A TWO-PORT MEMORY INTERFACE FOR MICROCOMPUTERS

by

Andrew Wilkins Wilson, Jr.

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 7 6

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: [signature]

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

John V. Wait Ph.D. — Sep 16, 1973

John V. Wait — Date

Professor of Electrical Engineering

PREFACE

This report is part of continuing research by The University of Arizona's Electrical Engineering Department into improved digital computer based systems for simulation and instrumentation. The device described within allows experiments involving simulation systems with multiple processors.

The author would like to acknowledge the assistance of his thesis advisor Dr. Granino Korn and the valuable suggestions of Hector Mery.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ABSTRACT

This report describes an interface between an IMP-16 microcomputer and a PDP-11 compatible two-port memory. The advantages of a two-port memory for interprocessor communication and the role of the interface in such a scheme are discussed. Next, operation of the interface's circuitry and its interaction with the microcomputer and two-port memory are explained. A multiprocessor system consisting of an IMP-16 microcomputer and a PDP-11 minicomputer is developed and several programs which demonstrate techniques for passing information between the processors are described. Finally, the use of this interface in a microprocessor development system is outlined.

CHAPTER 1

INTRODUCTION

Many applications of distributed computing require an array of microprocessors connected to a central computer which receives information from the microprocessors and, in turn, feeds them programs or data such as process-control setpoints. Since the larger computer can have software such as a disk operating system, it can also serve for program translation and for debugging the microprocessor subsystem. A system using dual-port memories for interprocessor communication is shown in Figure 1.

There are several forms which a multiprocessor system may take, each using a different method of data transfer (Korn, 1974). One method employs bus switches to connect the address and data lines of an array of processors to an array of memory and peripheral units, with the switch positions set under program control. A second method uses dual direct-memory access to transfer blocks of words from the memory of one computer to that of another through special hardware interfaces connected between pairs of computers. A third method uses so called bus windows which allows each computer to access another's memory,

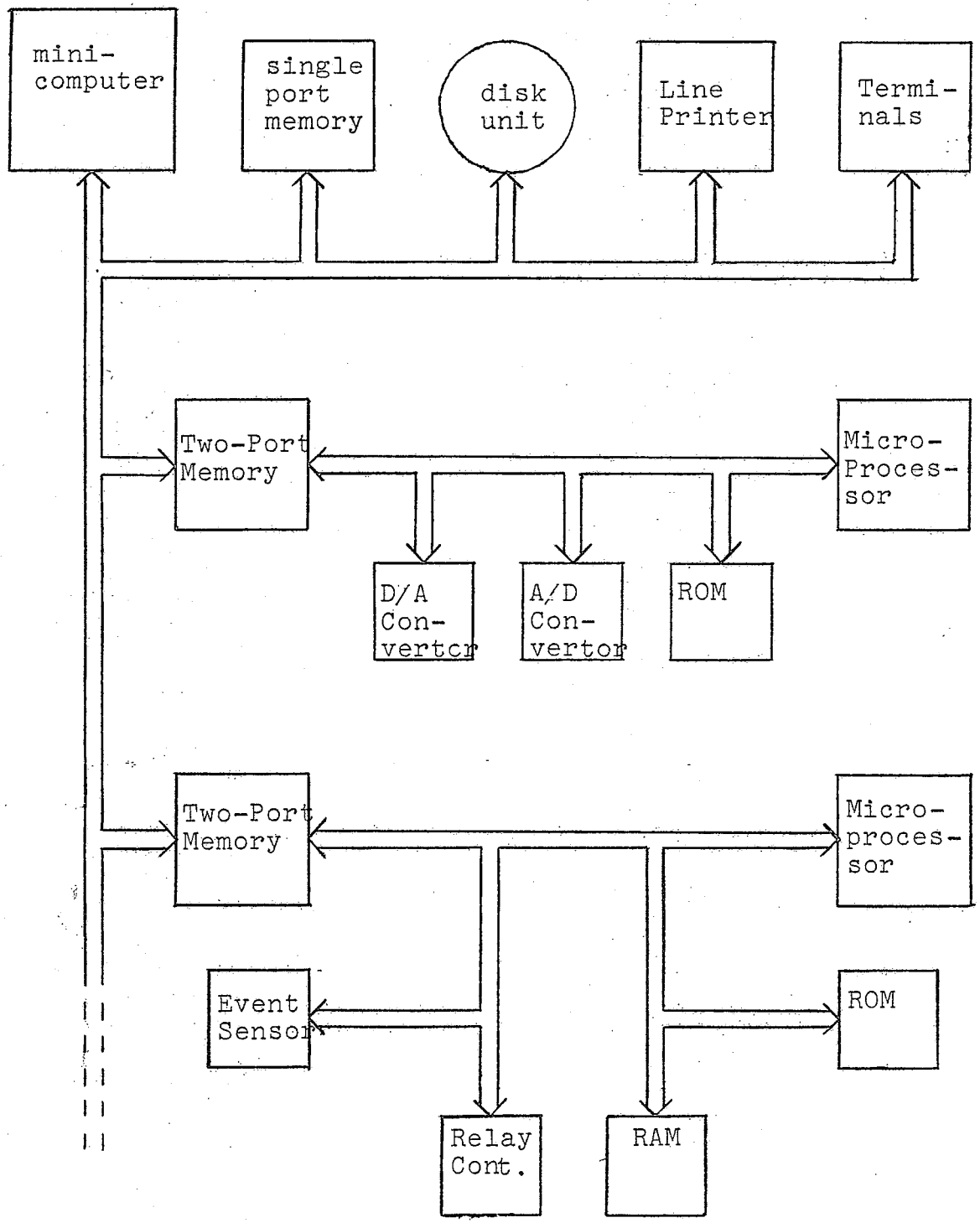


Figure 1. Typical Microprocessor Based Distributed Computing System

transferring one word at a time through direct-memory-access-type hardware.

Perhaps the best method uses multiport memories as links between the processors. A multiport memory is a memory unit which can be connected to several computers, and will grant each request for access in turn, according to a priority arrangement. In this system, programming is quite simple, since each processor can access the common memory as though it were its own. Data and instructions can be in one area of the two-port memory for the microprocessor to use, while the microprocessor can return data to the minicomputer by placing it in another area.

Two-port memory based multiprocessor systems are inexpensive and efficient. The extra memory is needed by the microprocessor anyway, and with modern integrated logic the added multiport hardware is inexpensive and readily available. Each processor can use its own memory bank without interfering with the other processors' simultaneous accessing of other memories, which can produce efficient multiprocessor operation.

CHAPTER 2

PROBLEM STATEMENT

The problem described in this report was to design an interface between a PDP-11 minicomputer and an IMP-16 microprocessor, which would permit reliable and efficient multiprocessing. The key elements of the system had already been purchased, so that many of the design requirements were pre-determined. Some attention had to be paid to the software which would be governing interprocessor communication, as well as the minicomputer's system software.

Design Requirements

The proposed multiprocessor system consists of a PDP-11 minicomputer with 16K words of resident memory, two disk drives and a graphics display, an IMP-16 microcomputer with 4K of resident memory, a Cambridge Memories 16K two-port memory with fixed priority, and the two-port memory interface. Because both computers and the two-port memory had already been purchased, the interface had to be designed to meet their signal requirements. Since the two-port memory makes up almost one-half of the PDP-11's memory, the interface should not monopolize it any more than

absolutely necessary. The interface should not unduly decrease the operating speed of the IMP-16 either, as it is already a fairly slow computer. The interface should allow the IMP-16 to use its own memory as well as the dual-port memory. Finally, some way had to be found to properly initiate IMP-16 execution of its programs once they have been loaded into the two-port memory.

Requirements for Multiport Operation

The heart of this multiprocessor system is the two-port memory. This memory must have the capability of switching between ports, connecting one or the other to its internal circuitry. In the event that both processors request memory access simultaneously, a priority arbitration scheme must decide which one is serviced first. Although some two-port memories have software selectible priority, the Cambridge Memory's unit has a fixed scheme which gives one particular port priority over the other. Even though this fixed scheme is less flexible, it is adequate in many cases, such as when one computer is subservient to another. In our system the IMP-16 will probably be accessing the two-port memory more often and more urgently, so it will be given prority.

The two computers must be able to signal task completion and requests for service to each other. There are several ways to keep each processor informed of what the other is doing, including hardware interruption, status flag sensing, and memory mailbox sensing. The interruption scheme requires the addition of expensive hardware interrupt interfaces to the multiprocessing system, although it may be required when multi-task software is used. Flag sensing also requires additional interfaces, but they are less expensive. The least expensive scheme, which is also quite fast, is the mailbox approach, where one or two locations in the two-port memory are reserved for transferring status information between processors. No extra hardware is required, as each computer simply sets bit patterns in the mailbox to indicate task completion or to request service. The requirements of low cost and simple software suggested the use of this scheme in our system.

Processor Bus Descriptions

Since the design of the interface is so dependent on the bus characteristics of the IMP-16 and the dual-port memory, a description of these buses is in order. The block diagram in Figure 2 shows how the system is interconnected. The two data buses serve as the sole

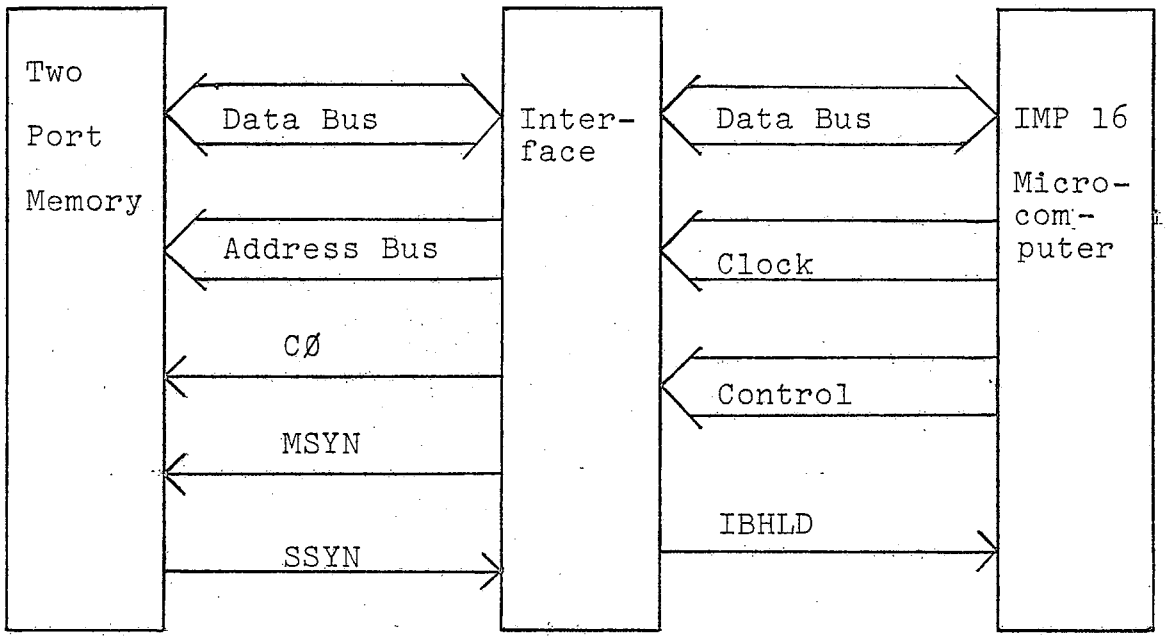


Figure 2. Block Diagram of the System
(See Figures 3 and 4 for timing.)

means of communication between the IMP-16 and the two-port memory. Therefore, an understanding of their signal characteristics is essential.

The IMP-16 is a 16-bit microprocessor with a multiplexed, synchronous data bus for communicating with memory and peripheral units. The data bus contains clock and strobe signals supplied by the IMP-16 which keep other devices synchronized with it. It also contains a set of sixteen bi-directional data lines which are used to transfer both addresses and data between the IMP-16 and its peripherals and memory.

An IMP-16 bus cycle consists of three phases, including an initial address specification phase, an intermediate waiting phase, and a final data transmission phase. Three control lines, the Interface Address Strobe (IADS), the Interface Write Data Strobe (IWDS), and the Interface Read Data Strobe (IRDS), specify the current bus phase to all devices connected to the bus, as shown in Figure 3. The IADS line is asserted during phase one, and indicates that address information is on the bus. The IWDS line is asserted during the second and third phases and may, if data is to be written to a peripheral, indicate that data is present on the bus. The IRDS line is asserted during the third phase and indicates that data is on the bus.

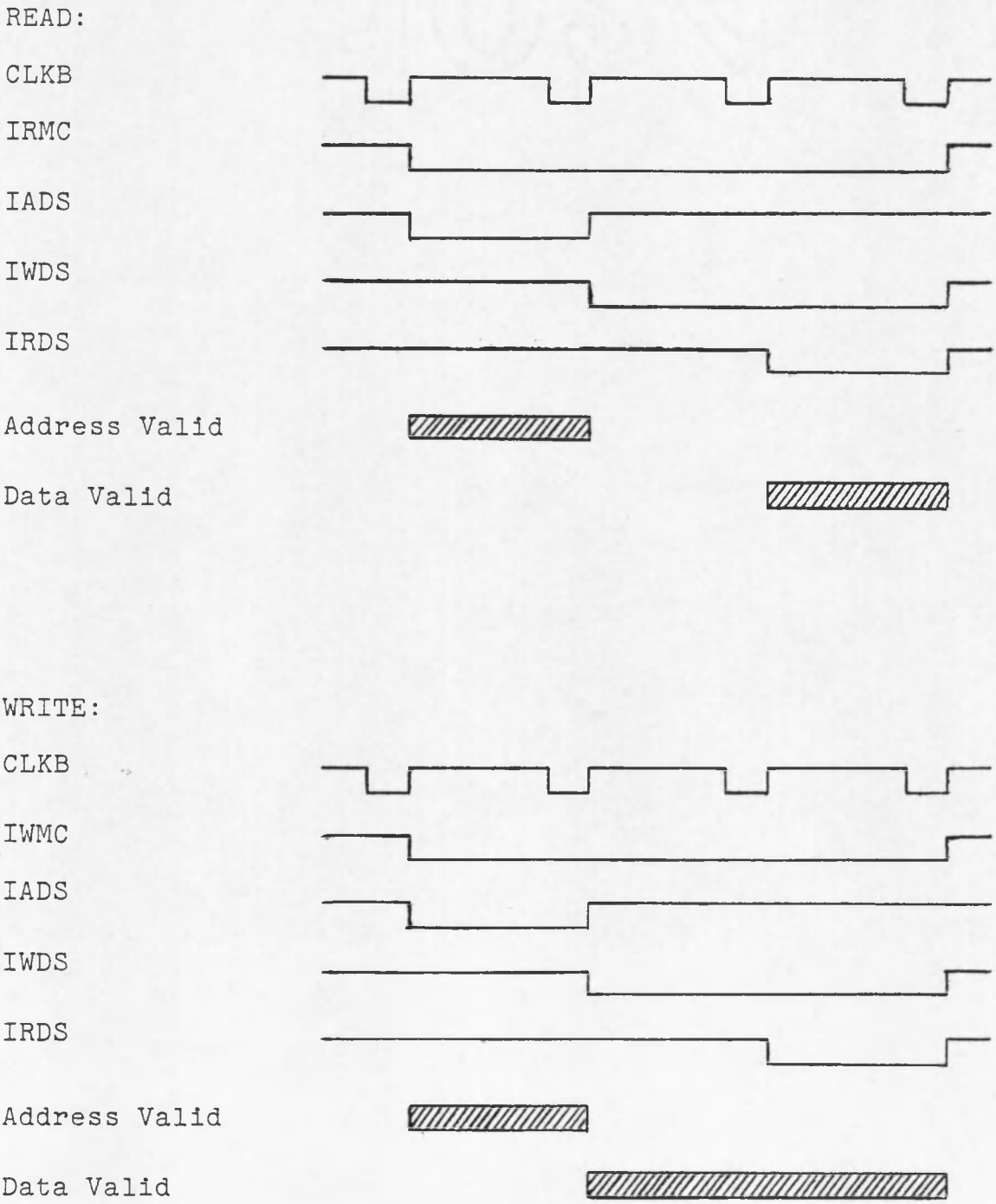


Figure 3. IMP-16 Timing Diagram

Together with the clock signals mentioned above, these signals completely define bus operation. The IMP-16 Users Manual (National Semiconductor, 1974) contains a detailed description of IMP-16 operation.

The PDP-11 is a 16-bit minicomputer with an asynchronous data bus, (UNIBUS ®), which connects all memory and input/output devices to the central processor. The UNIBUS has separate address and data lines, as well as control lines indicating what type of data transfer is taking place. Two control signals, Master Synchronization (MSYN) and Slave Synchronization (SSYN) maintain proper interlocking between devices connected to the bus.

Bus transactions begin with a master device placing address information and possibly data on the UNIBUS, as shown in Figure 4. After a 150 nanosecond delay, MSYN is asserted. When the addressed slave device has accepted the data, or placed its own on the bus, it asserts SSYN. Receipt of SSYN tells the master device to accept data, when necessary, and clear MSYN. When the slave device notices that MSYN has been cleared, it clears SSYN, thus ending the bus cycle. The PDP-11 Peripherals Handbook (Digital Equipment Corporation, 1975) contains a more detailed description of Unibus operation.

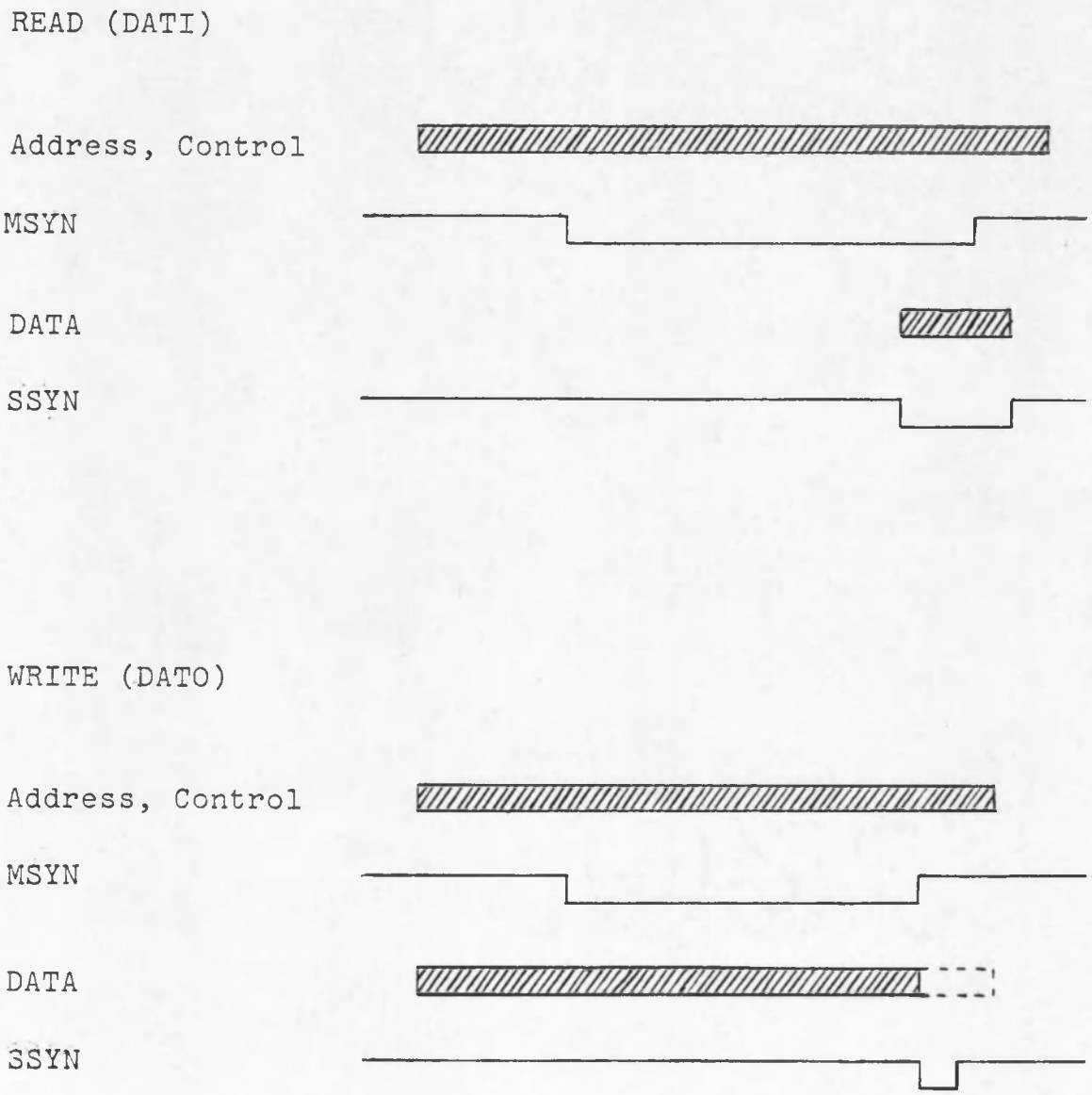


Figure 4. PDP-11 UNIBUS Timing Diagram

CHAPTER 3

INTERFACE DESIGN

Interface operation is mainly determined by the requirements of the IMP-16 and the two-port memory. Each requires a specific sequence of electrical signals between it and the interface to work properly.

The PDP-11 system uses monostable multivibrators and delay lines to establish UNIBUS timing. Since these devices are asynchronous rather than clocked, this allows maximum bus speed. Several designs of this type were tried, but the interface must be resynchronized with the IMP-16 at the end of the memory cycle, so no advantage is gained. Furthermore, use of crystal-controlled clock signals generated by the IMP-16 results in better noise immunity and temperature independence. Figures 5 and 6 describe the interface.

Memory Read Operation

All memory transactions begin with the IMP-16 placing address information on the bus and asserting the interface Address Strobe (IADS) line. The IMP-16 designates a memory read cycle by asserting the Interface Memory Read Cycle (IRMC) line. As the read timing diagram

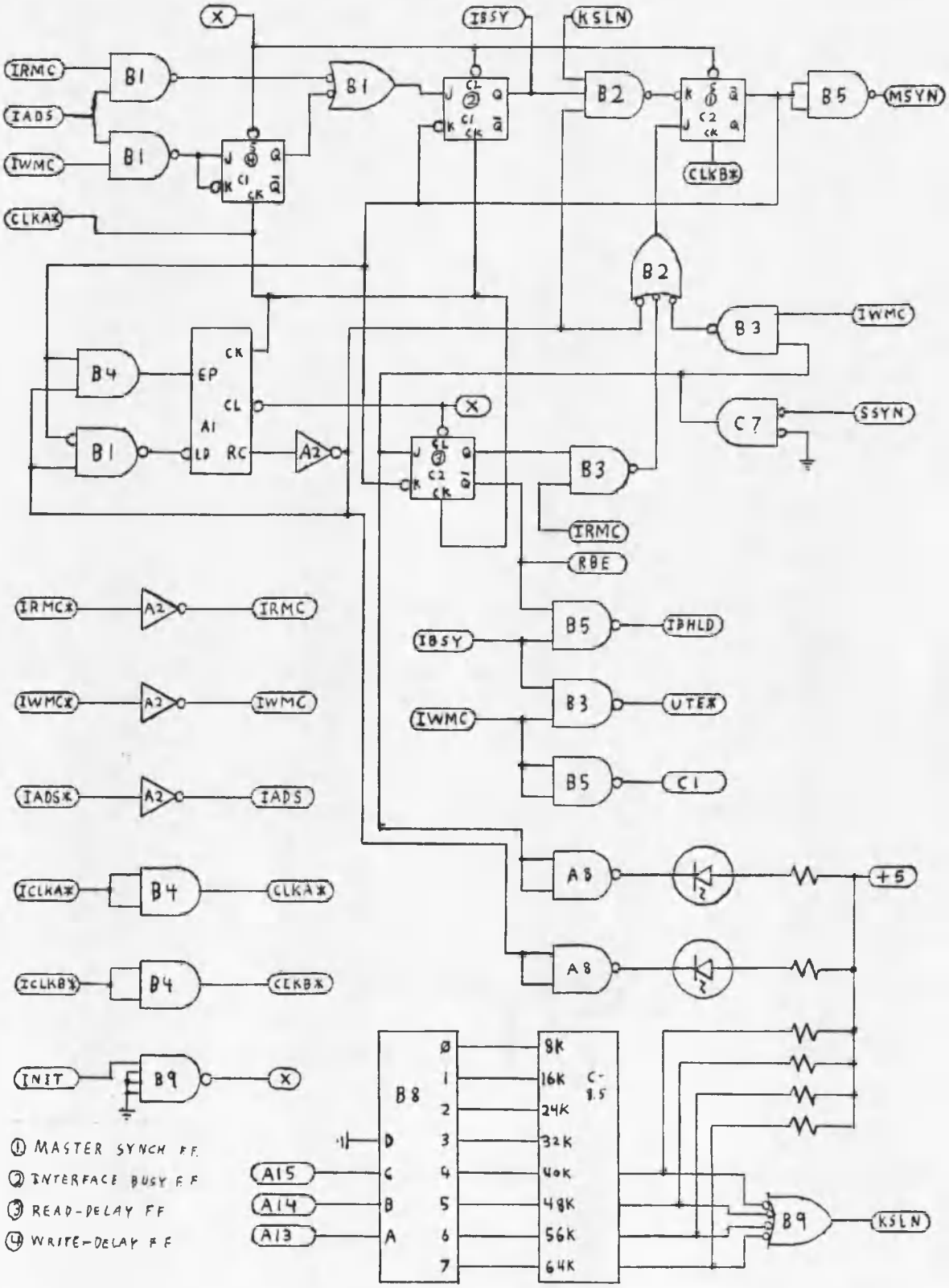


Figure 5. Interface Control Logic

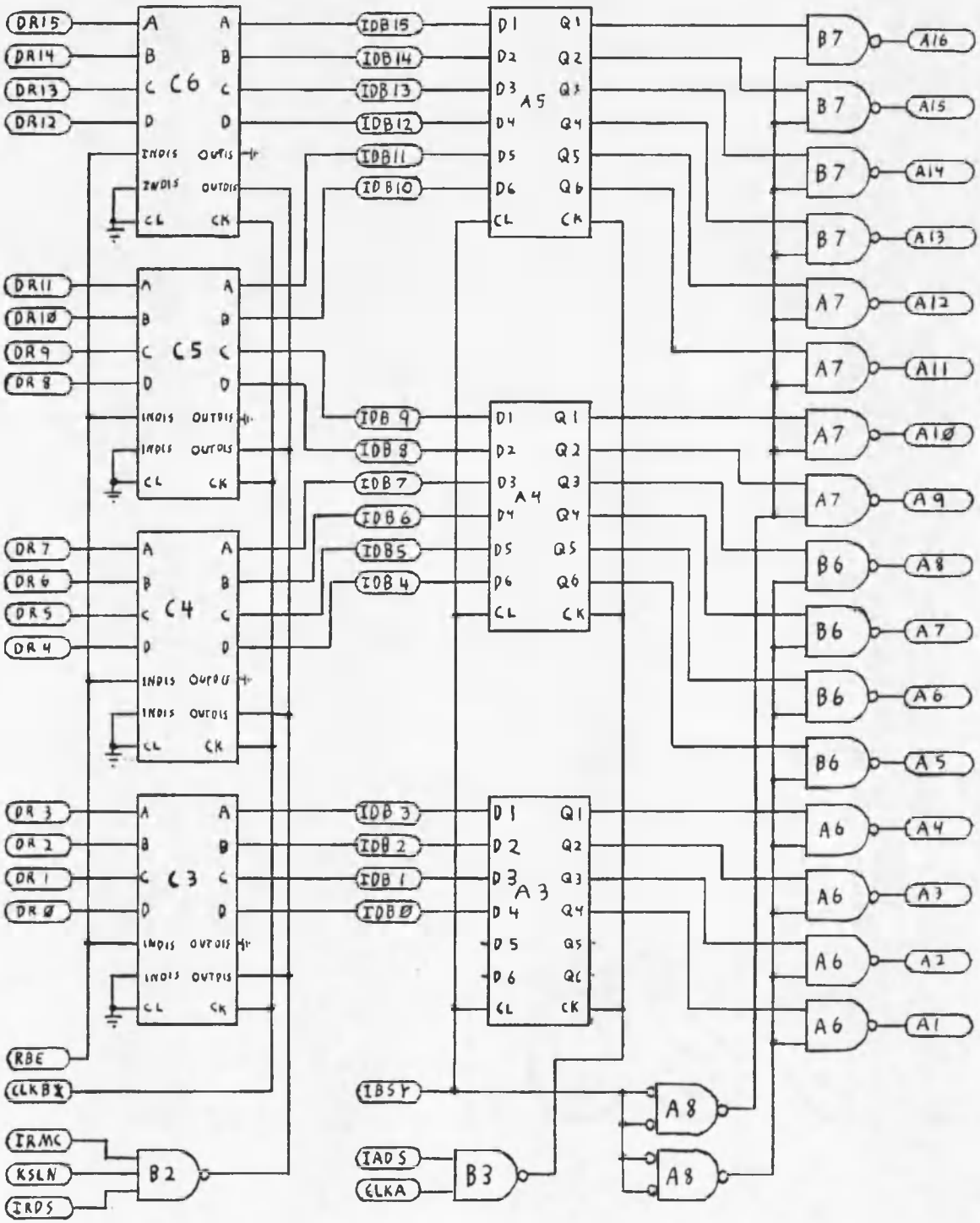


Figure 6. Interface Data Buffers

in Figure 7 shows, the interface stores the address in latches at the clock A pulse. At the same time, interface operation is initiated by setting the Interface Busy Flip Flop.

The TRUE condition of the Interface Busy Flip Flop places the memory address on the Unibus Address lines and asserts the Interface Bus Hold (IBHLD) line. The address information is sent to the two-port memory at this time, so that it can perform a read operation as soon as possible. The IBHLD line is asserted to allow extra time for the read operation to occur, and will not be cleared until data has been received from the two-port memory. Finally, if the memory address is in the correct space, the next clock B pulse (175 nanoseconds later) will set the Master Synchronization Flip Flop.

Assertion of the Master Synchronization Signal (MSYN) initiates a memory cycle in the two-port memory. For a memory read operation the interface sets C_1 to zero at the same time that it applies address information to the Unibus, telling the two-port memory whether to do a read or write cycle.

When the two-port memory has read the requested word, it places it on the Unibus and asserts the Slave Synchronization Signal (SSYN). The interface must wait at

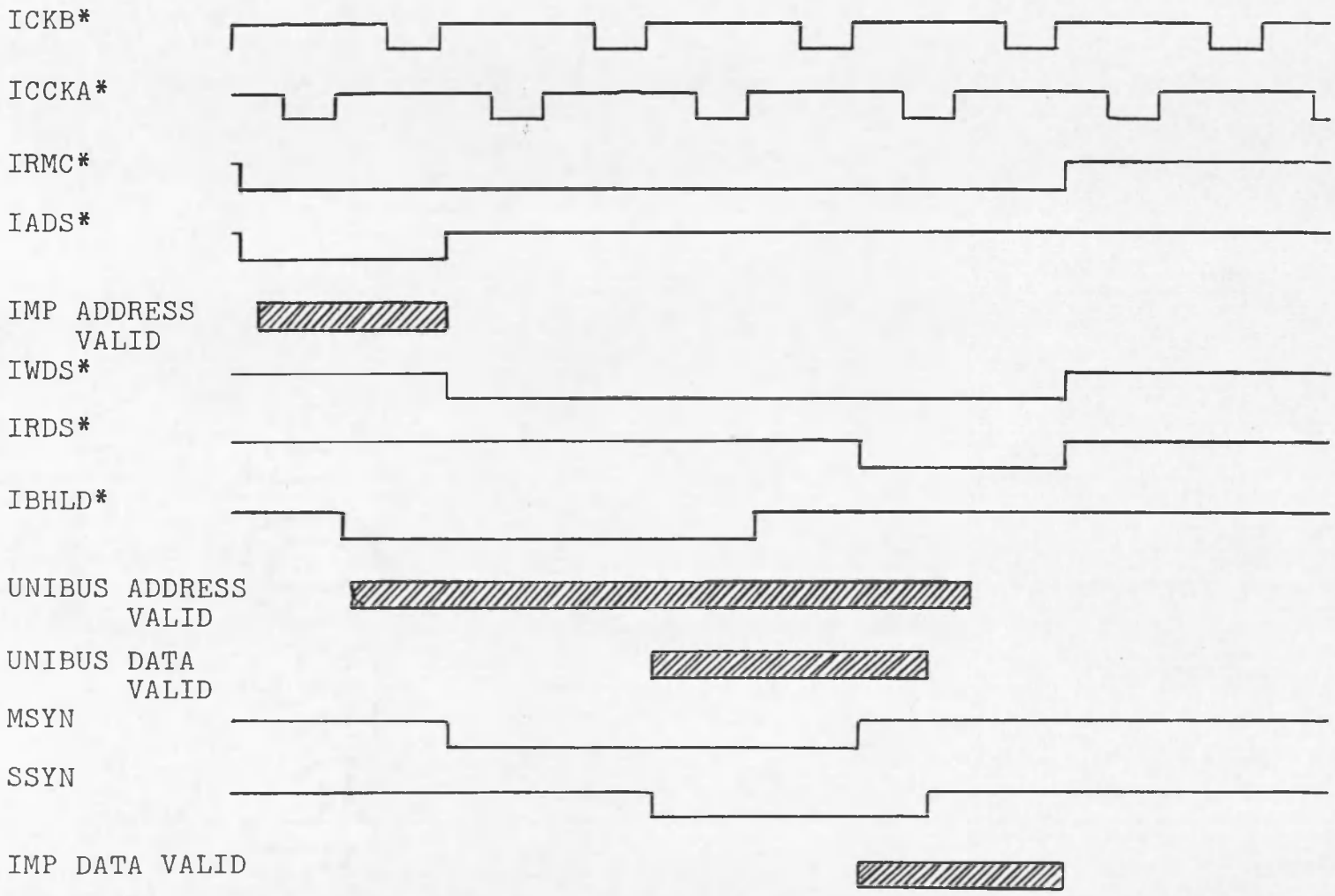


Figure 7. Read Timing Diagram

least 150 nanoseconds before accepting data. Therefore, SSYN sets the read-delay flip flop, rather than directly reseting MSYN. Since the read-delay flip flop uses clock A, while the master synchronization flip flop uses clock B, at least a 175 nanosecond delay will result. Once the read-delay flip flop is set, it resets the master synchronization flip flop and latches the data at clock B. It also releases the IBHLD line, allowing the IMP-16 to resume processing.

With MSYN cleared, the two-port memory clears SSYN and waits for another request from either the PDP-11 or the IMP-16. The IMP-16 recognizes the cleared IBHLD line at the same clock B pulse which latched data into the data holding latches. The IMP-16 accepts this data during the following clock B period. This completes a memory read cycle.

Memory Write Operation

Many aspects of a write cycle are identical with those of a read cycle, therefore only the differences will be described. The interface write timing diagram is shown in Figure 8. The interface stores address information sent to it during the first bus clock period, but does not begin a memory transaction yet. The write-delay flip flop is set

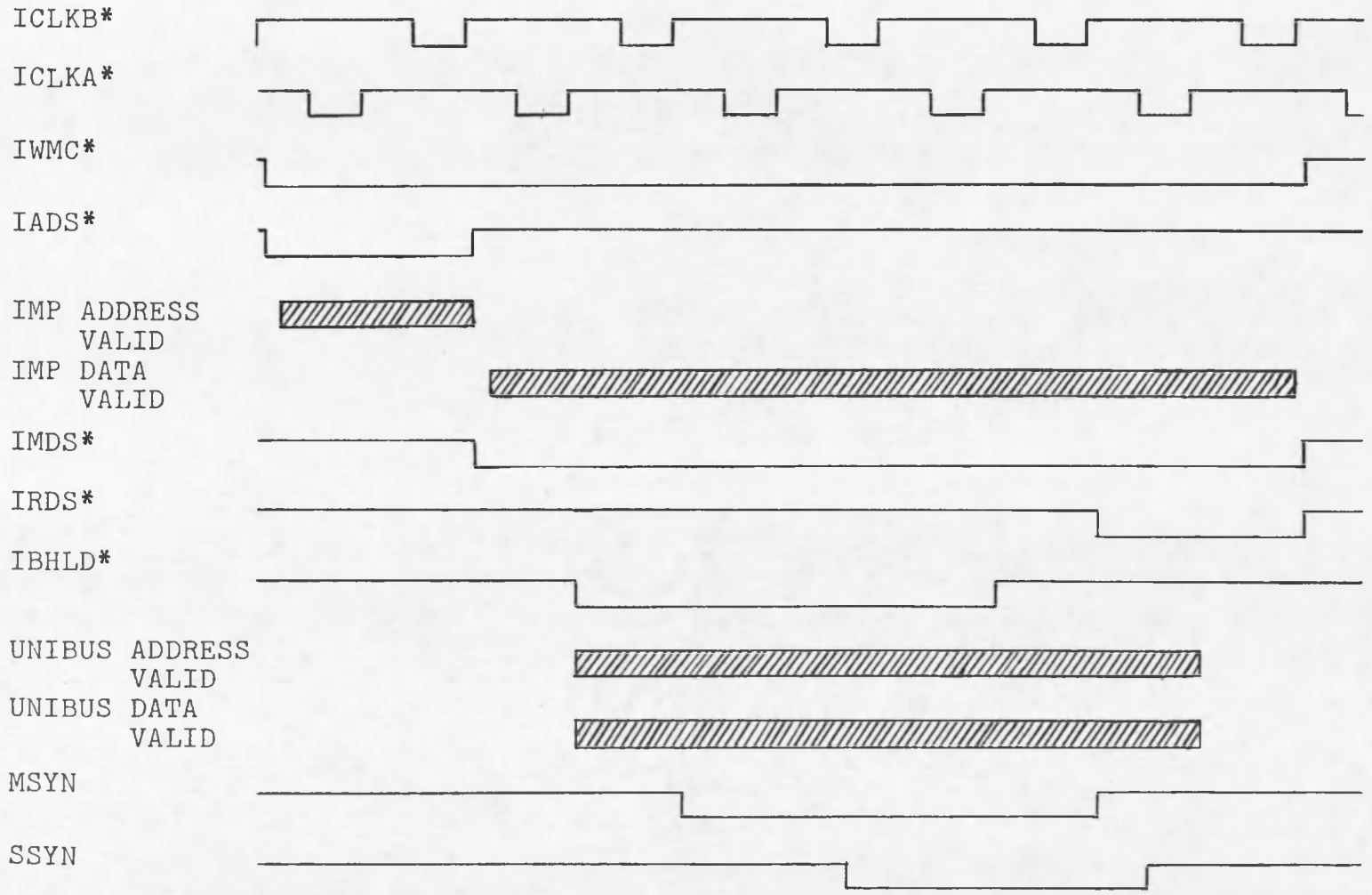


Figure 8. Write Timing Diagram

instead, which holds up the interface for one bus clock period, allowing time for the IMP-16 to place data on the System Data Bus.

After this delay, the interface-busy flip flop is set, initiating a memory operation. Both address and data are placed on the Unibus at this time, while C_1 and IBHLD are asserted. If the address is valid, MSYN is asserted 175 nanoseconds later. Receipt of this by the two-port memory initiates a write cycle.

Once it has accepted the data, the two-port memory asserts SSYN. This signal resets the master synchronization flip flop at the next clock B pulse. SSYN will also set the read delay flip flop at the first clock A pulse. If SSYN arrives after clock B but before clock A, the read-delay flip flop will clear IBHLD early, allowing the IMP-16 to complete its cycle one bus clock period sooner than it otherwise would.

As soon as the master synchronization flip flop is reset, it clears MSYN, and at the next clock A pulse all other flip flops are cleared. This ends the cycle.

Address Assignment

In order to allow other memory units to be connected directly to the IMP-16 bus, the interface verifies that the requested memory location is actually in the two-port

memory before it initiates a read or write cycle. If the requested location is outside the dual-port memory space, no two-port accessing is attempted, and the interface does not interfere with the IMP-16 bus cycle.

Address check logic determines whether the requested location is in a pre-assigned block of addresses. The three most significant bits of the address are sent to a binary to octal decoder. Each of its outputs represents an 8K word section of memory. Jumpers in the address select socket determine which sections are accessible by routing the signal to the interface-enable gate. The interface will place address and control information on the UNIBUS, but will not assert MSYN if it is not enabled.

After one IMP-16 bus clock period, the address will be removed, and the interface reset, as shown in Figure 9. The IBHLD line will be cleared, allowing a read cycle to continue without delay, and a write cycle to continue with only a short delay. This allows the IMP-16 to access internally stored programs, such as frequently used subroutines, without delay. Because MSYN is not asserted when using resident memory, the two-port memory is not bothered, allowing the PDP-11 to run at full speed.

Figure 9. Non-existent Address Timing

Trouble Indicators

In normal operation the two-port memory will respond in 300 to 1000 nanoseconds. If SSYN is not asserted after 3 microseconds a timing circuit in the interface will light a trouble indicator and disable the interface. The interface will not resume operation until the IMP-16's Master Reset Button is pressed.

The second indicator monitors SSYN. When the IMP-16 is properly accessing the two-port memory it glows dimly. If completely out it means the two-port memory is not being used, which may indicate trouble.

CHAPTER 4

PROGRAMMING

Although the interface could be connected to any PDP-11 compatible memory, the main advantage of this unit is that it allows connection to a two-port memory which is shared with a minicomputer. The minicomputer's extensive software can be used to create and assemble programs for the IMP-16, which are then placed in the two-port memory. The IMP-16 executes them, returning the results to the minicomputer for analysis, or displaying the results directly. Once debugged, the program may be stored in ROMs for use in the field.

Program Loading

One of the prinicpal uses of two-port memories is the transfer of programs from a central minicomputer to its associatied microprocessors. The minicomputer used in this system is a PDP-11 operating under the RT-11 operating system. The two-port memory is in the top half of the address space, and is used by RT-11 for storing the monitor and device handlers. Thus IMP-16 programs put in the two-port memory may interfere with proper PDP-11 operation. However, the top 4K words of the two-port memory cannot be

directly used by the PDP-11, so IMP-16 programs placed there cause no difficulty.

Putting programs into the top of the memory requires some ingenuity. Attempts to place data there directly will fail. The solution is to use the disk as a transfer medium, since it has a full 18-bit address register.

In order to demonstrate the feasibility of the two-port memory scheme, a simple software development program was written. This program allows a user to enter programs at the PDP-11 console terminal and place them in the two-port memory for execution by the IMP-16. It also allows data in the two-port memory to be displayed in the PDP-11 terminal for analysis.

The program accepts data in hexadecimal format and stores it in a 4K word buffer. This buffer can be displayed in hex for analysis. By typing in command characters, the contents of the buffer may be copied to or from the top of memory. Thus, programs may be entered into the buffer, and then transferred to the IMP-16 area for execution.

The program has five basic commands. The Insert (I) and View (V) commands are used to load and examine data. These commands are followed by hexadecimal numbers indicating the address or addresses desired. A single 4

digit hex number specifies one address; two hex numbers separated by a colon indicates a range of addresses. The insert routine types out each address as a prompt, then waits for the user to type in one word of data in hex. The view routine displays eight memory locations on each line, along with the starting address of that line. Typing a Quit (Q) command will abort the insert routine, in case you specify too large an address range.

To place data in the top 4K words of memory requires a send (S) command. To get it back requires a retrieve (R) command. These commands copy all 4K words at once, completely updating either the buffer or the top 4K.

Interprocessor Communication

More direct communication between the IMP-16 and the PDP-11 is also possible. Once a user written program is running on the PDP-11, interference with system programming is no longer a problem. Therefore the entire two-port memory may be used by the multiprocessor system. Communication areas may be set up in that portion of the two-port memory which is accessible to both computers, and a mailbox location used to pass status information between the processors. This results in a true distributed computer system.

Figure 10 shows a sample program which demonstrates how data can be passed from the IMP-16 to the PDP-11. The program generates the squares of the numbers from 1 to 256 and places them in the two-port memory. The development program described earlier can be used to access and display the results.

IMP-16 Monitor Programs

Because the IMP-16 begins executing programs as soon as the power is turned on, some sort of monitor program is necessary to keep the IMP-16 occupied until program loading is completed. This program may be put in a Read-only Memory (ROM) or in the top 4K of the two-port memory, although a ROM is more secure.

An example program is shown in Figure 11. This program clears a location in the dual-port memory and then waits for the PDP-11 to place a non zero number in it. This number will be the address where program execution should commence, and the monitor will do just that. After a program has been finished, the IMP-16 should be sent back to the monitor to wait for a new start address.

```
          ;A PROGRAM TO FIND THE SQUARES OF NUMBERS
          ;FROM 1 TO 256.  RETURNS THE RESULTS TO
          ;LOCATIONS 7200-727F.
;
;
71F0    890A    START:  LD      2, STAD(1)  ;INIT INDEX
71F1    4F7F            LI      3, COUNT    ;INIT COUNT
71F2    AD0A    LOOP:   ST      3, NUM(1)   ;MULTIPLICAND
71F3    3D81            RCPY    3, 1
71F4    0580            MPY     NUM(1)      ;FORM SQUARE
71F5    0008
71F6    A600            ST      1, 0(2)     ;STORE RESULT
71F7    4AFF            AISZ    2, -1       ;DEC INDEX
71F8    4BFF            AISZ    2, -1       ;DEC COUNT
71F9    21F8            JMP     LOOP(1)     ;NOT DONE
71FA    2501            JMP     @RETN(1)    ;MON RETURN
71FB    727F    STAD:   .WRD    727F
71FC    FFFE    RETN:   .WRD    FFFE
71FD    0000    NUM:    .WRD    0
                        .END
```

Figure 10. Multiply Test Program

```
;       A PROGRAM TO TRAP THE IMP-16 WHEN IT FINISHES
;       POWER UP INITIALIZATION.  THE IMP EXECUTES A
;       CONTINUOUS LOOP UNTIL PROGRAMS HAVE BEEN LOADED
;       FOR IT.
;
;
7FF0    3A82    WAIT:   RXOR    2,2         ;CLEAR R2
7FF1    B90B            ST      2, @MBP(1)  ;STO MAILBOX
7FF2    910A    LOOP:   LD      0, @MBP(1)  ;LOAD MAILBOX
7FF3    11FE            BOC     1, LOOP     ;TST FOR 0
7FF4    9908            LD      2, @MBP(1)  ;GET STRT LOC
7FF5    2200            JMP     0(2)        ;GO!
7FFD                    .=7FFD
7FFD    7FFF    MBP:    .WRD    7FFF        ;MAILBOX PTR
7FFE    21F1            JMP     WAIT(1)     ;BEGIN PGM
7FFF    0000            .WRD    0           ;CURRENT PTR
                        .END
```

Figure 11. Monitor Program

CHAPTER 5

RESULTS

Initial testing of the two-port memory interface was done with special repetitive programs stored in ROM. These programs continuously read or write from the two-port memory, allowing easy checkout of all signals on an oscilloscope. This was used to test the initial prototype as well as the finished interface.

A PDP-11 type memory was simulated by a switch register and a bank of light emitting diodes (LED). Using a simple program stored in ROM, the switch register contents were repeatedly fetched by the computer and stored in the LED bank. Unfortunately, those LEDs which should have been off glowed slightly, indicating that false data was being read and written. After much troubleshooting, the problem was found to be one of inadequate ground planes in the prototype.

Apparently the IMP-16 is very sensitive to ringing and ground problems. To counter this, a ground matrix and twisted wire pairs were used in the final version. Each integrated circuit ground was connected to the four adjacent grounds. All signal lines between the IMP-16

and the interface use twisted pairs with one member grounded. As a final precaution, the power ground was run with 10 gauge wire.

Because it does not have a halt state, the IMP-16 begins running programs as soon as the power is turned on. It is possible to store a waiting loop program in ROM that keeps it busy while programs are prepared, but this has not been done yet. Instead, address line A15 has been disconnected inside the interface so that the IMP-16 will begin at the second to last location in the two-port memory. This means that a program must be waiting there for the IMP-16 to use. Since the two-port memory is a core memory, a simple monitor program, such as the one described in the previous chapter, should remain there indefinitely.

The interface was built using a Digital Equipment Corporation Flip-Chip prototyping card and two Unibus Tranceiver cards. Two connection blocks for this were installed next to the IMP-16, and leftover slots were used for the UNIBUS connection points. Wire-Wrapping ® was used for most of the connections to save time and make future modifications easier.

To use the multiprocessor system, plug one end of the UNIBUS and a UNIBUS terminator card into the vacant

slots in the two-port memory. Plug the other end of the UNIBUS and a second terminator card into the two slots of the interface. Turn the PDP-11 on first, then the IMP-16. As the IMP-16 and interface are currently configured, the IMP-16 will immediately begin running programs in the top 4K words of memory.

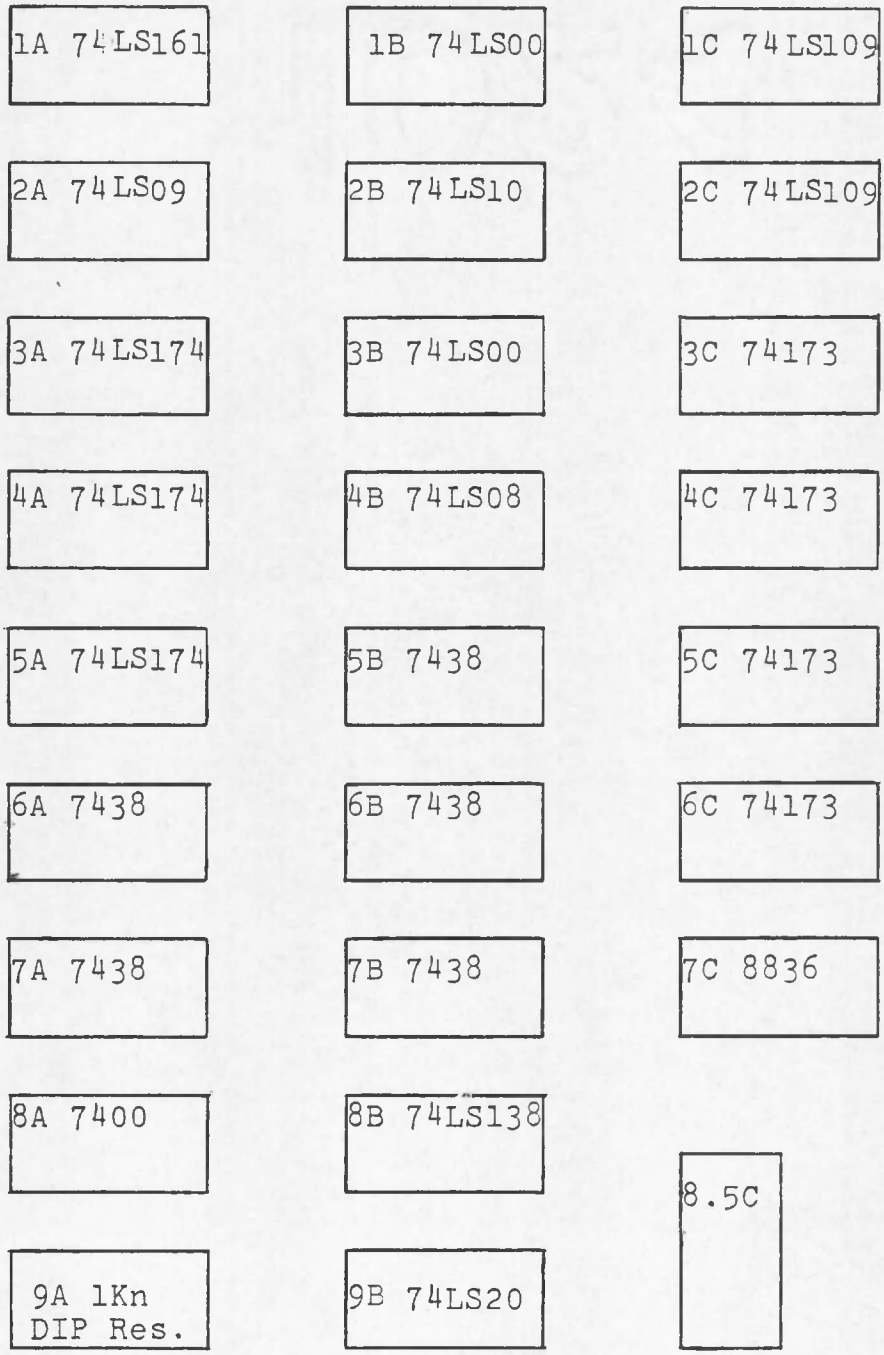
The two-port memory interface works reliably. Although the PDP-11 operating system posed some programming problems, they were not insurmountable. Several test and demonstration programs have been run on the finished system without failure. Our results indicate that two-port memories are a simple and powerful method of creating a multiprocessor system. Two-port memory based multiprocessing systems should prove useful in many distributed computing applications.

The program development software described above only just begins to utilize the power of the PDP-11 minicomputer. The use of a cross-assembler to generate microcomputer programs on the PDP-11 is one possibility, but for microprocessor applications in such areas as control and filtering, a more advanced system is desirable. A good idea for further research is the use of MICRODARE/ELEVEN to produce programs for the IMP-16 (Korn, 1975).

MICRODARE employs BASIC as an operating system and program-editing facility. The PDP-11 translates block-diagram language programs into binary code, which will be loaded directly into the two-port memory, where it is then executed by the microprocessor. By adding digital-to-analog and analog-to-ditital converters or other specialized interfaces to the IMP-16, sophisticated process-control and instrumentation systems could be quickly and easily developed.

APPENDIX A

WIRE LIST


1A 74LS161
1B 74LS00
1C 74LS109
2A 74LS09
2B 74LS10
2C 74LS109
3A 74LS174
3B 74LS00
3C 74173
4A 74LS174
4B 74LS08
4C 74173
5A 74LS174
5B 7438
5C 74173
6A 7438
6B 7438
6C 74173
7A 7438
7B 7438
7C 8836
8A 7400
8B 74LS138
8.5C
9A 1Kn DIP Res.
9B 74LS20

A1

Pin	to	to	to
1	C1-1		
2	A2-5		
3			
4			
5			
6			
7	B4-3		
8	GND		
9	B1-3		
10			
11			
12			
13			
14			
15	A2-3		
16	+5		

A2

Pin	to	to	to
1	BR2		
2	B2-3		
3	A1-15		
4	B2-11	B1-2	
5	B4-8	A1-2	
6	B3-2		
7	A2-8		
8	GND	A2-7	
9			
10	B1-4	B3-1	
11	BV2		
12	B5-1	B1-5	
13	BS2		
14	B1-12	B2-5	
15	BU2		
16	+5		

A3

Pin	to	to	to
1	A4-1		
2	A6-1		
3	BF2	C3-3	
4	BF1	C3-4	
5	A6-4		
6	BE2	C3-5	
7	A6-15		
8	GND		
9	A4-9		
10	A6-12		
11	BE1	C3-6	
12			
13			
14			
15			
16	+5		

A4

Pin	to	to	to
1	A3-1	A5-1	
2	A7-15		
3	BK2	C5-5	
4	BK1	C5-6	
5	A7-13		
6	BJ2	C4-3	
7	B6-1		
8	GND		
9	A3-9	A5-7	
10	B6-4		
11	BJ1	C4-4	
12	B6-15		
13	BH2	C4-5	
14	BH1	C4-6	
15	B6-12		
16	+5		

A5

Pin	to	to	to
1	A4-1	A9-1	
2	B7-1		
3	BN2	C6-3	
4	BN1	C6-4	
5	B7-4		
6	BM2	C6-5	
7	B7-15		
8	GND		
9	A4-9		
10	B7-12		
11	BM1	C6-6	
12	A7-1		
13	B62		
14	B61		
15	A7-4		
16	+5		

A6

Pin	to	to	to
1	A3-2		
2	A6-5	A6-14	
3	AD2		
4	A3-5		
5	A6-2	B6-2	B6-2
6	AC1		
7	A6-8		
8	GND	A6-7	
9			
10	AB1		
11	A6-14	B6-14	
12	A3-10		
13	AB2		
14	A6-2	A6-11	
15	A3-7		
16	+5		

A7

Pin	to	to	to
1	A5-12		
2	A7-5	A7-14	A8-13
3	AJ2		
4	A5-15		
5	A7-2	B7-2	
6	AH1		
7	A7-8		
8	GND	A7-7	
9			
10	AF1		
11	A7-14	B7-14	
12	A4-5		
13	AH2		
14	A7-2	A7-11	
15	A4-2		
16	+5		

A8

Pin	to	to	to
1	C7-14	A8-2	
2	A8-1		
3	L2		
4	A2-3	A8-5	
5	A8-4		
6	61		
7	A8-8		
8	GND	A8-7	
9			
10	A6-5		
11	A8-12		
12	A8-11	A8-14	C1-7
13	A7-2		
14	A8-15	A8-12	
15	A8-14		
16	+5		

B1

Pin	to	to	to
1	C2-10		
2	A2-4		
3	A1-9		
4	A2-10	B1-13	
5	A2-12		
6	C1-13		
7	GND		
8	C1-2		
9	C1-10		
10	B1-11		
11	B1-10		
12	A2-14		
13	B1-4		
14	+5		
15			
16			

B2

Pin	to	to	to
1	B3-11		
2	B3-8		
3	A2-2		
4	B2-10	B9-6	
5	A2-14		
6	C3-1		
7	GND		
8	C2-13		
9	B3-5	C1-6	
10	B2-4		
11	A2-4	B2-13	
12	C2-14		
13	B2-11	B4-2	
14	+5		
15			
16			

B3

Pin	to	to	to
1	A2-10		
2	A2-6		
3	A3-9		
4	A5-2	B3-13	
5	B2-9	B5-9	
6	BP2		
7	GND		
8	B2-2		
9	C2-6		
10	B2-5		
11	B2-1		
12	C7-14		
13	B3-4		
14	+5		
15			
16			

B4

Pin	to	to	to
1	B5-13		
2	B2-13		
3	A1-7		
4	B3-6		
5	A5-5		
6	B2-4		
7	GND		
8	A2-5		
9	B4-10		
10	B4-9	B51	
11	C3-7		
12	B4-13		
13	B4-12	BR1	
14	+5		
15			
16			

B5

Pin	to	to	to
1	B5-2	A2-12	
2	B5-1		
3	AM2		
4			
5			
6			
7	GND		
8	BU1		
9	B3-5		
10	C2-7		
11	A62		
12	C2-9	B5-13	
13	B5-12	B4-1	
14	+5		
15			
16			

B6

Pin	to	to	to
1	A4-7		
2	A6-5	B6-5	
3	AF2		
4	A4-10		
5	B6-2		
6	AE1		
7	B6-8		
8	GND	B6-7	
9			
10	AD1		
11	B6-14		
12	A4-13		
13	AE2		
14	A6-11	B6-11	
15	A4-12		
16	+5		

B7

Pin	to	to	to
1	A5-2		
2	A7-5	B7-5	
3	A61		
4	A5-5		
5	B7-2		
6	AK1		
7	B7-8		
8	GND	B7-7	
9			
10	AJ1		
11	B7-14		
12	A5-10		
13	AK2		
14	A7-11	B7-11	
15	A5-7		
16	+5		

B8

Pin	to	to	to
1	A5-7		
2	A5-5		
3	A5-2		
4	B8-5		
5	B8-4	B8-8	
6	B8-16		
7	C8.5-1		
8	GND		
9	C8.5-2		
10	C8.5-3		
11	C8.5-4		
12	C8.5-5		
13	C8.5-6		
14	C8.5-7		
15	C8.5-8		
16	+5		

C1

Pin	to	to	to
1	C1-11	A1-1	
2	B1-8		
3	C2-9		
4	C1-12		
5			
6	B2-9		
7	A8-12		
8	GND		
9			
10	B1-9		
11	C1-1	C2-1	
12	C1-4	C2-4	
13	C1-14	B1-6	
14	C1-13		
15			
16	+5		

C2

Pin	to	to	to
1	C1-11	C2-11	
2	B3-12		
3	C2-9	B5-12	
4	C1-12	B4-8	
5			
6	B3-9		
7	C3-10	B5-10	
8	GND		
9	C1-3	C2-3	
10	B1-1		
11	C2-1	B9-	
12	B4-11		
13	B2-8		
14	B2-12		
15			
16	+5		

C3

Pin	to	to	to
1	B2-6	C4-1	
2	C3-9		
3	A3-3		
4	A3-4		
5	A3-6		
6	A3-11		
7	B4-11	C4-7	
8	GND	C3-9	
9	C3-8	C3-15	
10	C2-7	C4-10	
11	DR0		
12	DR1		
13	DR2		
14	Dr3		
15	C3-9	C3-2	
16	+5		

C4

Pin	to	to	to
1	C3-1	C5-1	
2	C4-9		
3	A4-6		
4	A4-11		
5	A4-13		
6	A4-14		
7	C3-7	C5-7	
8	GND	C4-9	
9	C4-8	C4-15	
10	C3-10	C5-10	
11	DR4		
12	DR5		
13	DR6		
14	DR7		
15	C4-9	C4-2	
16	+5		

C5

Pin	to	to	to
1	C4-1	C6-1	
2	C5-9		
3	A5-13		
4	A5-14		
5	A4-3		
6	A4-4		
7	C4-7	C6-7	
8	GND	C5-9	
9	C5-8	C5-15	
10	C4-10	C6-10	
11	DR8		
12	DR9		
13	DR10		
14	DR11		
15	C5-9	C5-2	
16	+5		

C6

Pin	to	to	to
1	C5-1		
2	C6-9		
3	A5-3		
4	A5-4		
5	A5-6		
6	A5-11		
7	C5-7		
8	GND	C6-9	
9	C6-8	C6-15	
10	C5-10		
11	DR12		
12	DR13		
13	DR14		
14	DR15		
15	C6-9	C6-2	
16	+5		

C7

Pin	to	to	to
1	C7-8		
2			
3			
4			
5			
6			
7			
8	GND	C7-2	C7-12
9			
10			
11			
12	C7-8		
13	AM1		
14	B3-12		
15			
16			

C8,5

Pin	to	to	to
1	B8-7		
2	B8-9		
3	B8-10		
4	B8-11		
5	B8-12		
6	B8-13		
7	B8-14		
8	B8-15		
9			
10			
11			
12			
13	B9-1		
14	B9-2		
15	B9-4		
16	B9-5		

A9

Pin	to	to	to
1	B9-12		
2	B9-1		
3	B9-2		
4	B9-4		
5	B9-5		
6			
7			
8	GND		
9			
10			
11			
12			
13			
14			
15			
16	+5		

B9

Pin	to	to	to
1	C8.5-13	A9-2	
2	C8.5-14	A9-3	
3			
4	C8.5-15	A9-4	
5	C8.5-16	A9-5	
6	B2-4		
7	B9-8		
8	GND	B9-7	
9			
10	C2-1		
11	BP1		
12	A9-1	B9-14	
13			
14	B9-12	B9-15	
15	B9-14		
16	+5		

Pin	to	to	to
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			

Pin	to	to	to
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			

LIST OF REFERENCES

Digital Equipment Corporation. "PDP-11 Peripherals Handbook." Maynard, Massachusetts, 1975.

Korn, Granino A. "A Comparison of Interprocessor Communication Schemes," Computer Science Laboratory Report Number 210, University of Arizona, November 1974.

Korn, Granino A. "A Proposed Method for Simplified Microcomputer Programming," Computer, October 1975.

National Semiconductor. "IMP-16 Users Manual." Santa Clara, California, 1974.