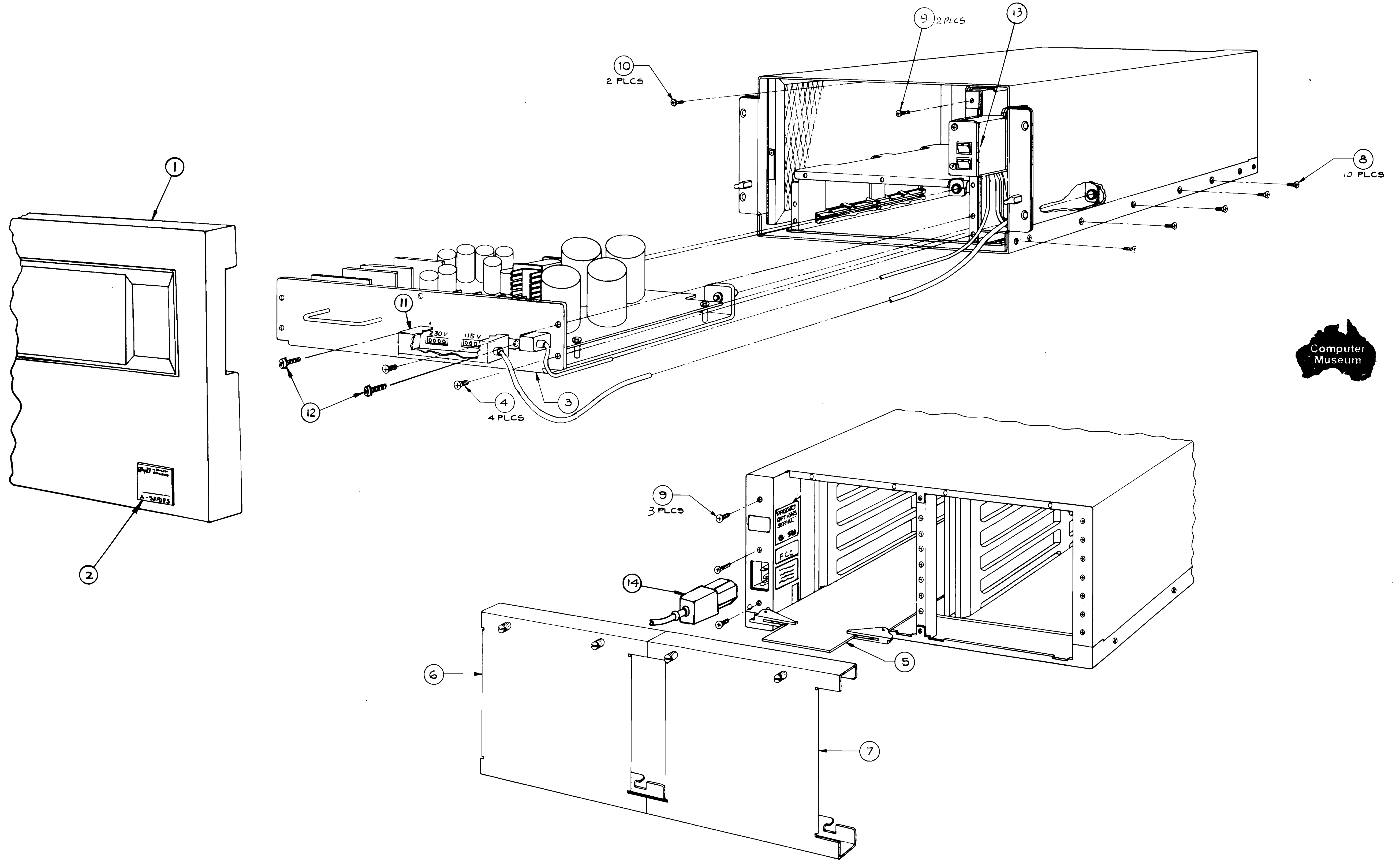
REFERENCE DESIGNATIONS

A	= assembly
B	= motor, synchro
BT	= battery
C	= capacitor
CB	= circuit breaker
CR	= diode
DL	= delay line
DS	= indicator
E	= Misc electrical parts
F	= fuse
FL	= filter
J	= receptacle connector
K	= relay
L	= inductor
M	= meter
P	= plug connector
Q	= semiconductor device other than diode or integrated circuit
R	= resistor
RT	= thermistor
S	= switch
T	= transformer
TB	= terminal board
TP	= test point
U	= integrated circuit, non-repairable assembly
V	= vacuum tube, photocell, etc.
VR	= voltage regulator
W	= jumper wire
X	= socket
Y	= crystal
Z	= tuned cavity, network

ABBREVIATIONS

A	= amperes
ac	= alternating current
Ag	= silver
Al	= aluminum
ar	= as required
adj	= adjust
assy	= assembly
b	= base
bp	= bandpass
bpi	= bits per inch
blk	= black
blu	= blue
brn	= brown
brs	= brass
Btu	= British thermal unit
Be Cu	= beryllium copper
cpi	= characters per inch
coll	= collector
cw	= clockwise
ccw	= counterclockwise
cer	= ceramic
com	= common
crt	= cathode-ray tube
CTL	= complementary-transistor logic
cath	= cathode
Cd pl	= cadmium plate
comp	= composition
conn	= connector
compl	= complete
dc	= direct current
dr	= drive
DTL	= diode-transistor logic
depc	= deposited carbon
dpdt	= double-pole, double-throw
dpst	= double-pole, single-throw
em	= emitter
ECL	= emitter-coupled logic
ext	= external
encap	= encapsulated
elctlt	= electrolytic
F	= farads
FF	= flip-flop
flh	= flat head
flm	= film
fxd	= fixed
filh	= fillister head
G	= giga (10^9)
Ge	= germanium
gl	= glass
gnd	= ground(ed)
gra	= gray
grn	= green
H	= henries
Hg	= mercury
hr	= hour(s)
Hz	= hertz
hdw	= hardware
hex	= hexagon, hexagonal
ID	= inside diameter
IF	= intermediate frequency
in.	= inch, inches
I/O	= input/output
int	= internal
incl	= include(s)
insul	= insulation, insulated
impgrg	= impregnated
incand	= incandescent
ips	= inches per second
k	= kilo (10^3), kilohm
lp	= low pass
m	= milli (10^{-3})
M	= mega (10^6), megohm
My	= Mylar
mfr	= manufacturer
mom	= momentary
mtg	= mounting
misc	= miscellaneous
met. ox.	= metal oxide
mintr	= miniature
n	= nano (10^{-9})
nc	= normally closed or no connection
Ne	= neon
no.	= number
n.o.	= normally open
np	= nickel plated
NPN	= negative-positive-negative
NPO	= negative-positive zero (zero temperature coefficient)
NSR	= not separately replaceable
NRFR	= not recommended for field replacement
OD	= outside diameter
OBD	= order by description
orn	= orange
ovh	= oval head
oxd	= oxide
p	= pico (10^{-12})
PC	= printed circuit
PCA	= printed-circuit assembly
PWB	= printed-wiring board
phh	= phillips head
pk	= peak
p-p	= peak-to-peak
pt	= point
prv	= peak inverse voltage
PNP	= positive-negative-positive
pwv	= peak working voltage
porc	= porcelain
posn	= position(s)
pozi	= pozidrive
rf	= radio frequency
rdh	= round head
rms	= root-mean-square
rwv	= reverse working voltage
rect	= rectifier
r/min	= revolutions per minute
RTL	= resistor-transistor logic
s	= second
SB, TT	= slow blow
Se	= selenium
Si	= silicon
scr	= silicon controlled rectifier
sst	= stainless steel
stl	= steel
spcl	= special
spdt	= single-pole, double-throw
spst	= single-pole, single-throw
Ta	= tantalum
td	= time delay
Ti	= titanium
tgl	= toggle
thd	= thread
tol	= tolerance
TTL	= transistor transistor logic
U(μ)	= micro (10^{-6})
V	= volt(s)
var	= variable
vio	= violet
Vdcw	= direct current working volts
W	= watts
ww	= wirewound
wht	= white
WIV	= working inverse voltage
yel	= yellow

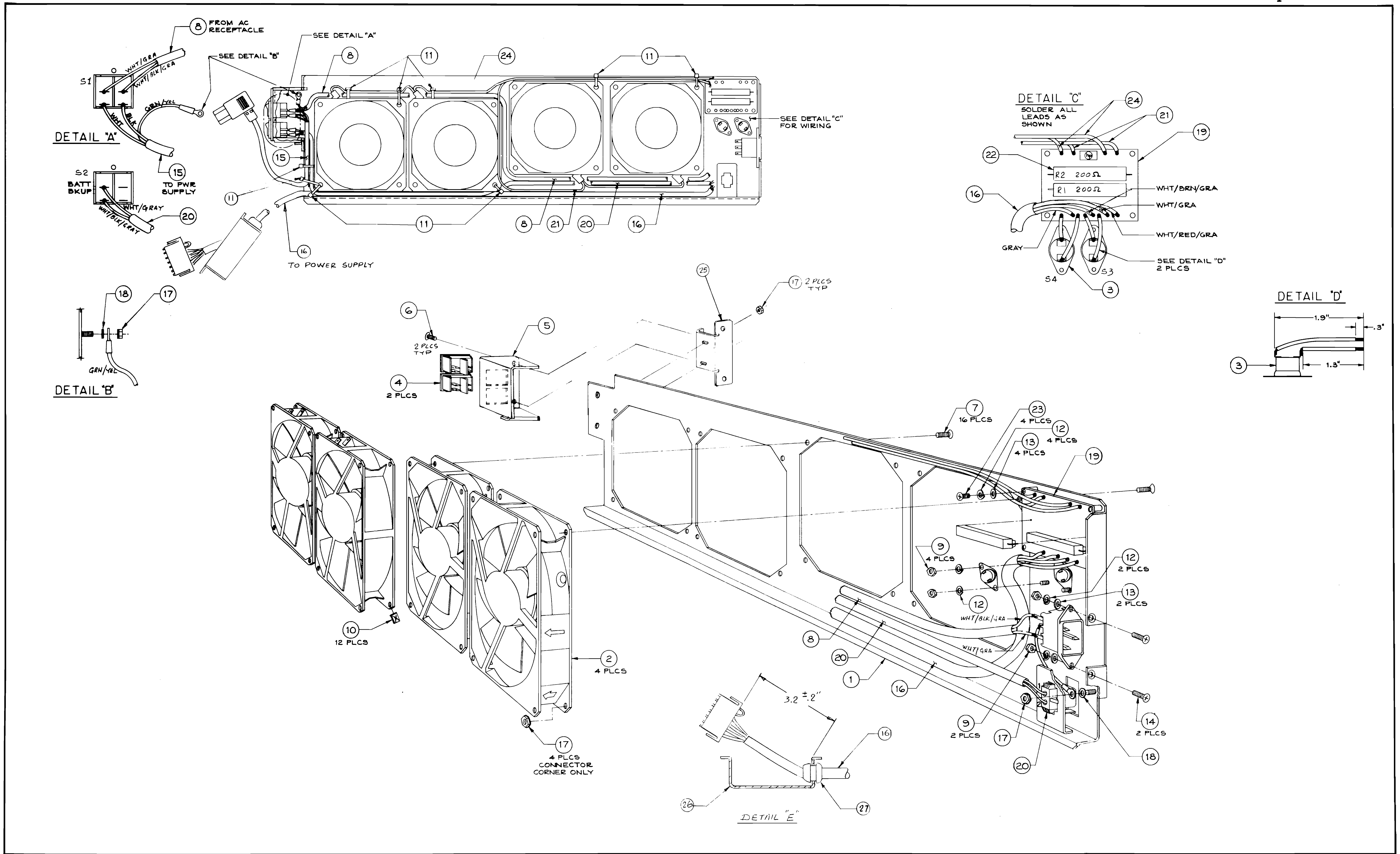
8500-15



D12025-90002-1 REV.A

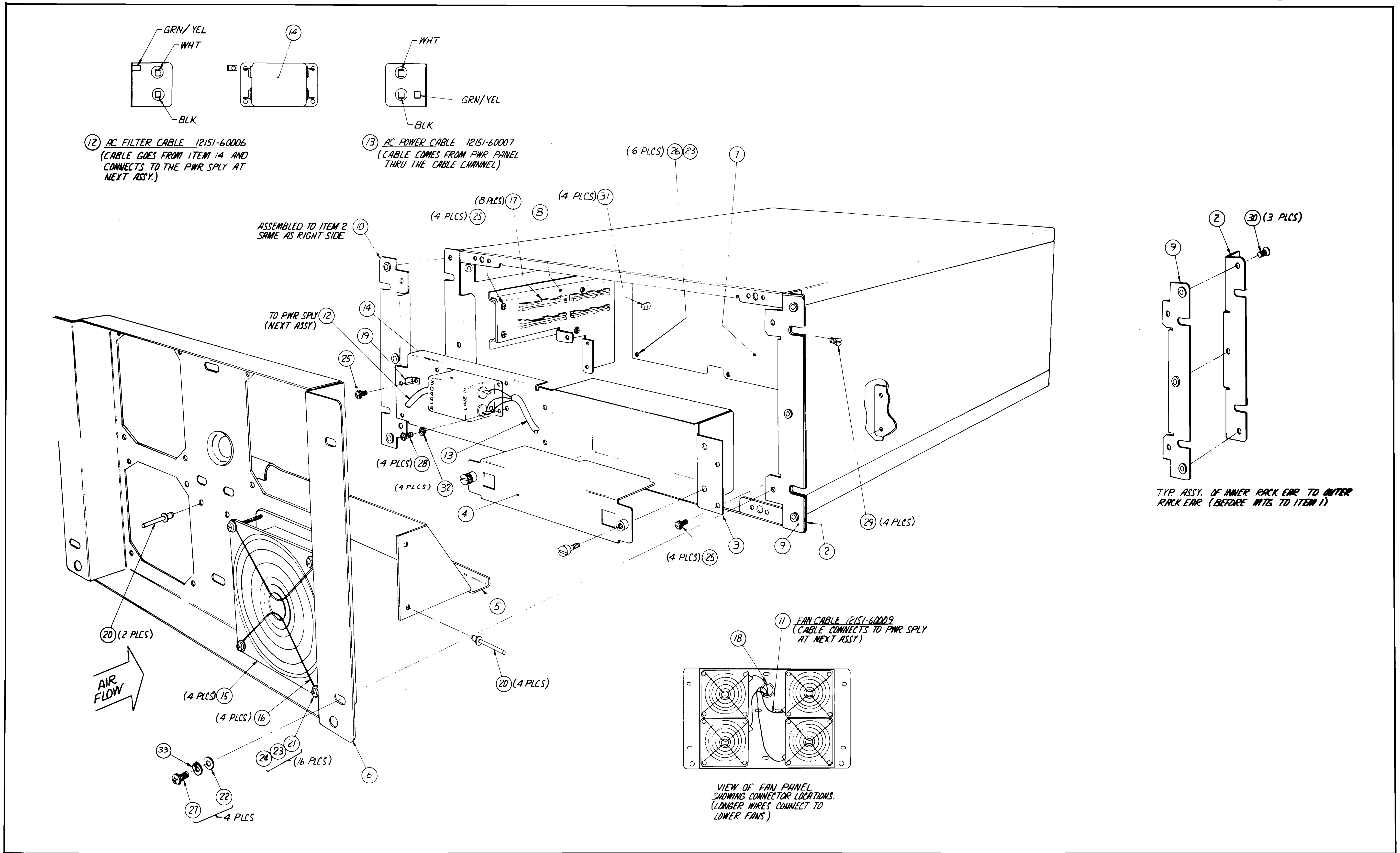
Figure 9-3. HP 12025A Exploded Views, Front And Rear 9-13/9-14

Computer Museum



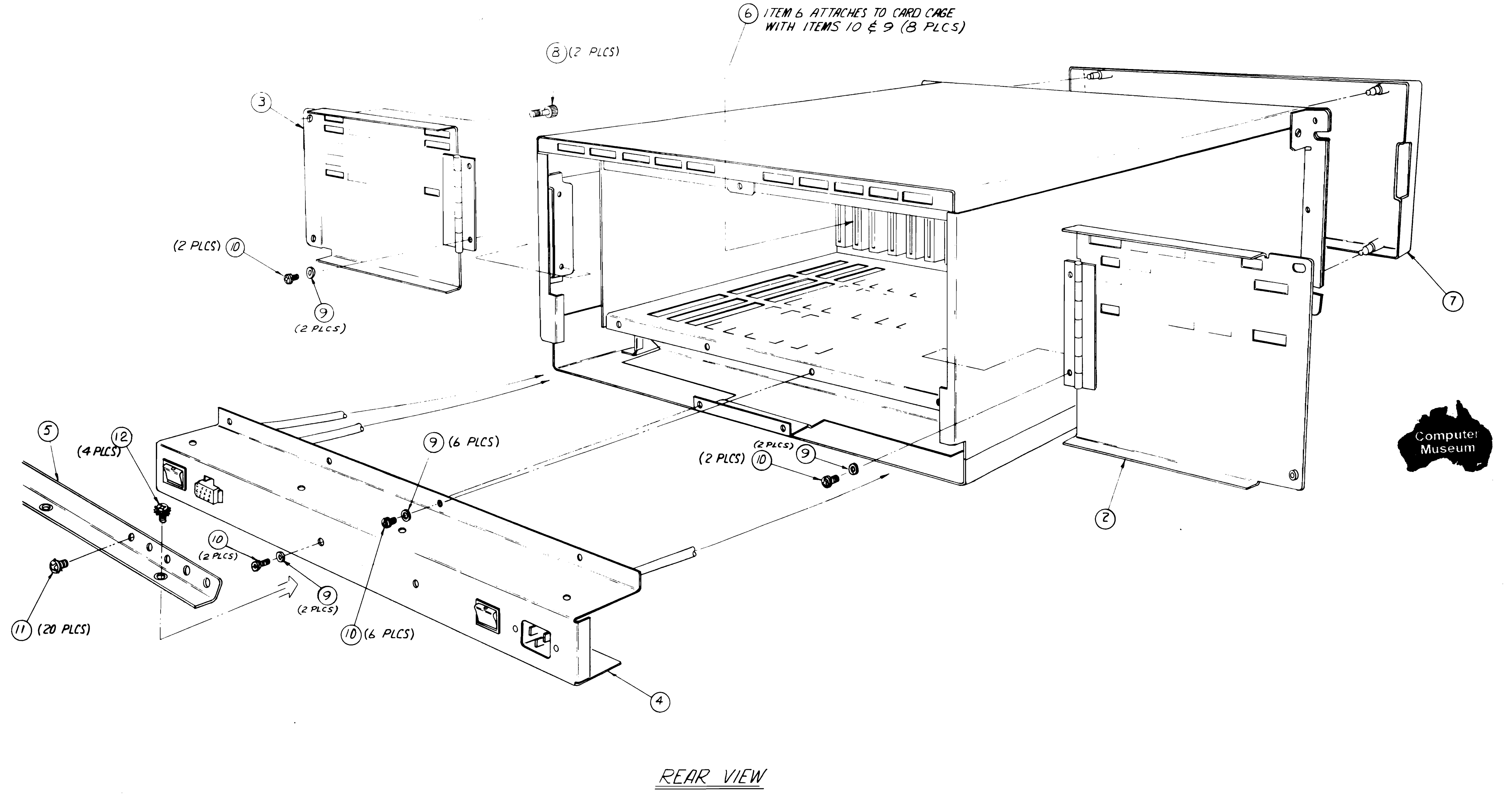
D02430-60005-1 REV.F

Figure 9-4. Power Distribution Unit Exploded View (HP 12025A)


GRN/YEL
WHT
BLK
14
WHT
GRN/YEL
BLK
12 AC FILTER CABLE 12151-60006 (CABLE GOES FROM ITEM 14 AND CONNECTS TO THE PWR SPLY AT NEXT ASSY.)
13 AC POWER CABLE 12151-60007 (CABLE COMES FROM PWR PANEL THRU THE CABLE CHANNEL)
(6 PLCS) 26 23
7
(8 PLCS) 17
(4 PLCS) 31
(4 PLCS) 25
8
ASSEMBLED TO ITEM 2 SAME AS RIGHT SIDE 10
2
30 (3 PLCS)
9
TO PWR SPLY 12 (NEXT ASSY)
14
19
25
(4 PLCS) 28
(4 PLCS) 32
13
4
(4 PLCS) 25
3
9
2
29 (4 PLCS)
TYP. ASSY. OF INNER RACK EAR TO OUTER RACK EAR (BEFORE MTG. TO ITEM 1)
5
20 (2 PLCS)
20 (4 PLCS)
11 FAN CABLE 12151-60009 (CABLE CONNECTS TO PWR SPLY AT NEXT ASSY)
18
AIR FLOW
(4 PLCS) 15
(4 PLCS) 16
24 23 21 (16 PLCS)
6
33
22
27
4 PLCS
VIEW OF FAN PANEL SHOWING CONNECTOR LOCATIONS. (LONGER WIRES CONNECT TO LOWER FANS)

D12151-60001-1 REV.E

Figure 9-5. HP 12025B Exploded View, Front



Computer Museum

D12151-60001-2 REV.E

Figure 9-6. HP 12025B Exploded View, Rear

Chapter 10
Reference

Introduction

The HP 12025A/B I/O Extender can only be used with HP A-Series computers. This chapter provides a list of A-Series computer manuals that will help you to understand the operation of the A-Series computers and the I/O extender.

Manuals and Other Reference Documents

The following manuals and documents are helpful for an in-depth understanding of the I/O extender and the A-Series computers:

a. HP 1000 A600/A600+ Computer Reference Manual, part no. 02156-90001.

b. HP 1000 A600/A600+ Computer Installation and Service Manual, part no. 02156-90002.

c. HP 1000 A700 Computer Reference Manual, part no. 02137-90001.

d. HP 1000 A700 Computer Installation and Service Manual, part no. 02137-90002.

e. HP 1000 A900 Computer Reference Manual, part no. 02139-90001.

f. HP 1000 A900 Computer Installation and Service Manual, part no. 02139-90002.

g. HP Micro/1000 System/Computer Installation and Service Manual, part no. 02430-90001.

h. HP 1000 A/L-Series Computer I/O Interfacing Guide, part no. 02103-90005.

i. A600/A600+ Computer Engineering Reference Documentation, part no. 02156-90003.

j. A700 Computer Engineering Reference Documentation, part no. 02137-90005.

k. A900 Computer Engineering Reference Documentation, part no. 02139-90003.

l. Bipolar LSI 1984 Databook, Fifth Edition, by Monolithic Memories, Inc.

m. AMD 20-Pin PAL Family Data Sheet by Advanced Micro Devices, Inc.

n. Integrated Fuse Logic Data Manual 1984 by Signetics Corporation.

Chapter 11 Product History

Since this manual is a First Edition covering new products, it does not include any product history.

Chapter 12
Diagrams

Introduction

This chapter provides timing diagrams and schematic diagrams for the IOC and EXT cards of the extender. Also included are power distribution diagrams.

NOTE

Many of the schematic diagrams and parts lists in this document contain the generic part number of IC packs. These are for your convenience in referring to pack diagrams and general operating conditions in semiconductor manufacturer's catalogs. The generic part number, however, should not be used to order replacement parts. Many parts used in Hewlett-Packard equipment are purchased with special specifications and tolerances, or may undergo special testing and treatment (such as burn-in). Replacement parts therefore must be ordered using the Hewlett-Packard part number to insure that the replacement part will restore the equipment to its proper operating condition.

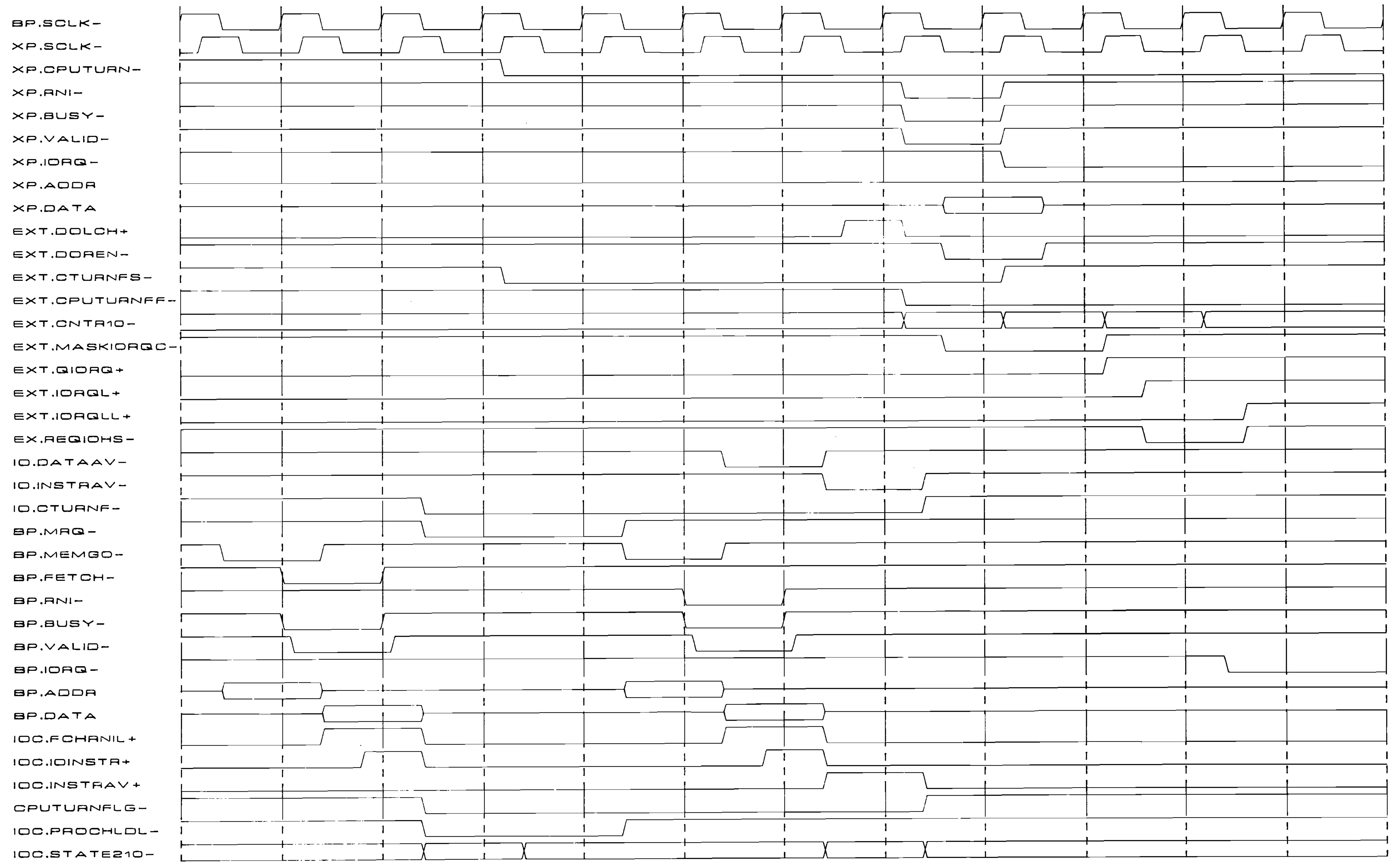


Figure 12-1. Timing Diagram For I/O Instruction Broadcasting (With A600+/A700 Host)

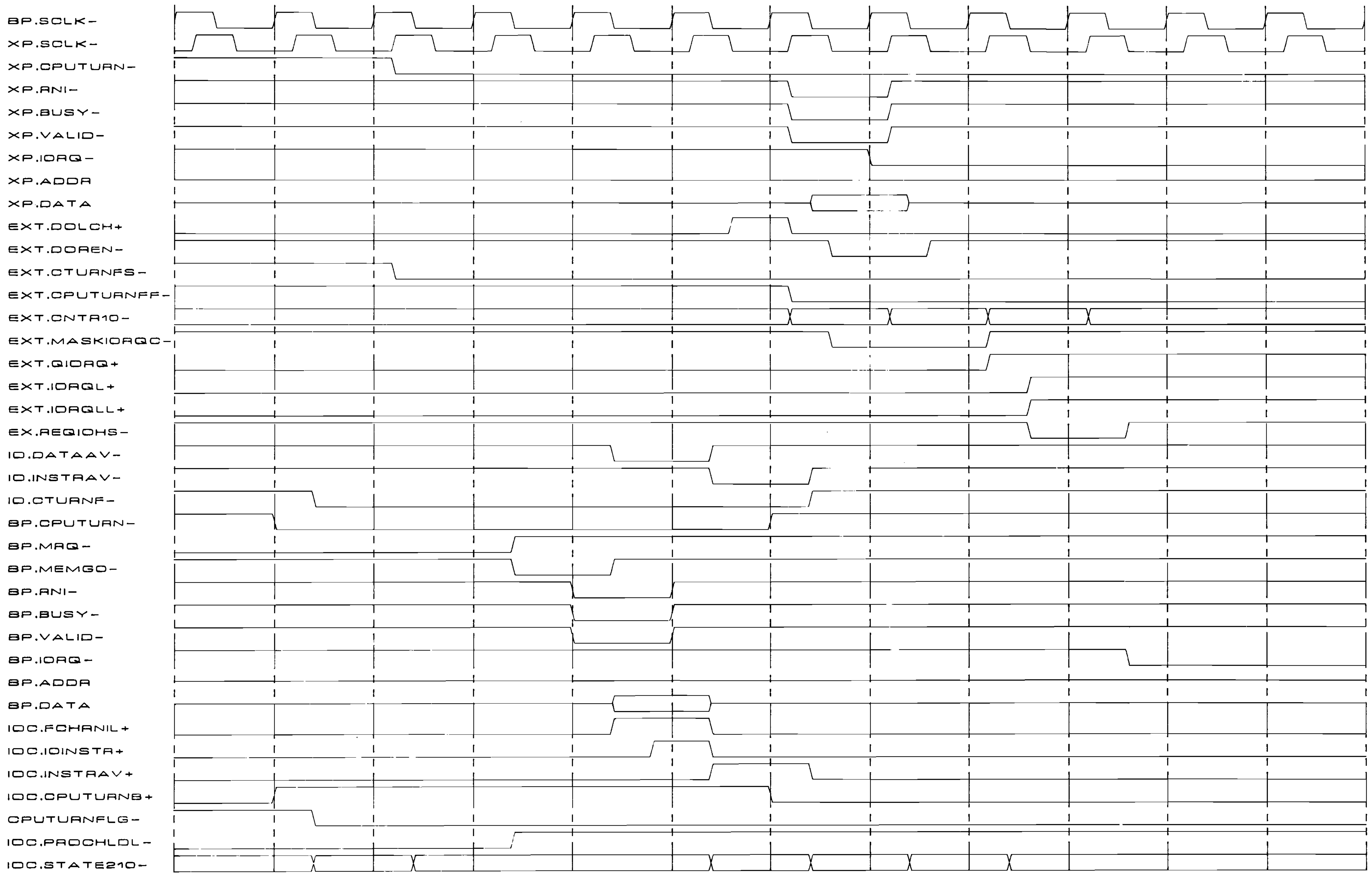


Figure 12-2. Timing Diagram For I/O Instruction Broadcasting (With A900 Host)

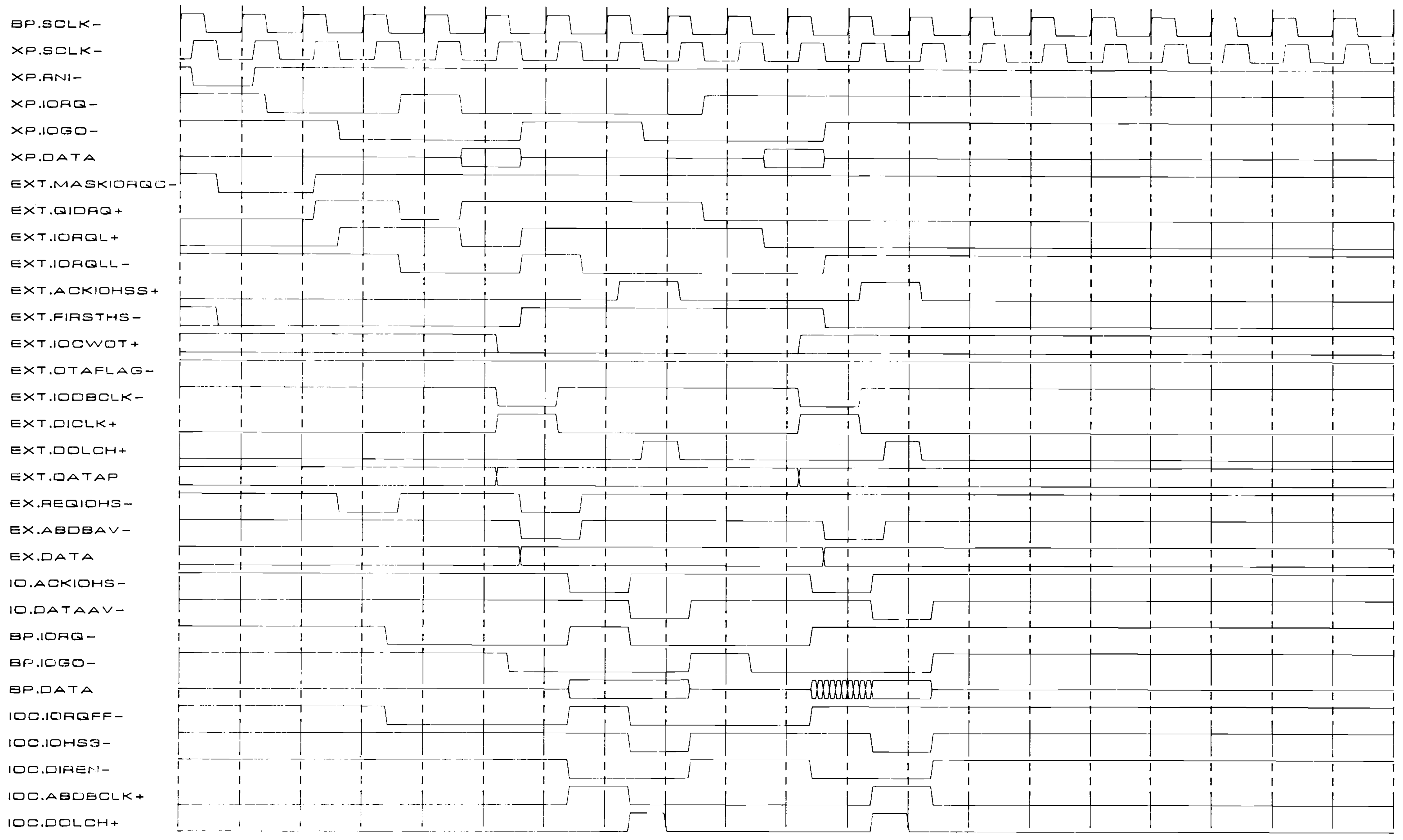


Figure 12-3. Timing Diagram For I/O Read (LI*/MI*)

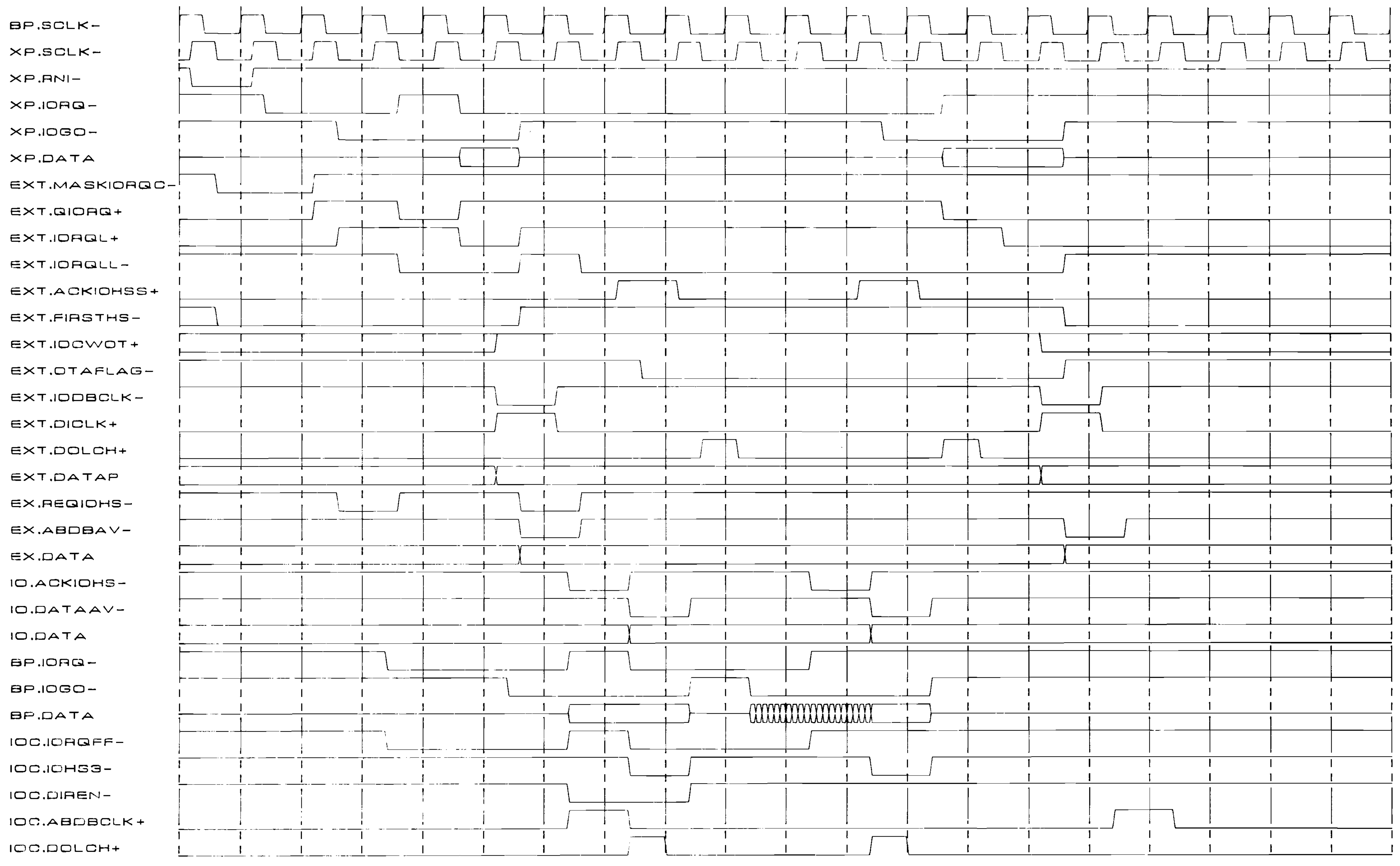


Figure 12-4. Timing Diagram For I/O Write (OT* Except SC=0/2)

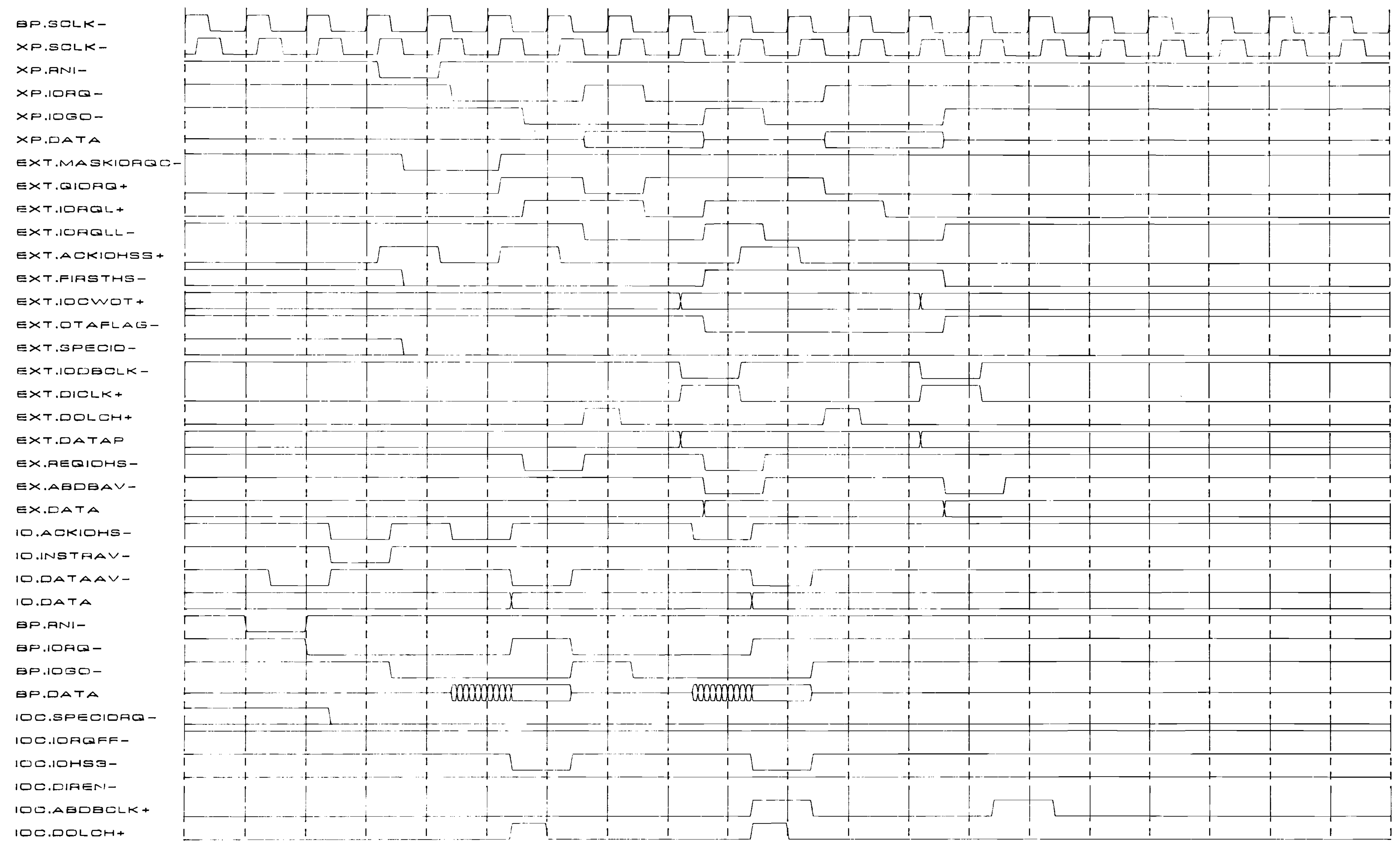


Figure 12-5. Timing Diagram For I/O
Write (OT* 0 And OT* 2)

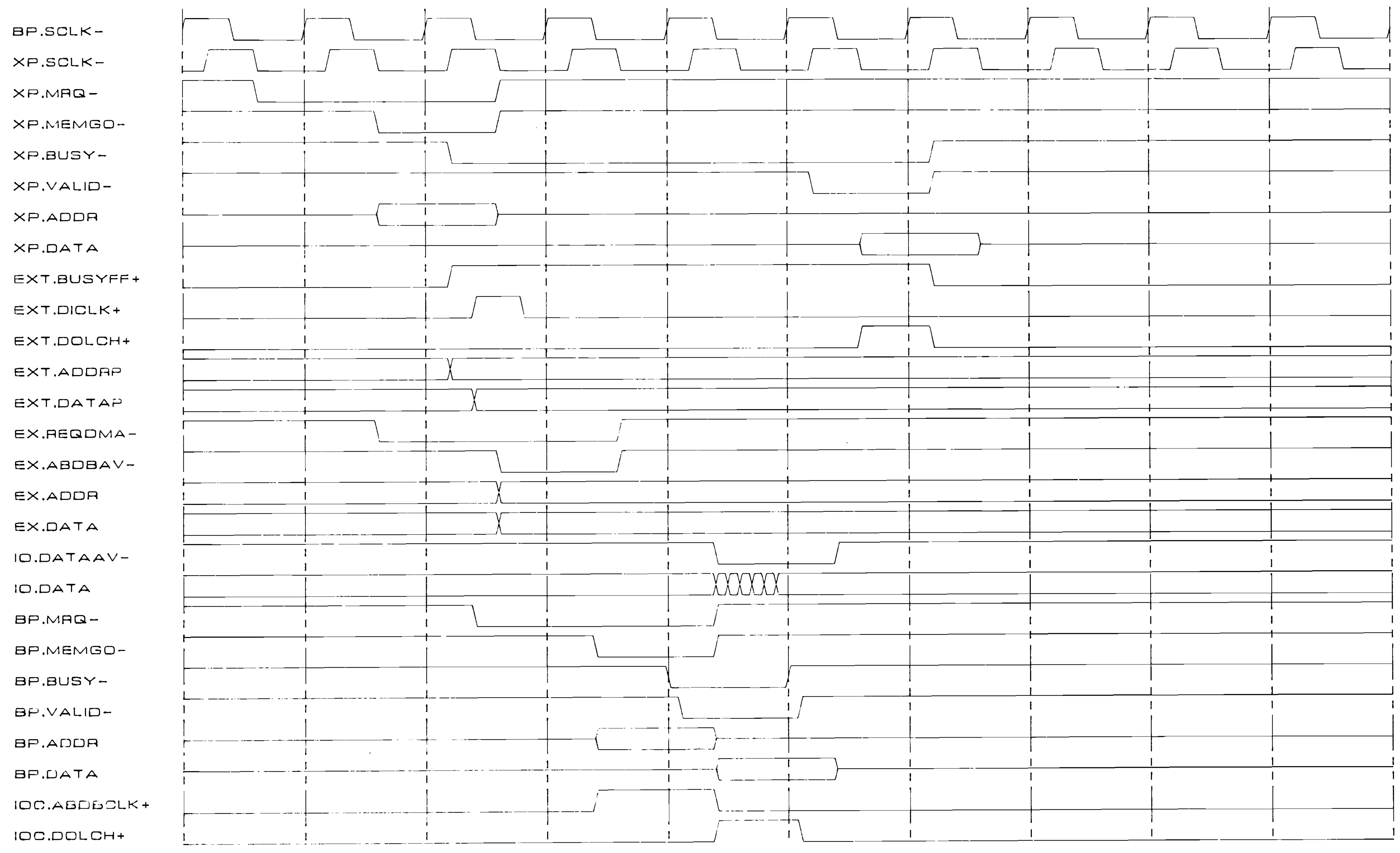


Figure 12-6. Timing Diagram For DMA Read

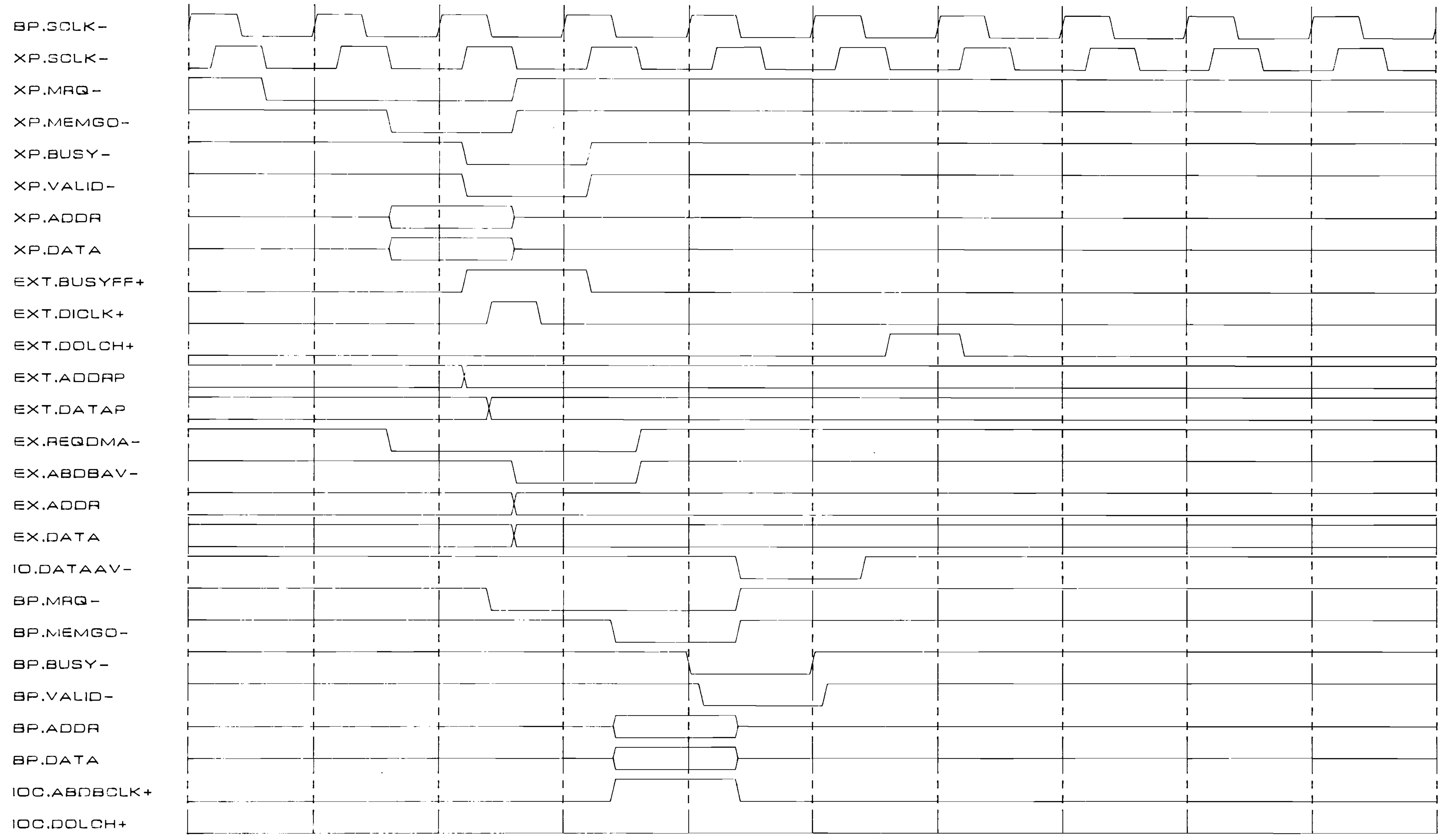


Figure 12-7. Timing Diagram For DMA Write

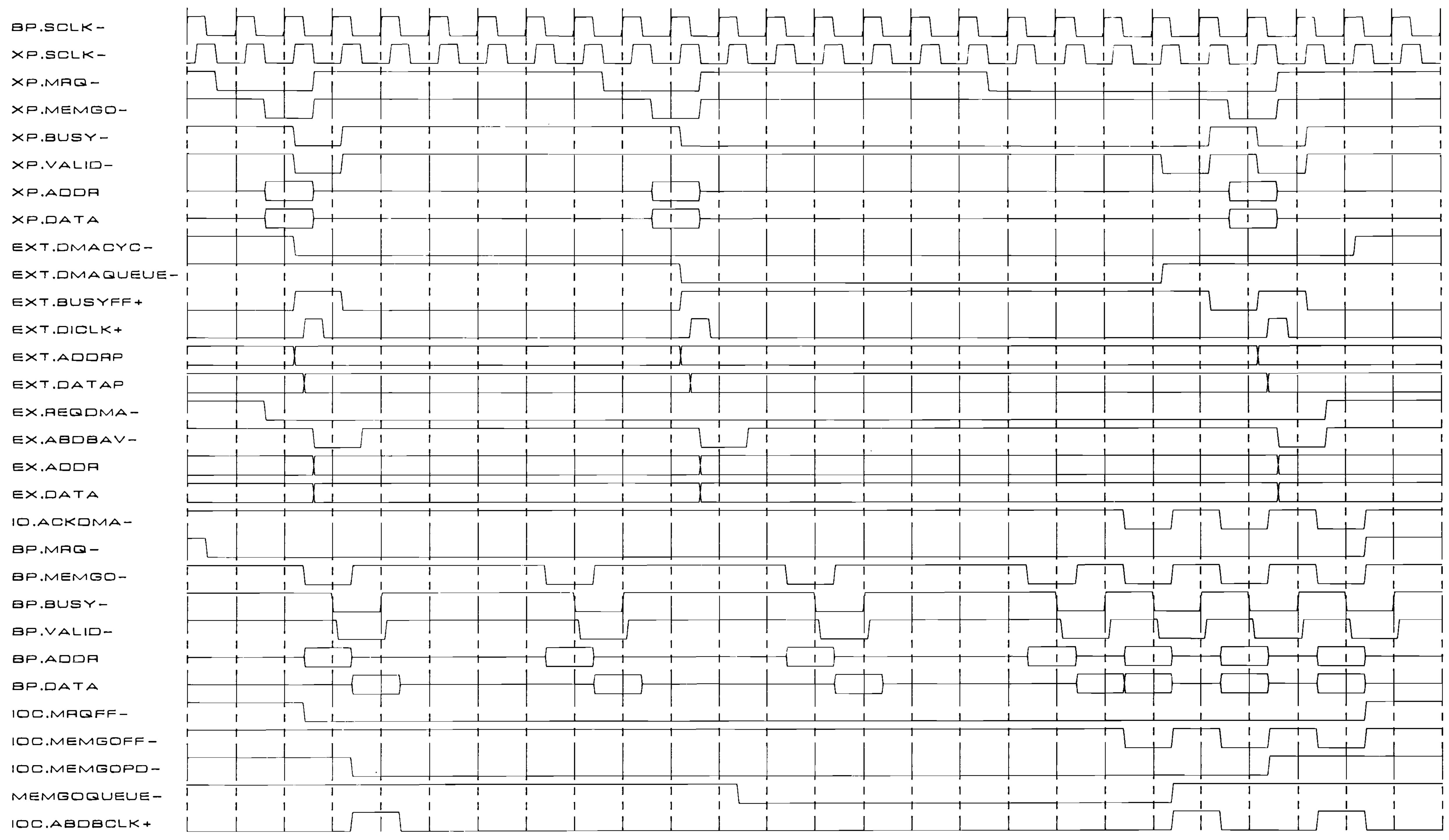


Figure 12-8. Timing Diagram For DMA Write, High-Speed DMA Queue

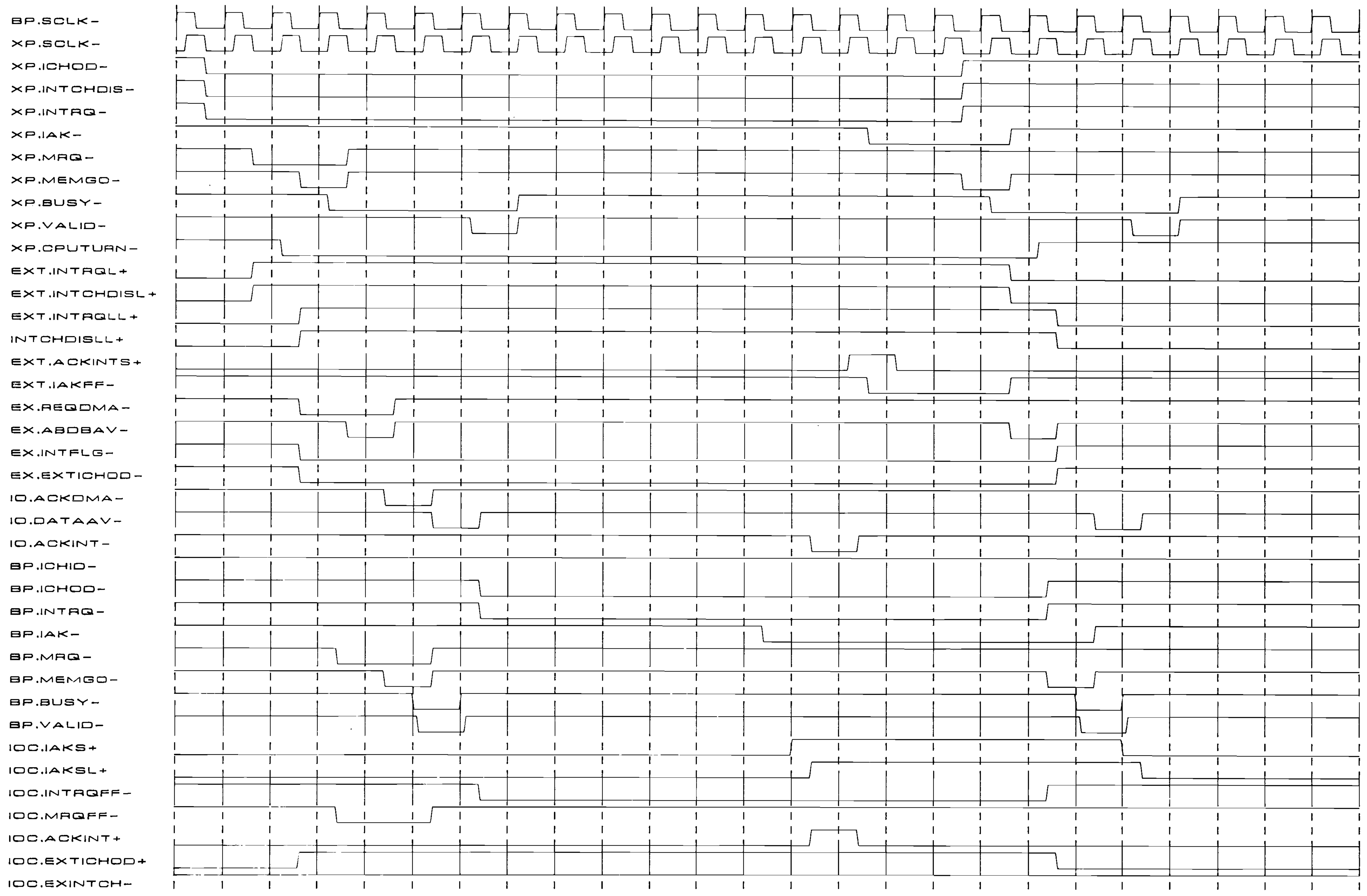


Figure 12-9. Timing Diagram For Interrupt Processing

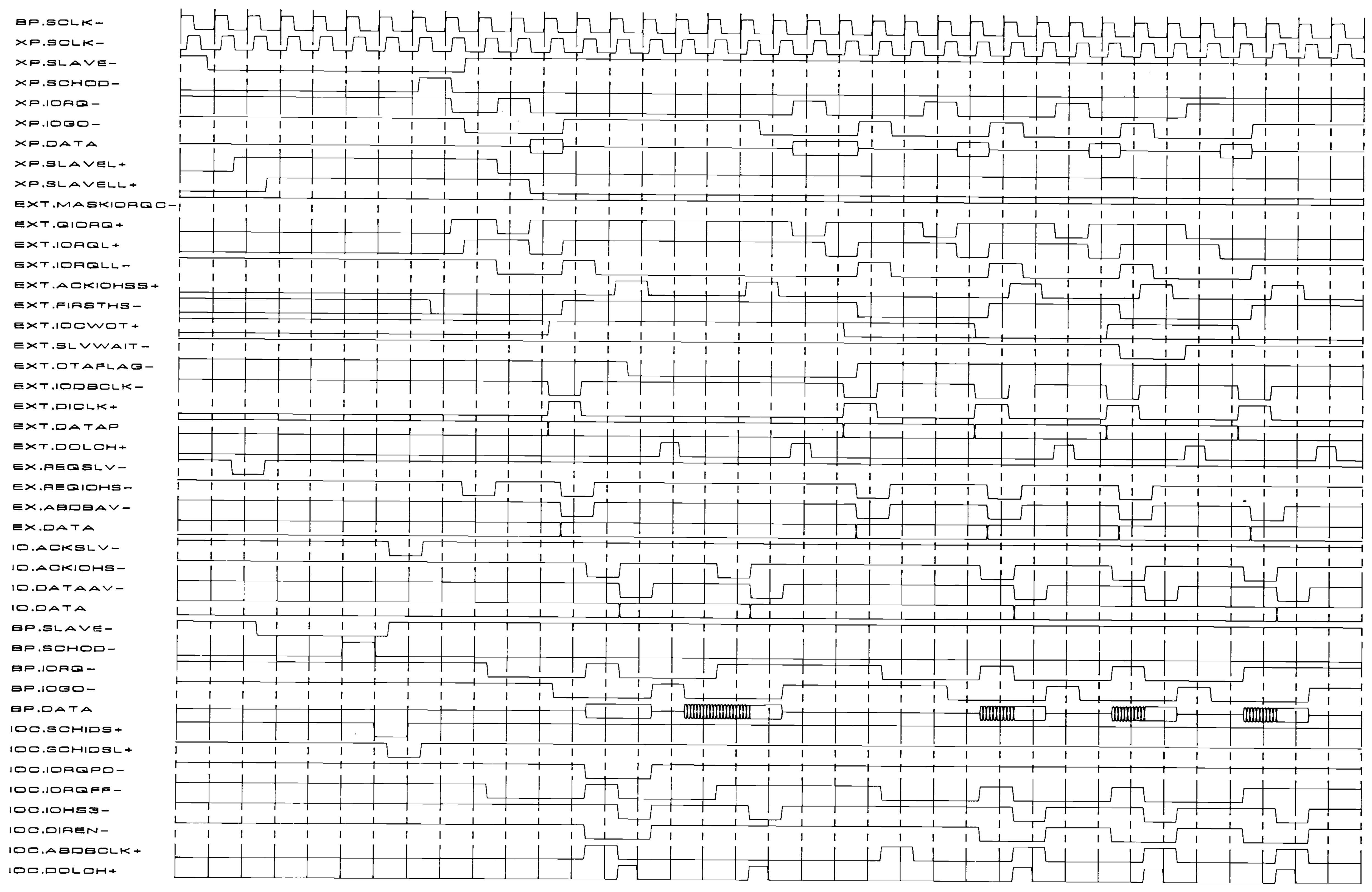


Figure 12-10. Timing Diagram For Break (Slave Mode) Processing

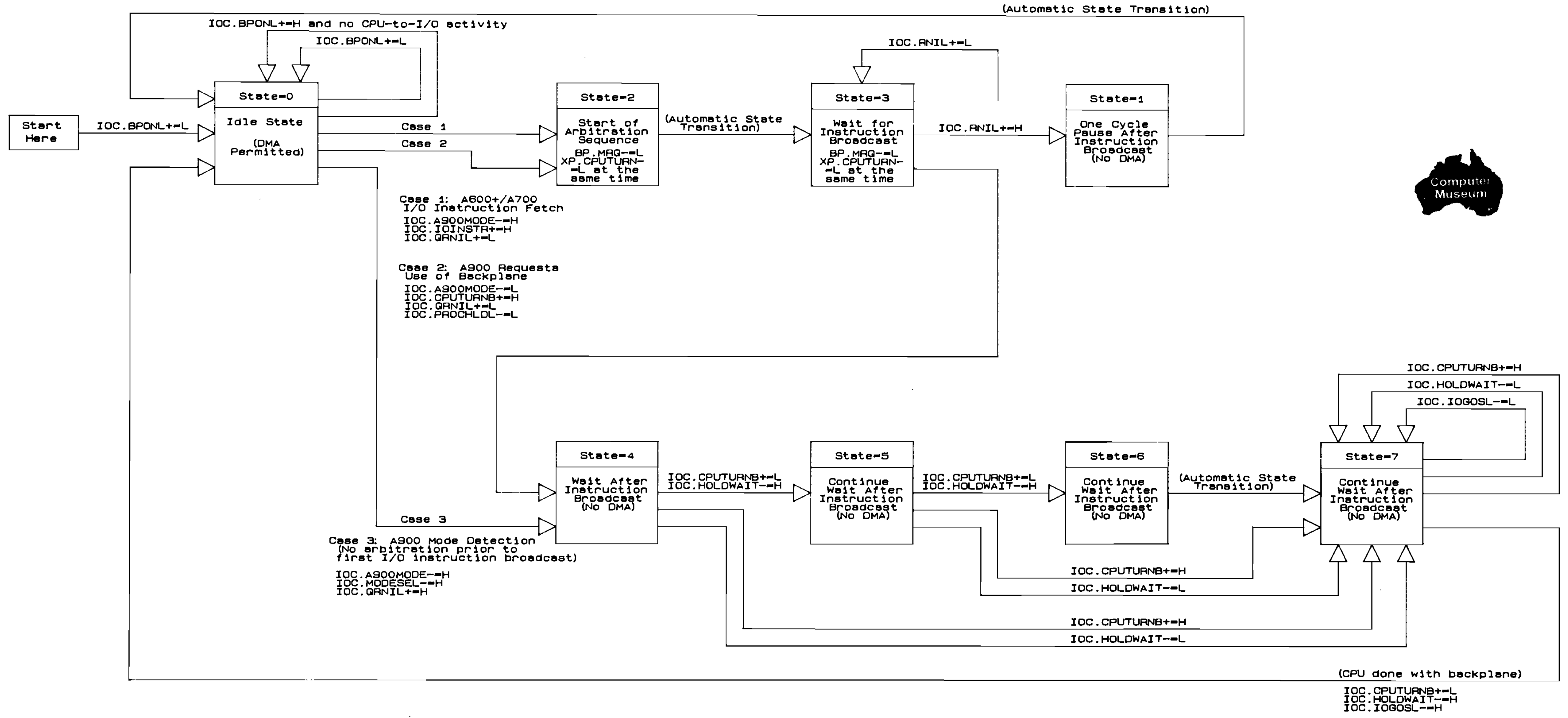
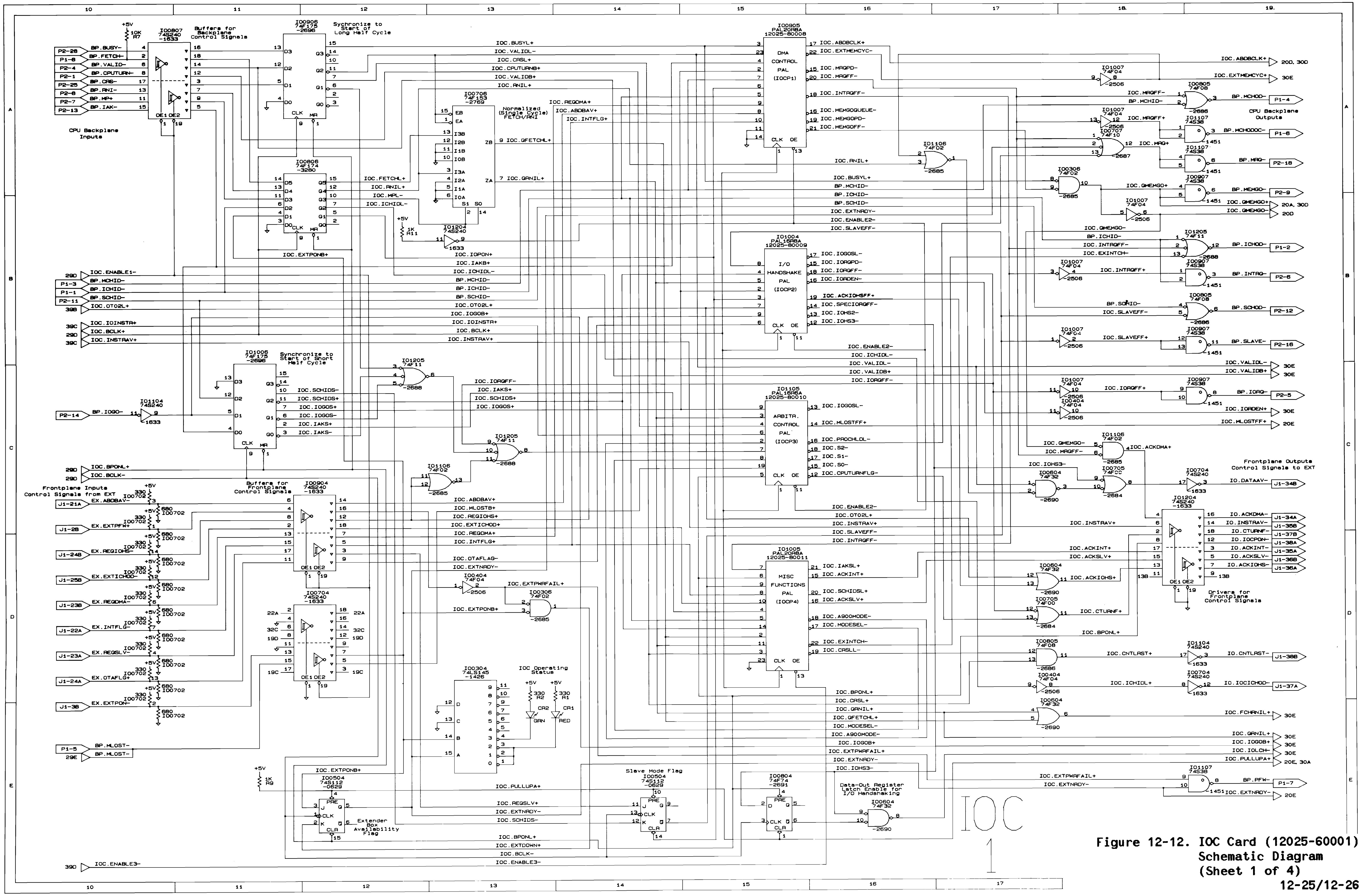

Start Here
IOC.BPONL+=L
State=0
Idle State
(DMA Permitted)
IOC.BPONL+=H and no CPU-to-I/O activity
IOC.BPONL+=L
(Automatic State Transition)
Case 1
Case 2
State=2
Start of Arbitration Sequence
BP.MRQ-=L XP.CPUTURN-=L at the same time
(Automatic State Transition)
State=3
Wait for Instruction Broadcast
BP.MRQ-=L XP.CPUTURN-=L at the same time
IOC.RNIL+=L
IOC.RNIL+=H
State=1
One Cycle Pause After Instruction Broadcast (No DMA)
Case 1: A600+/A700 I/O Instruction Fetch
IOC.A900MODE-=H
IOC.IOINSTR+=H
IOC.QRNIL+=L
Case 2: A900 Requests Use of Backplane
IOC.A900MODE-=L
IOC.CPUTURNB+=H
IOC.QRNIL+=L
IOC.PROCHLDL-=L
Case 3
Case 3: A900 Mode Detection (No arbitration prior to first I/O instruction broadcast)
IOC.A900MODE-=H
IOC.MODESEL-=H
IOC.QRNIL+=H
State=4
Wait After Instruction Broadcast (No DMA)
IOC.CPUTURNB+=L
IOC.HOLDWAIT-=H
State=5
Continue Wait After Instruction Broadcast (No DMA)
IOC.CPUTURNB+=L
IOC.HOLDWAIT-=H
State=6
Continue Wait After Instruction Broadcast (No DMA)
(Automatic State Transition)
State=7
Continue Wait After Instruction Broadcast (No DMA)
IOC.CPUTURNB+=H
IOC.HOLDWAIT-=L
IOC.IOGOSL-=L
IOC.CPUTURNB+=H
IOC.HOLDWAIT-=L
IOC.CPUTURNB+=H
IOC.HOLDWAIT-=L
(CPU done with backplane)
IOC.CPUTURNB+=L
IOC.HOLDWAIT-=H
IOC.IOGOSL-=H

Figure 12-11. State Transition Diagram



Buffers for Backplane Control Signals
CPU Backplane Inputs
Sychronize to Start of Long Half Cycle
Normalized (Single Cycle) FETCH/RNI
DMA CONTROL PAL (IOCP1)
I/O HANDSHAKE PAL (IOCP2)
ARBITR. CONTROL PAL (IOCP3)
MISC FUNCTIONS PAL (IOCP4)
CPU Backplane Outputs
Synchronize to Start of Short Half Cycle
Frontplane Inputs Control Signals from EXT
Buffers for Frontplane Control Signals
IOC Operating Status
Frontplane Outputs Control Signals to EXT
Drivers for Frontplane Control Signals
Extender Box Availability Flag
Slave Mode Flag
Data-Out Register Latch Enable for I/O Handshaking
IOC
1

Figure 12-12. IOC Card (12025-60001) Schematic Diagram (Sheet 1 of 4)

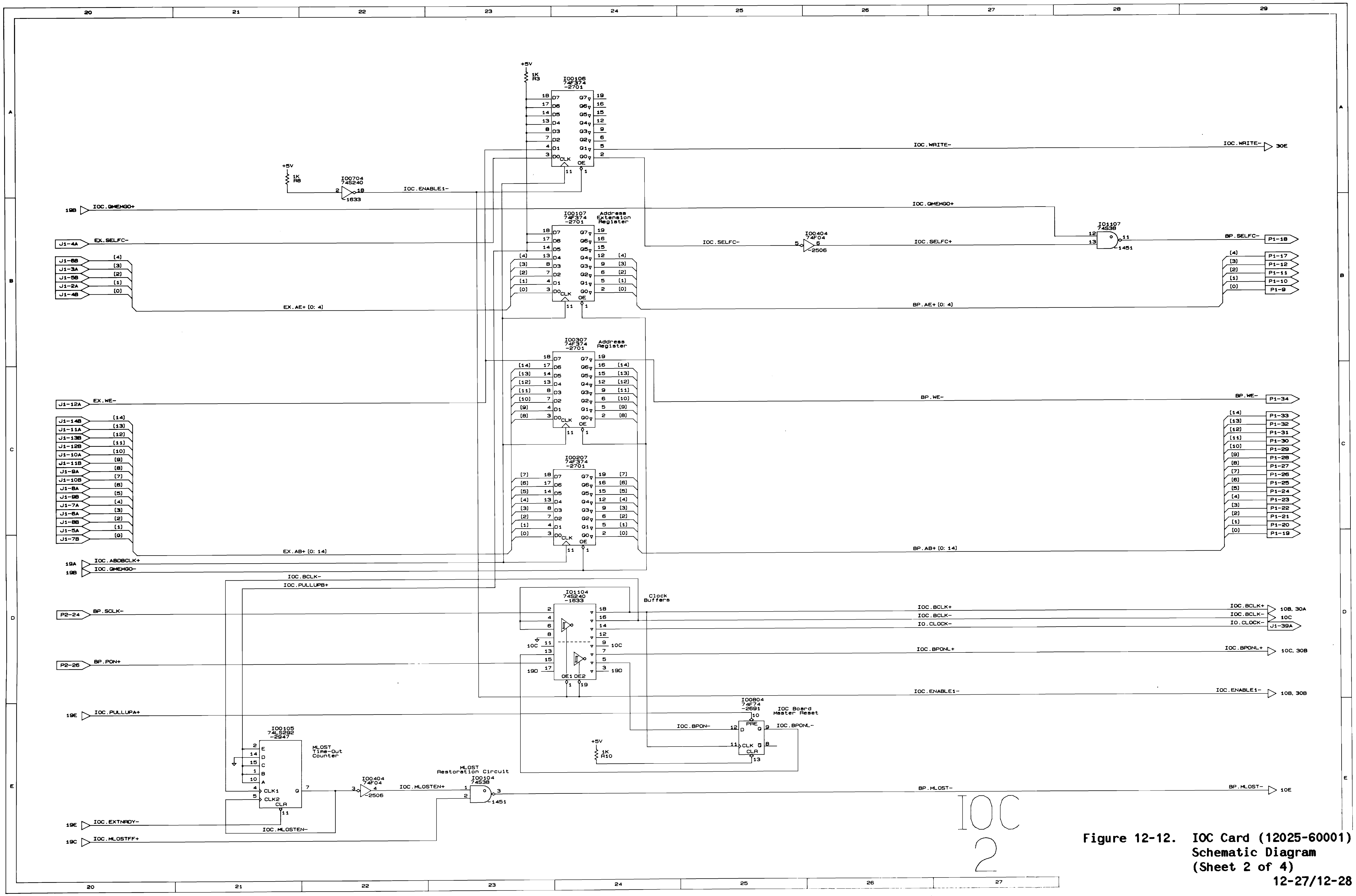


Figure 12-12. IOC Card (12025-60001) Schematic Diagram (Sheet 2 of 4)

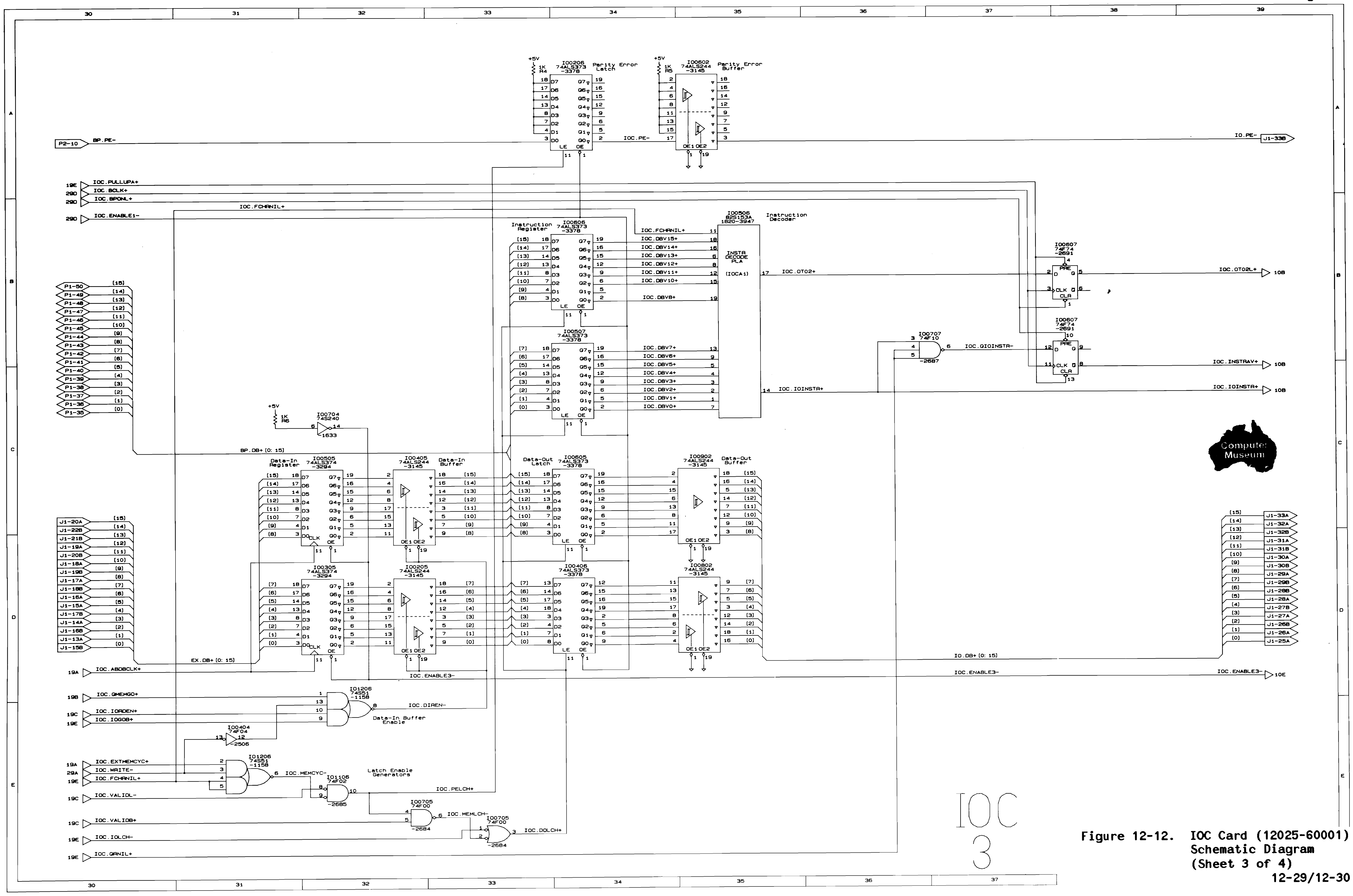


Figure 12-12. IOC Card (12025-60001) Schematic Diagram (Sheet 3 of 4)

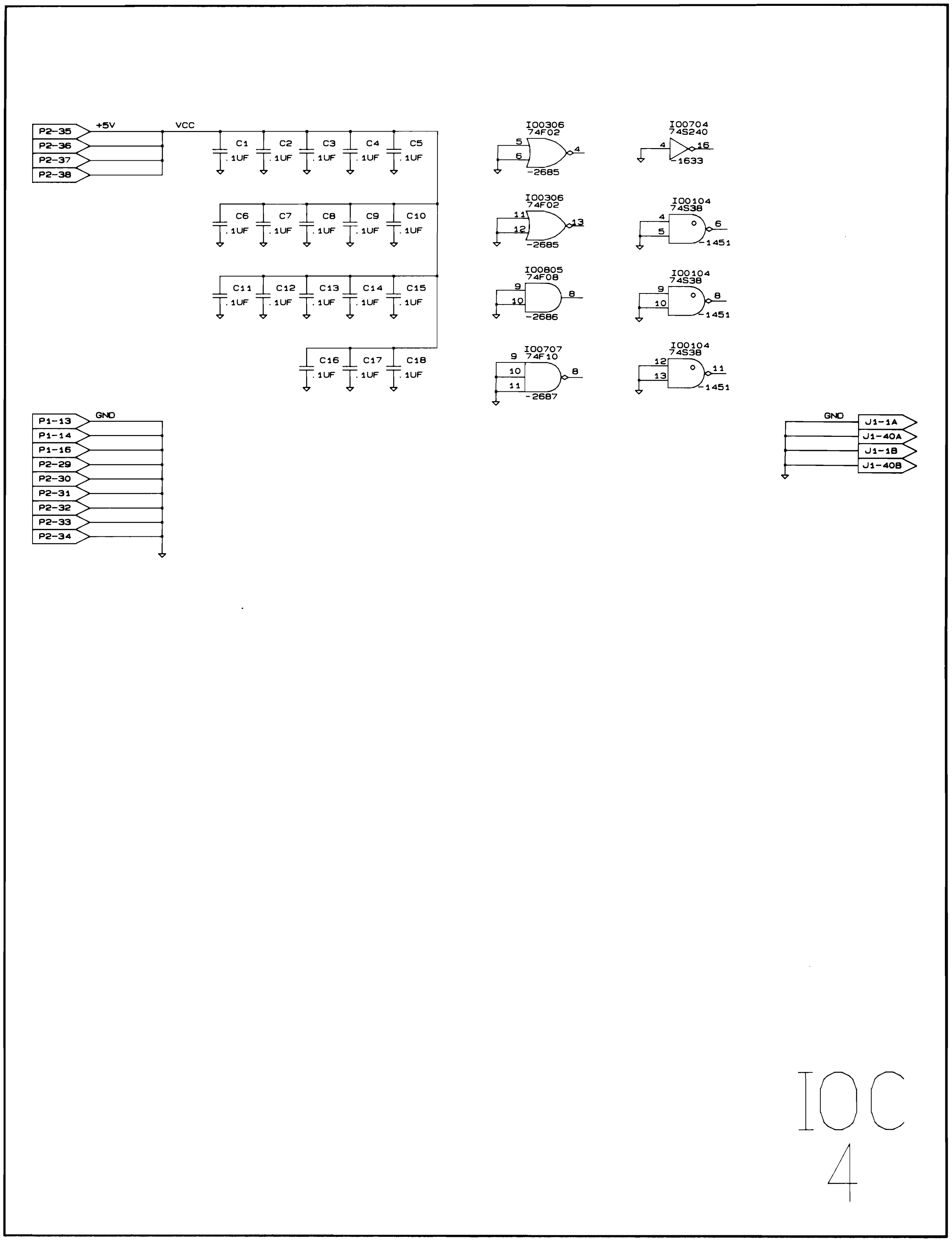
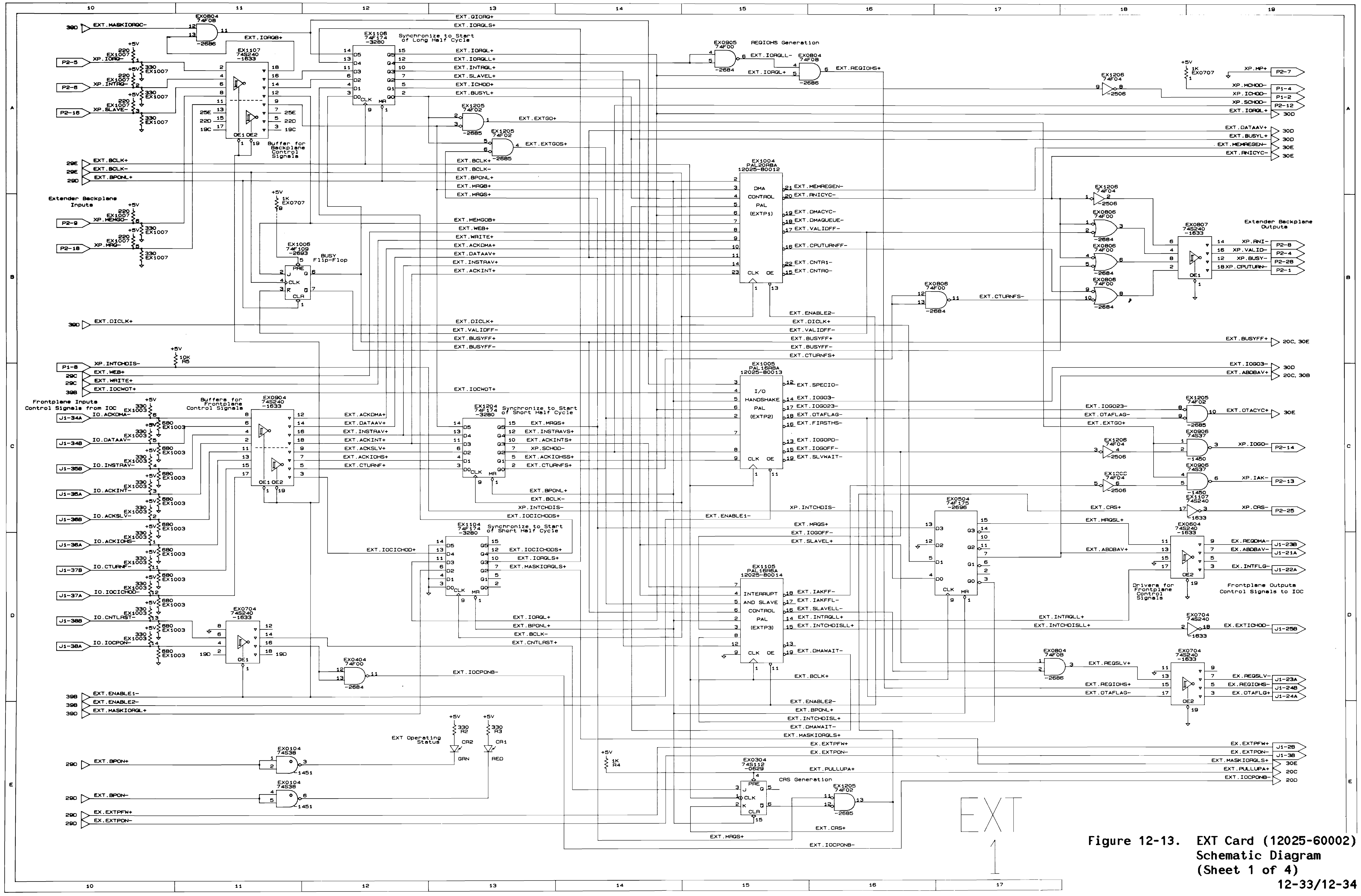


Figure 12-12. IOC Card (12025-60001) Schematic Diagram (Sheet 4 of 4)



Synchronize to Start of Long Half Cycle
EX1106 74F174 -3280
EX1107 74S240 -1633
Buffer for Backplane Control Signals
Extender Backplane Inputs
EX1006 74F109 -2693
BUSY Flip-Flop
REGIONS Generation
EX1004 PAL20R8A 12025-80012
DMA CONTROL PAL (EXTP1)
EX1005 PAL16R8A 12025-80013
I/O HANDSHAKE PAL (EXTP2)
EX1105 PAL16R6A 12025-80014
INTERRUPT AND SLAVE CONTROL PAL (EXTP3)
Extender Backplane Outputs
EX0807 74S240 -1633
Frontplane Inputs Control Signals from IOC
Buffers for Frontplane Control Signals
EX0904 74S240 -1633
EX1204 74F174 -3280
Synchronize to Start of Short Half Cycle
EX1104 74F174 -3280
EX0504 74F175 -2696
Drivers for Frontplane Control Signals
Frontplane Outputs Control Signals to IOC
EX0704 74S240 -1633
EXT Operating Status
GRN
RED
EX0304 74S112 -0629
CRS Generation
EXT
1

Figure 12-13. EXT Card (12025-60002) Schematic Diagram (Sheet 1 of 4)

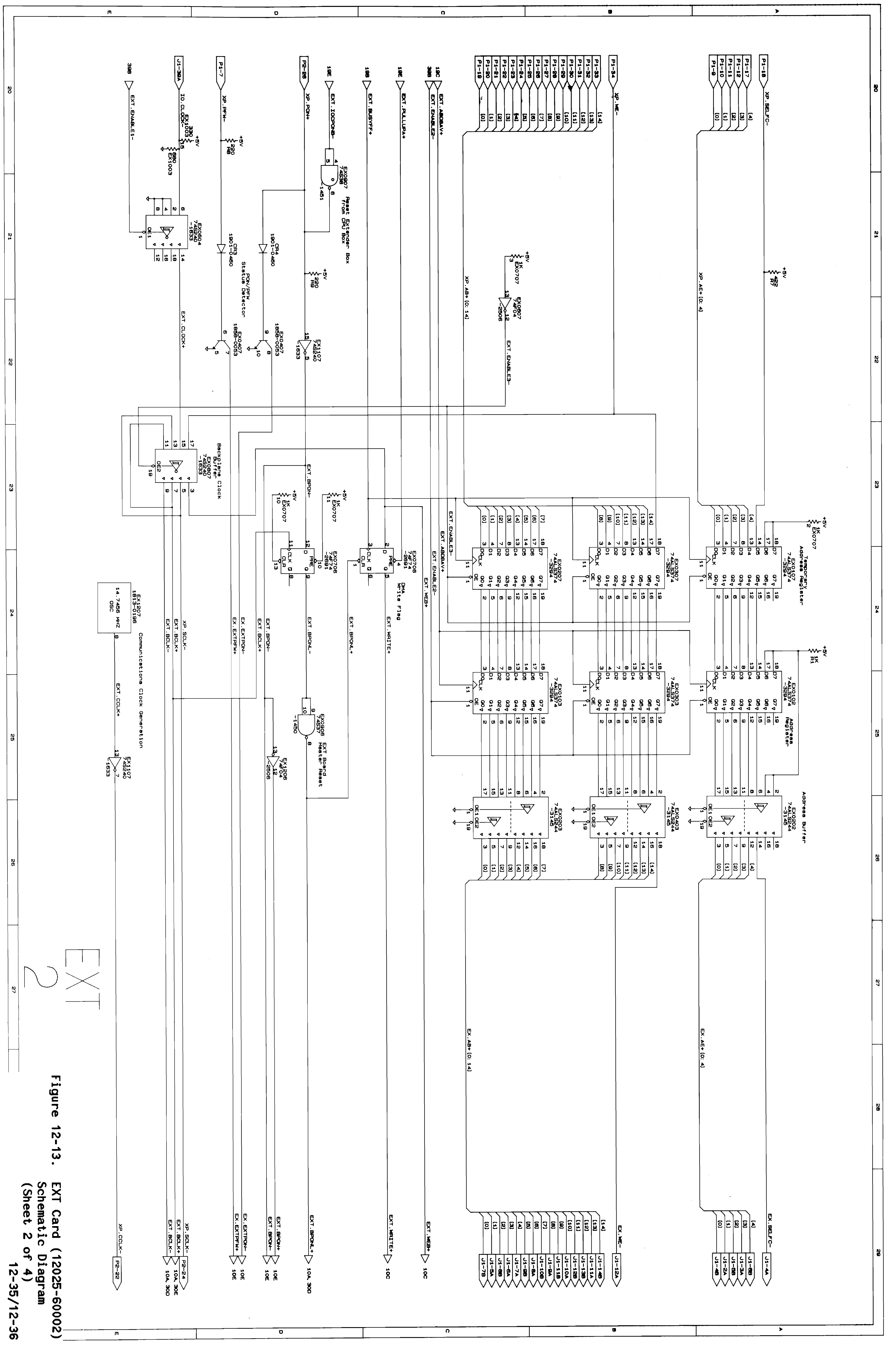


Figure 12-13. EXT Card (12025-60002) Schematic Diagram (Sheet 2 of 4)

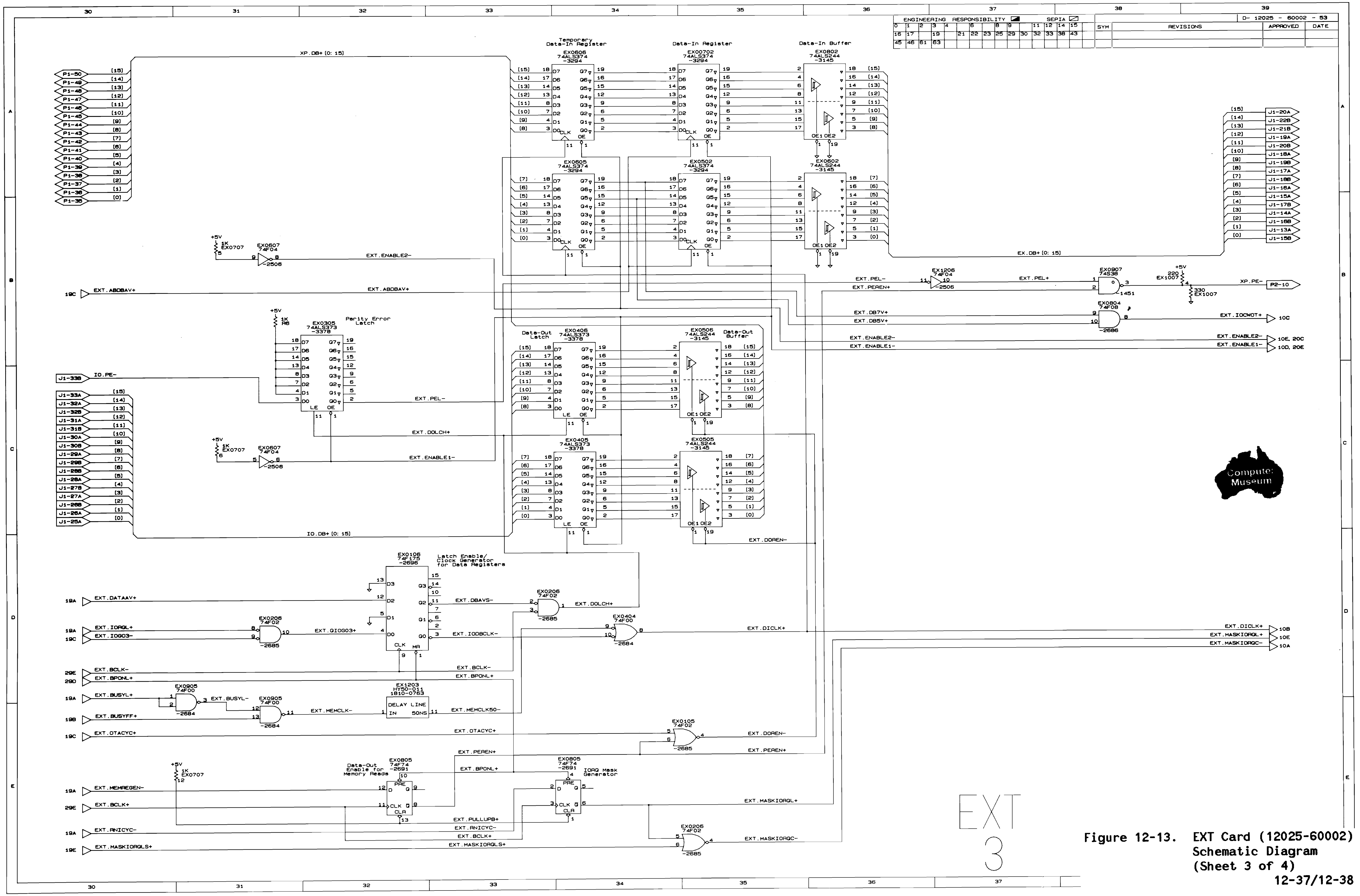


Figure 12-13. EXT Card (12025-60002) Schematic Diagram (Sheet 3 of 4)

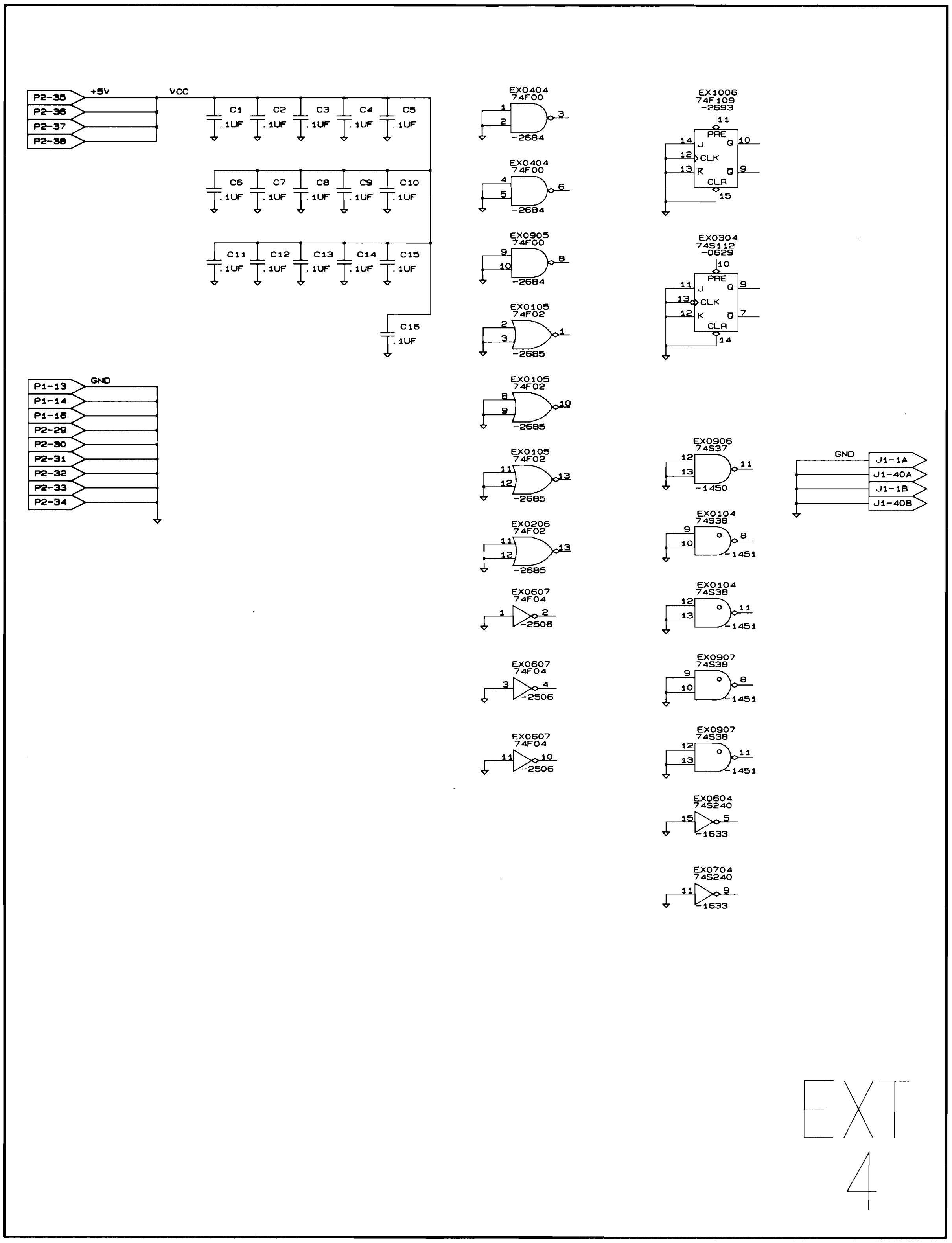
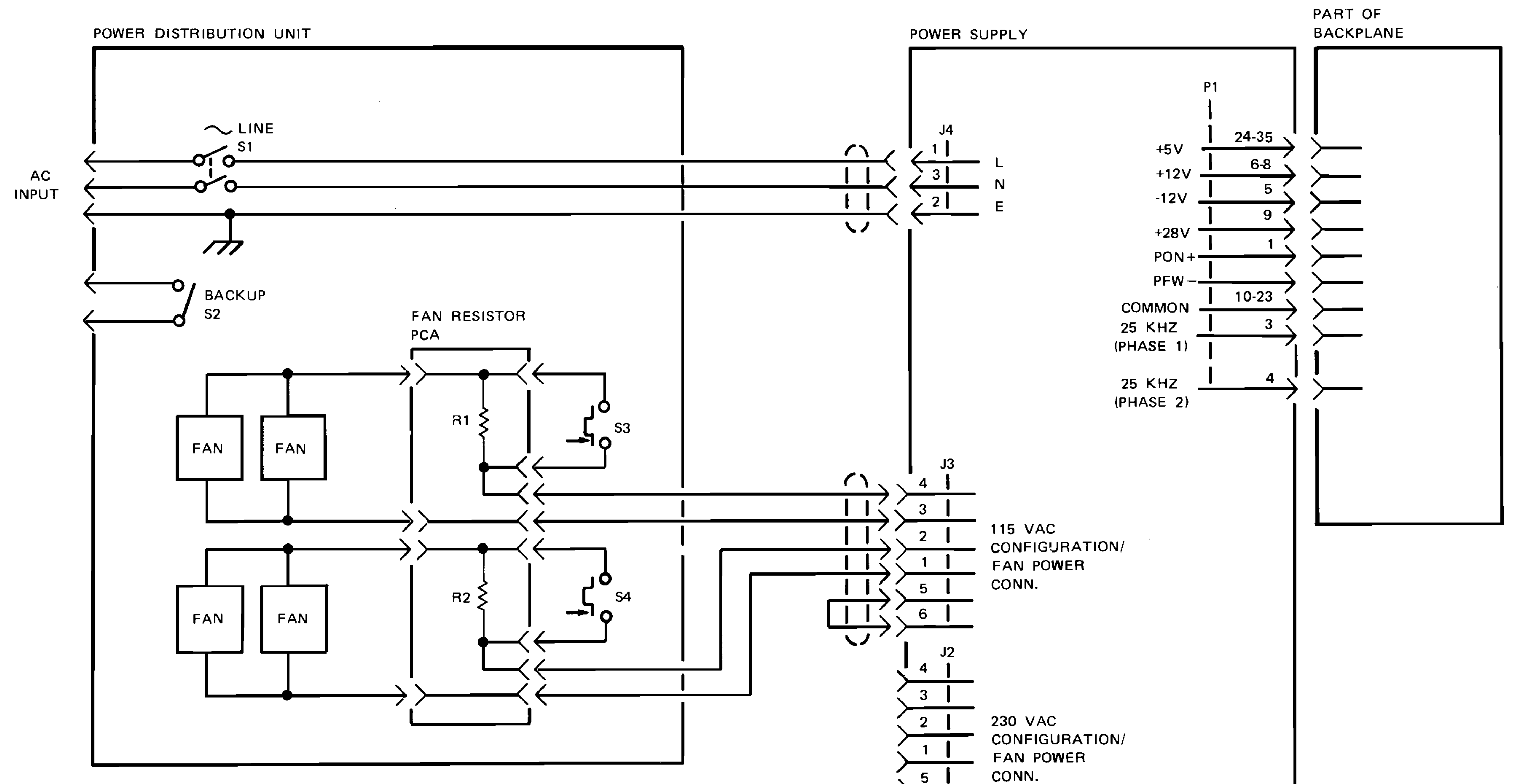


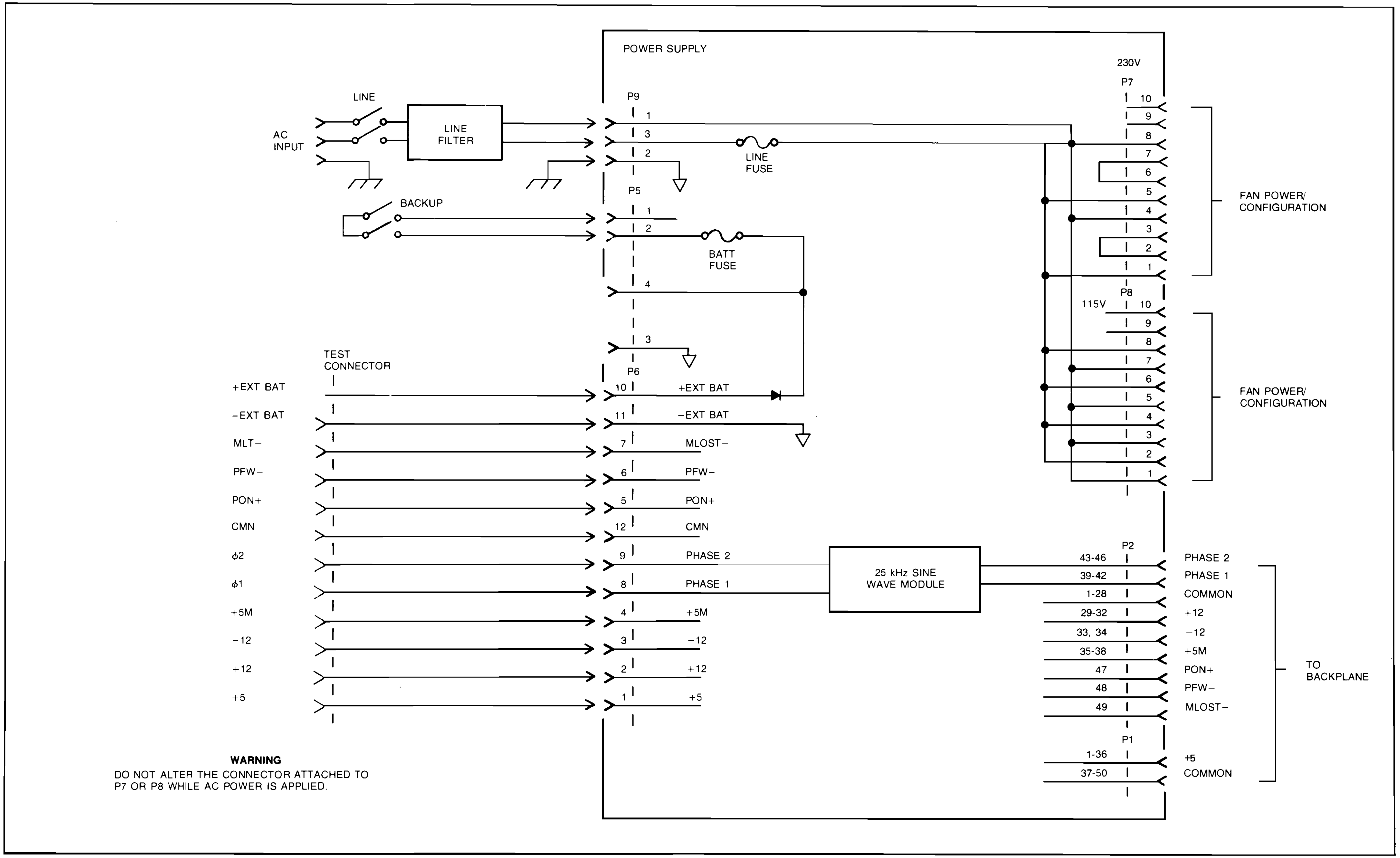
Figure 12-13. EXT Card (12025-60002) Schematic Diagram (Sheet 4 of 4)



WARNING
DO NOT ALTER THE CONNECTOR ATTACHED TO J2 OR J3 WHILE AC POWER IS APPLIED.

8400-85

Figure 12-14. HP 12025A Power Dist. Schematic Diagram



WARNING
DO NOT ALTER THE CONNECTOR ATTACHED TO P7 OR P8 WHILE AC POWER IS APPLIED.

Figure 12-15. HP 12025B Power Dist. Schematic Diagram

INDEX