

AD-A061 675

AIR FORCE AVIONICS LAB WRIGHT-PATTERSON AFB OHIO  
PDP-11 MICROCOMPUTER TEST STAND.(U)  
SEP 78 D J SCHILLER  
AFAL-TR-78-191

F/G 9/2

UNCLASSIFIED

NL

1 OF  
AD  
A061 675



LEVEL II (1)

AFAL-TR- 78-191

PDP-11 MICROCOMPUTER TEST STAND

SYSTEM TECHNOLOGY BRANCH  
SYSTEM AVIONICS DIVISION



SEPTEMBER 1978  
TECHNICAL REPORT AFAL-TR- 78-191  
FINAL REPORT FOR PERIOD 1 JULY 1975 - 31 MARCH 1978

Approved for public release; distribution unlimited.

AIR FORCE AVIONICS LABORATORY  
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES  
AIR FORCE SYSTEMS COMMAND  
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

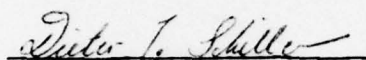


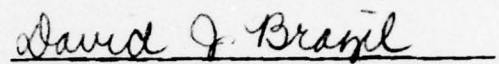
78 11 24 018

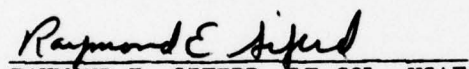
# NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

  
DIETER J. SCHILLER  
Project Engineer  
System Technology Branch

  
DAVID J. BRAZIL, CAPTAIN, USAF  
Tech Mgr, Software & Processor Gp  
System Technology Branch

  
RAYMOND E. SIFERD, LT COL, USAF  
Chief  
System Avionics Division

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

78 11 24 018

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM																
1. REPORT NUMBER AFAL-TR-78-191	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER																
4. TITLE (and Subtitle) PDP-11 MICROCOMPUTER TEST STAND		5. TYPE OF REPORT & PERIOD COVERED Final Report 1 July 75 - 31 March 78																
7. AUTHOR(s) Dieter J. Schiller		6. PERFORMING ORG. REPORT NUMBER																
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Avionics Laboratory (AAT-2) AF Wright Aeronautical Laboratories, AFSC Wright-Patterson Air Force Base, Ohio 45433		8. CONTRACT OR GRANT NUMBER(s)																
11. CONTROLLING OFFICE NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element 62204F Project 2003, Task 200304 Work Unit 20030412																
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Air Force Avionics Laboratory (AAT-2) AF Wright Aeronautical Laboratories, AFSC Wright-Patterson Air Force Base, Ohio 45433		12. REPORT DATE September 78																
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		13. NUMBER OF PAGES																
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		15. SECURITY CLASS. (of this report) Unclassified																
18. SUPPLEMENTARY NOTES		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE																
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Microcomputers Hardware Software PDP-11		<table border="1"> <tr> <td colspan="2">ACCESSION for</td> </tr> <tr> <td>NTIS</td> <td>White Section <input checked="" type="checkbox"/></td> </tr> <tr> <td>DOC</td> <td>Buff Section <input type="checkbox"/></td> </tr> <tr> <td>UNANNOUNCED</td> <td><input type="checkbox"/></td> </tr> <tr> <td colspan="2">JUSTIFICATION</td> </tr> <tr> <td colspan="2">BY</td> </tr> <tr> <td colspan="2">DISTRIBUTION/AVAILABILITY CODES</td> </tr> <tr> <td>Dist.</td> <td>AVAIL and/or SPECIAL</td> </tr> </table>	ACCESSION for		NTIS	White Section <input checked="" type="checkbox"/>	DOC	Buff Section <input type="checkbox"/>	UNANNOUNCED	<input type="checkbox"/>	JUSTIFICATION		BY		DISTRIBUTION/AVAILABILITY CODES		Dist.	AVAIL and/or SPECIAL
ACCESSION for																		
NTIS	White Section <input checked="" type="checkbox"/>																	
DOC	Buff Section <input type="checkbox"/>																	
UNANNOUNCED	<input type="checkbox"/>																	
JUSTIFICATION																		
BY																		
DISTRIBUTION/AVAILABILITY CODES																		
Dist.	AVAIL and/or SPECIAL																	
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The interface described in this report permits a Digital Equipment Corporation PDP-11 computer to control and monitor up to four separate microcomputers. The microcomputers execute their instructions out of the PDP-11 memory. Break point registers are included such that a print-out may be obtained of the micro-computer's register contents, etc., after execution of each instruction.																		



REPORT DOCUMENTATION PAGE		REPORT NUMBER	
1. REPORT NUMBER		AFAL-TR-78-191	
2. TITLE (and Subtitle)		PDP-11 MICROCOMPUTER TEST STAND	
3. TYPE OF REPORT & PERIOD COVERED		Final Report 1 July 75 - 31 March 78	
4. AUTHOR		Dieter J. Schiller	
5. PERFORMING ORG. REPORT NUMBER			
6. CONTRACT OR GRANT NUMBER(s)			
7. PERFORMING ORGANIZATION NAME AND ADDRESS		Air Force Avionics Laboratory (AAT-2) AF Wright Aeronautical Laboratories, AFSC Wright-Patterson Air Force Base, Ohio 45433	
8. CONTROLLING OFFICE NAME AND ADDRESS			
9. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS		Program Element 6230AF Project 2003, Task 200304 Work Unit 20030412	
10. REPORT DATE		8 September 78	
11. NUMBER OF PAGES			
12. SECURITY CLASS. (of this report)		Unclassified	
13. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		Air Force Avionics Laboratory (AAT-2) AF Wright Aeronautical Laboratories, AFSC Wright-Patterson Air Force Base, Ohio 45433	
14. DISTRIBUTION STATEMENT (of this Report)			
Approved for public release; distribution unlimited.			
15. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)			
16. SUPPLEMENTARY NOTES			
17. KEY WORDS (Location on cover sheet if necessary and identify by block number)			
Microcomputers Hardware Software PDP-11			
18. ABSTRACT (Continue on reverse side if necessary and identify by block number)			
The interface described in this report permits a Digital Equipment Corporation PDP-11 computer to control and monitor up to four separate microcomputers. The microcomputers execute their instructions out of the PDP-11 memory. Break point registers are included such that a print-out may be obtained of the micro-computer's register contents, etc., after execution of each instruction.			

## FOREWORD

This report describes an in-house effort conducted by personnel of the System Technology Branch (AAT-2), System Avionics Division (AA), Air Force Avionics Laboratory, Air Force Wright Aeronautical Laboratories, Wright-Patterson Air Force Base, Ohio, under Project 2003, "Avionic System Design Technology", Task 200304, "Avionic Processing System Design", Work Unit 20030412, "PDP-11 Microcomputer Test Stand".

The work reported herein was performed during the period 1 July 1975 to 31 March 1978, under the direction of the author, Mr. Dieter Schiller (AFAL/AAT-2), project engineer. The report was released by the author in September 1978.

The author wishes to thank Mr. Alfred J. Scarpelli (AFAL/AAT-2) for his assistance in the software development and testing phase of the program.

This report is intended to be a users manual as well as a final report.

## TABLE OF CONTENTS

SECTION		PAGE
I	INTRODUCTION	1
	1.1 Purpose	1
	1.2 Background	2
	1.3 Objective	3
	1.4 Results	4
	1.5 Interface Description	4
	1.6 Modes of Operation	6
	1.7 Interface Functions	6
II	HARDWARE ORGANIZATION	8
	2.1 Register Description	8
	2.2 Hardware Operations	13
III	SOFTWARE/HARDWARE INTERACTION	20
	3.1 Start-Up Interrupt	20
	3.2 Loading The Interface Registers	22
	3.3 No Trace Read	26
	3.4 No Trace Write	29
	3.5 Trace Read	31
	3.6 Trace Write	33
IV	CONCLUSION	36
	APPENDIX A	37
	APPENDIX B	40
	B.0. Requirements	41
	B.1 Signal Definitions	42

# LIST OF ILLUSTRATIONS

FIGURE		PAGE
1	System Block Diagram	5
2	Interface Registers	8
3	Control Register	8
4	Status Register	11
5	Data Out Register	13
6	Start-Up Interrupt Timing	14
7	No Trace Read Timing	15
8	No Trace Write Timing	16
9	Trace Real Timing	18
10	Trace Write Timing	19
11	Start-Up Interrupt Flowchart	20
12	No-Trace Read Flowchart	27
13	No-Trace Write Flowchart	30
14	Trace Read Flowchart	32
15	Trace Write Flowchart	35
16	Schematic of Interface	38
B.1	I-Bus Signal List	51
B.2	I-Bus Control & Data Transfer Timing	53
B.3	I-Bus Interrupt Interactions with Normal Bus Control and Data Transfers	54
B.4	CLEAR-RESET Timing	55
B.5	TRQ Detail	56



## SECTION I

### INTRODUCTION

#### 1.1 Purpose of This Report.

This technical report serves the dual function of providing a formal report on the work done under Work Unit 2003-04-12, Distributed Microcomputer Network Test Stand, and a User's Manual for the hardware and software developed on this effort.

In order to be able to operate the interface, the user must familiarize himself with the PDP-11 assembly language and the DR-11C general purpose interface module. Although the interface was designed for a PDP-11/20, any of the PDP-11 family computers may be used as the system monitor. The documentation needed by the user, other than this report is the 'Processor Handbook' and the 'Peripherals Handbook'. Both of these documents are paperbacks and are available from the Digital Equipment Corporation, Maynard, MA.

Besides the PDP-11, the user must also thoroughly understand the I-Bus. The I-Bus is the internal bus used by each of the microcomputers (See Figure 1). It consists of 38 data transfer lines (16 of which are data and 16 are address lines), 3 interrupt control lines and 3 bus control lines. The bus control lines are not used by this interface. For a detailed I-Bus description see Appendix B.

This report is organized into three sections. Section I discusses the background, purpose and functions of the control panel. In Section II the hardware is described in detail including all of the internal registers. Section II presents the software/hardware interface and includes the PDP-11

code necessary to perform each of the interface functions described in Section I. In addition to the three sections, there are two appendices. Appendix A contains the gate level diagram of the interface while Appendix B consists of the I-Bus description.

## 1.2 Background.

The original objective of this work unit was to provide computer performance monitoring and control hardware and software for the four micro-computer breadboard to be developed on Work Unit 2003-04-13, Distributed Microcomputer Network for Avionics. Three PDP-11 interfaces were planned, which were to perform the following functions:

- (1) Individual microcomputer control,
- (2) Global and local intercomputer bus traffic monitoring,
- (3) Simulated I/O injection into the breadboard microcomputers.

The associated support software to control these interfaces and allow user interaction with the system was included. After this effort was initiated, the D&F for Work Unit 2003-04-13 was disapproved. In order to still demonstrate the concept of the Distributed Microcomputer Network for Avionics, work unit 2003-04-11 entitled Distributed Processor/Memory Simulation Experiments and work unit 2003-04-14 called DMNA Bus Monitor Interface were created instead of using actual microcomputers for the Distributed Micro-computer Network, laboratory PDP-11's were to be used. The purpose of WU 14 was to develop three bus-control interface units which were to provide the communication channel developed for the DMNA between three PDP-11's.

The software for the avionic computers was developed under wu 2003-04-11. The local executive, global executive, fault-detection/recovery and test

programs were first written in the microcomputer's instruction set and thoroughly debugged in AFAL's DEC-10 instruction level simulator. After successful operation in the DEC-10, the programs were translated into PDP-11 assembly language.

The contractor who was to build the bus-control interface units for the PDP-11's (Work Unit 2003-04-14) ran out of funds before the project could be completed and due to no additional funds being available for this effort, work was halted. The actual laboratory demonstration for WU 2003-04-11 and 14 could not be done since the necessary hardware was never completed.

The concepts evolved during the course of this effort provided a departure point for specifying a baseline set of monitoring system features for Work Unit 2003-04-17, "Design of a User-Oriented Microcomputer and Monitoring System for Avionics Application" as specified in Appendix B of that Statement of Work.

### 1.3 Objective.

Design, build, demonstrate and document a PDP-11 interface that



will allow limited (primarily non-real-time) computer performance monitoring and control (CMAC) avionic microcomputers utilizing a single-bus-oriented internal hardware architecture, specifically the I-Bus described in Appendix B.

#### 1.4 Results

The interface described in the remainder of this report was designed, constructed, tested, and successfully demonstrated. The hardware consists of 4 cards of logic of about 150 IC's, packaged in wirewrap form. A set of assembly language routines were written to realize the functions listed in Paragraph 1.6.

The demonstration of the interface was done using a combination of digital logic, switches, and signal generators to simulate the missing microcomputer's I-Bus interface functions. Each function was exercised in this manner.

#### 1.5 Interface Description.

The interface developed on this effort allows the PDP-11 software to perform the function of a traditional computer control panel. With the exception of breakpoint register functions, the microcomputer must be halted or slowed down while the interface is performing its functions, hence it is essentially a non-real-time device. The interface can control up to four separate microcomputers via multiplexing hardware. Which microcomputer is being controlled at any given time is selected under software control. The microcomputers, themselves, may all be the same, or they may be different. The only requirement is that each microcomputer conforms to the internal bus (I-Bus) specifications described in Appendix B.



An interesting feature of the interface is that it allows the microcomputers to use the PDP-11 memory as a simulation of its own memory. Hence, only a microcomputer CPU is necessary for the interface to operate. Also, simulated input can be transmitted to the microcomputer and output received. Figure 1 is the block diagram for the control panel interface.

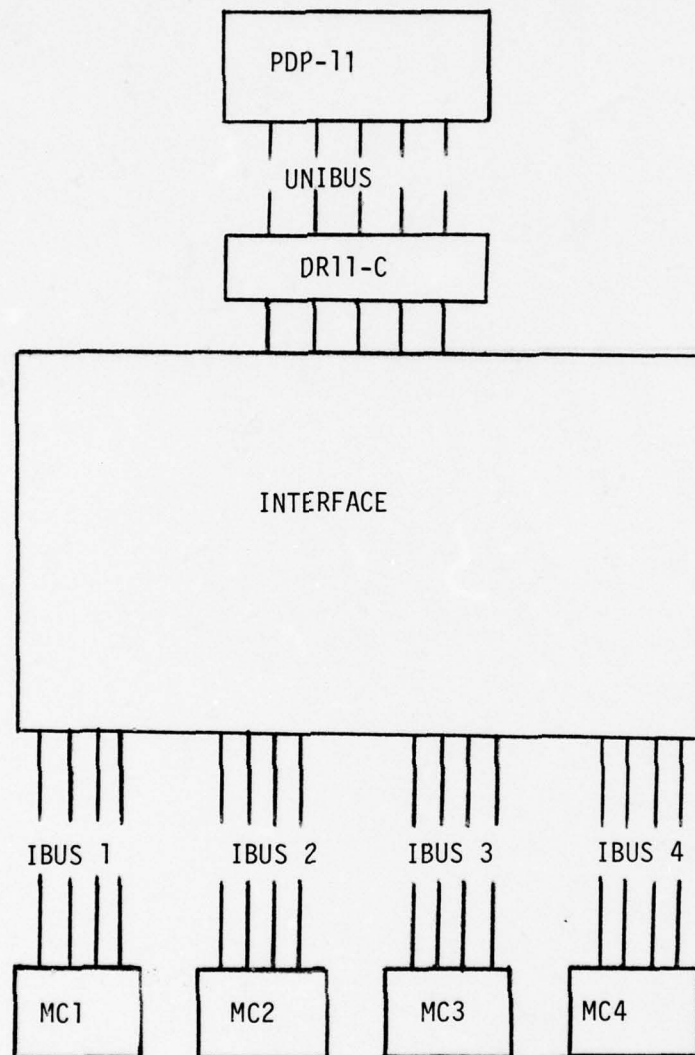


Figure 1  
System Block Diagram

### 1.6 Modes of Operation.

Two modes of operation are possible. The first is the free-running mode. Here, the microcomputer gets a starting address from the PDP-11. The microcomputer will load this value into its program counter and execute the program. Upon program completion only the final results will be available.

The second mode of operation is the trace mode. This feature permits the microcomputer to operate in a pseudo-real-time fashion. In order to invoke the trace function, the programmer must load the lower break point register with the lower address bound and the upper break point register with the upper address bound.

It is possible to obtain a trace of the entire program or of only a selected part.

### 1.7 Interface Functions.

This control panel interface is capable of performing 13 distinct tasks. These operations are as follows:

- a. Load upper break point register.
- b. Load lower break point register.
- c. Load control register.
- d. Read status register.
- e. Issue a microcomputer startup interrupt.
- f. Read address lines.
- g. Read data lines
- h. Read master ID lines
- i. Select one out-of-four microcomputer.

- j. Perform a read operation without tracing.
- k. Perform a read operation with tracing.
- l. Perform a write operation without tracing.
- M. Perform a write operation with tracing.

NOTE: The last four tasks consists of a combination of the features listed in (a) through (i).

## SECTION II

### HARDWARE ORGANIZATION

#### 2.1 Register Description.

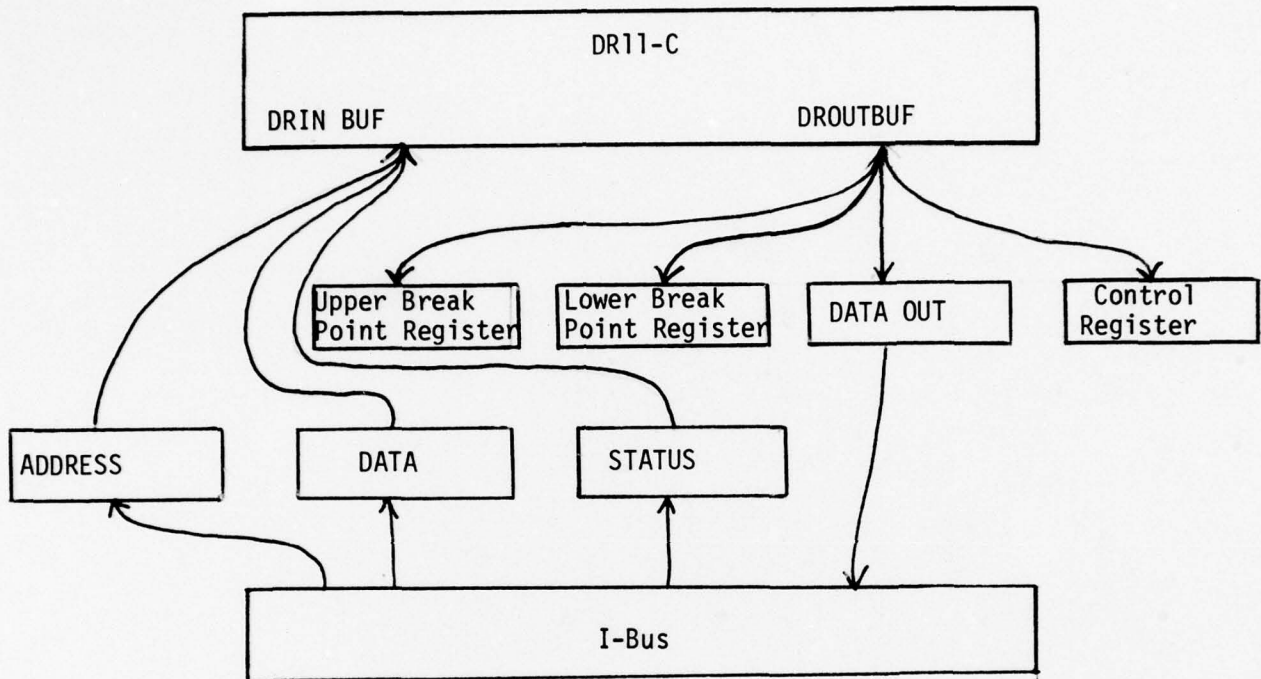


Figure 2  
Interface Registers

##### 2.1.1 Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACKEN			CRINT	CRBPE		CRINHB		MUX		CRTACKI	write		CRTACKM	read	

Figure 3  
Control Register



2.1.1.1 Read: Bits "0" and "1" determine which interface register gets placed on the DR11-C data-in lines. The truth table indicates which register gets read.

Bit 1	Bit 0	Register
0	0	Interface status
0	1	Address
1	0	Data
1	1	MID

Note: Only the four least significant bits of the MID are used. The upper twelve bits are meaningless during a "read Master ID" operation.

#### 2.1.1.2 CRTACKM (Control Register Transfer Acknowledge Memory).

Control register bit "2" is used during a read or write operation.

2.1.1.3 Write: Bits "3" and "4" determines which interface register is to be written into from the DR11-C output buffer.

Bit 4	Bit 3	Register
0	0	Data Out
0	1	Break point #1 (lower)
1	0	Break point #2 (upper)
1	1	Unused

#### 2.1.1.4 CRTACKI (Control Register Transfer Acknowledge Interrupt).

Bit "5" of the control register is used during a PDP-11 to microcomputer start-up interrupt procedure.

#### 2.1.1.5 MUX.

Bits "6" and "7" determine which one of four microcomputers the PDP-11 "talks to".

Bit 7	Bit 6	Microcomputer #
0	0	1
0	1	2
1	0	3
1	1	4

#### 2.1.1.6 CRBPE (Control Register Breakpoint Enable).

Bit "11" determines whether the "trace" function is active (11 or not 10).

#### 2.1.1.7 CRINT.

Bit "K" is used to send an interrupt to one of the microcomputers.

#### 2.1.1.8 TACKEN.

Enables the transfer-acknowledge signal.

#### 2.1.1.9 Unused Bits.

Control register bits 8,9,10,13 and 14 are unused at this time.

#### 2.1.2 Status Register.

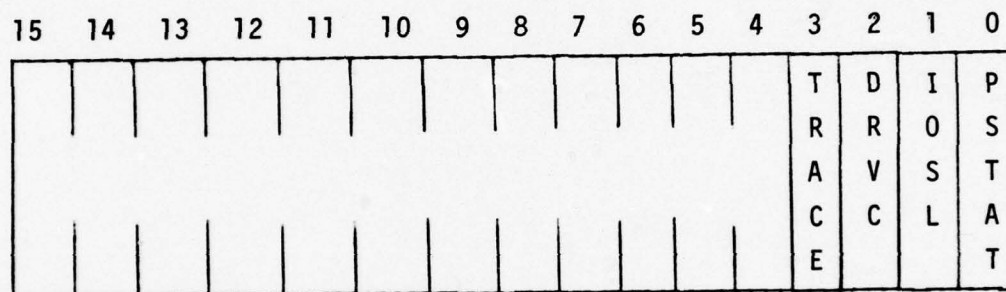


Figure 4  
Status Register

Only bits 0-4 are used in the status register. Their meaning is as follows:

##### 2.1.2.1 PSTAT

PSTAT (Bit 0) in connection with the data transfer signals indicates the state that the processor is in and the nature of the data and address line.

##### 2.1.2.1.1 Program Counter and Instruction Register.

If the processor is receiving data (DRCV is true (L) ) and PSTAT is true (L), then the address lines have the PC on them and when TACK comes true, the information on the data lines is the first word of the instruction.

#### 2.1.2.1.2 Interrupt State.

If the processor is sending data (DRCV is false (H) and PSTAT is true (L), then the processor is saving its program counter and status word. The data on this address lines is the interrupt stack pointer and the first word sent, when the processor status (PSTAT) and transfer request (TRQ) comes true the first time, is the old PC. The second word sent, when the PSTAT and TRQ come true the second time, is the old status word.

#### 2.1.2.2 IOSL (BIT 1).

This bit tells if a memory or I/O operation is taking place.

#### 2.1.2.3 DRCV (Bit 2).

The "data receive bit indicates in which direction the information transfer is to take place.

If DRCV is low (true) a SENB operation is taking place. The master shall send 16 bits of data and store it at the address location specified by the address lines.

If DRCV is high, then a RECEIVE operation is in progress.

#### 2.1.2.4 TRACE (Bit 3).

The "trace" bit indicates whether or not the break-point register received an address match. Note, that the address must have been placed on the address lines by a valide"master".

#### 2.1.3 Data Out Register.



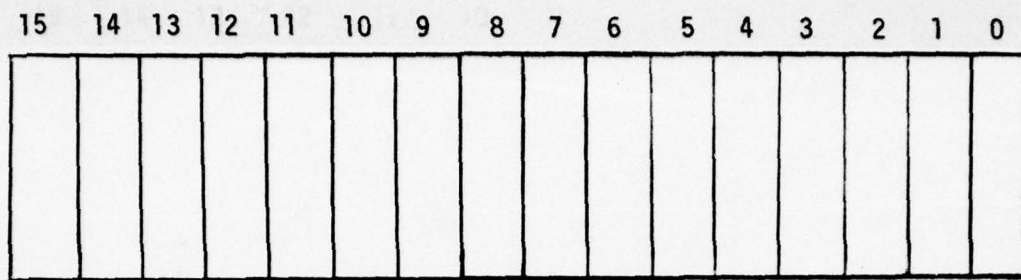


Figure 5  
Data Out Register

The Data Out Register is used to transfer data from the PDP-11 to the microcomputers.

#### 2.1.4 Upper/Lower Break-Point Register.

These two registers are functionally identical. Each register gets loaded with a 16 bit address. The lower BPR contains the lower address space bound, while the upper BPR contains the upper bound. If the bit is enabled, then the tracing functions will occur whenever an address falls between the upper and lower limit of the break point registers.

#### 2.1.5 Address Register.

The address register is a 16-bit register which contains the PDP-11 address which the microcomputer wants to access.

### 2.2 Hardware Operation

#### 2.2.1 Start-Up Interrupt.

Whenever the PDP-11 wishes to send a start-up interrupt to any one of the four microcomputers (determined by the MUX bit) then the programmer must set the appropriate bits in the control register which

in turn, causes the IRQ signal to be asserted. As soon as the microcomputer is ready to honor the interrupt, it will assert the IACKI (interrupt-acknowledge-in) signal and remove IRQ. After the PDP-11 sees that the microcomputer has acknowledged the interrupt request it will place the trap vector location on the 'data out' lines and then raise the transfer acknowledge (TACK) signal. The microcomputer will take the trap vector address off the data lines and remove IACKI.

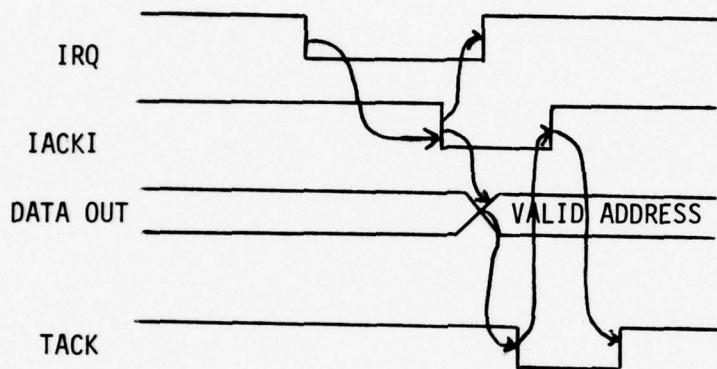


Figure 6

#### Start-Up Interrupt Timing

Upon removal of the IACKI signal the interface clears TACK and the system is ready for another operation.

The start up interrupt is used to send the beginning address of a program to the microcomputer. The microcomputer will return the starting address, place it into its program counter and then start executing from that location until a return from interrupt instruction is encountered. During the trace function, the start up interrupt is used to invoke the trace routine and upon its completion control will be returned to the interrupted program.

### 2.2.2 No Trace Read (Read).

Whenever the microcomputer wants to obtain data from memory (The PDP-11 in this case), it places the desired address on the address lines and then asserts TRQ (transfer request). The transition of the TRQ signal causes an interrupt in the PDP-11. Next the software determines whether a read, write and trace operation is desired.

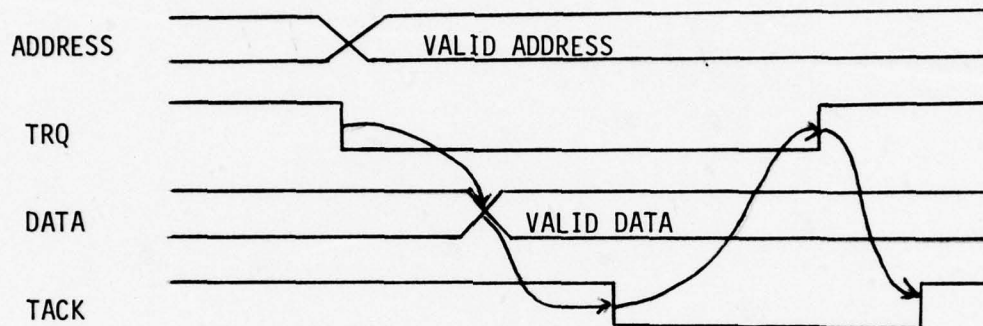


Figure 7

No Trace Read Timing

In this case the PDP will fetch the data at the desired memory location and place it on the data lines. After the data has settled, the interface issues TACK (transfer acknowledge). As soon as the microcomputer detects the asserted TACK, it takes the data off the data lines and then removes TRQ. The clearing of TRQ causes TACK to be restored, thereby completing the cycle.

### 2.2.3 No Trace Write.

The interface action for this type operation is very similar to the procedure described in 2.22. Whenever the microcomputer wants to perform a "write" operation it places the address, where the information is to be written, on the address lines and the data on the data lines. After a wait time of approximately 250 ns, the microcomputer then asserts the TRQ signal. Again, as in 2.22, interrupt is caused at the PDP-11 which will then read the status register to determine what type of operation is to be performed. Since a write operation is called for the PDP will read both the address register and the data register. After interrogation of these two registers PDP software performs the write operation. When this is completed then the TACK signal gets asserted. After receiving TACK, the microcomputer removes TRQ which is term is used to restore TACK.

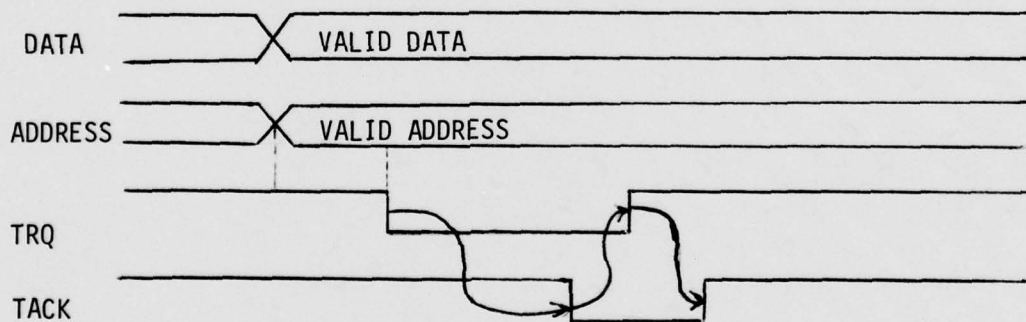


FIGURE 8  
NO TRACE WRITE TIMING



#### 2.2.4 TRACE READ:

If tracing is desired, then the appropriate bits in the control register must be enabled. The microcomputer places the address from where it wishes to get information on the address lines and then pulls down the TRQ signal. Since tracing is enabled the address on the address lines gets compared to the upper and lower break point register contents. If it falls within the bounds of these registers then the comparators fire and cause an interrupt in the microcomputer. The microcomputer will not honor the interrupt until the present instruction is completed.

As soon as TRQ goes low, the PDP gets interrupted and the software determines what action to take depending on the condition of the status registers. Since we are talking about a trace-read operation, the PDP will take the address off the address lines, read the appropriate PDP-11 memory location and place that information on the data lines. After a 250 ns settling time (actually much greater since done in software) the PDP causes the TACK signal to be asserted. The microcomputer takes the data off the data lines and then removes the TRQ, which in turn releases TACK. The upward transition of TACK signifies the end of the read cycle.

The processor will now honor the interrupt that was set by asserting IACKI. As soon as the interface detects IACKI, it removes IRQ. The PDP responds to IACKI by placing the trap vector location on the data lines and then asserting the TACK. As soon as the processor detects TACK it grabs the information off the data lines and removes IACKI. This action resets TACK and thus completes the interrupt. The microcomputer will place the trap vector location into its program counter and then proceed with manual data transfer operation.

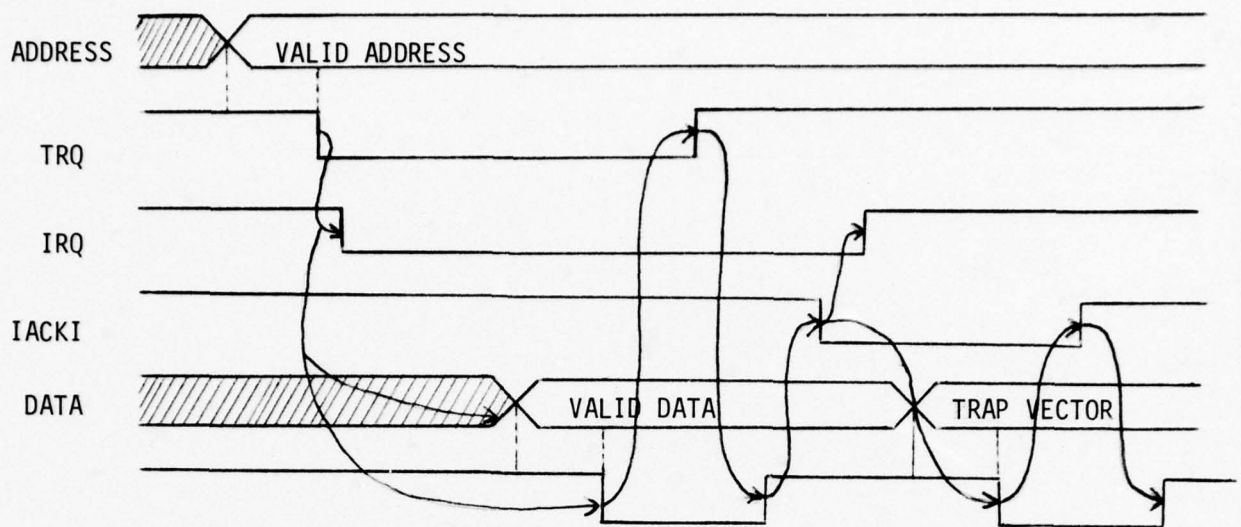


Figure 9  
Trace Real Timing

#### 2.2.5 Trace Write.

On a write operation the microcomputer places the address on the address lines and the data on the data lines prior to assertion TRQ. As soon as TRQ goes low, the comparator fires (during a trace operation) and the interface asserts the interrupt request line (IRQ). The microcomputer will not respond to this request until the write operation is completed. Once the PDP-11 detects the TRQ signal, it reads the interface status register and determines the type of operation to be performed. Since we are talking about a write operation the PDP must fetch both the information on the address lines and that on the data lines. The PDP software then writes the data into the desired memory location. As soon as that is completed the PDP sends the TACK signal

causes the TACK to be released. Since the write operation is now completed, the microcomputer will honor the interrupt by asserting IACK. The PDP will respond by placing the trap vector address on the data lines and then asserting TACK. When the microcomputer sees TACK it takes the data off the data lines and then removes IACK which in turn resets TACK and thereby completes the operation.

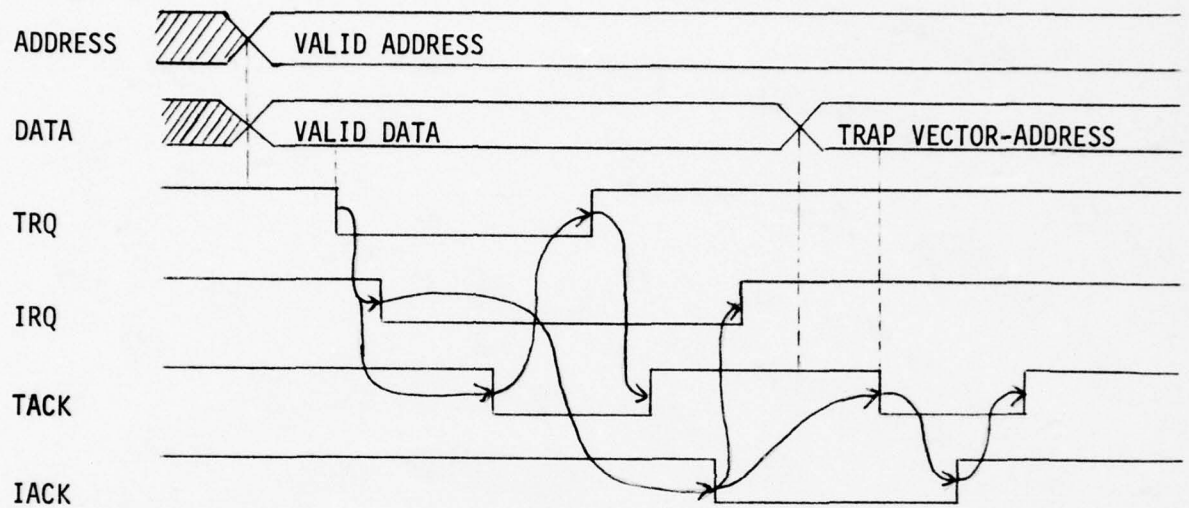


Figure 10  
Trace Write Timing

SECTION III  
SOFTWARE/HARDWARE INTERACTION

3.1 Start-Up Interrupt.

The following flowchart describes the necessary sequence of events to send a start-up interrupt to a microcomputer.

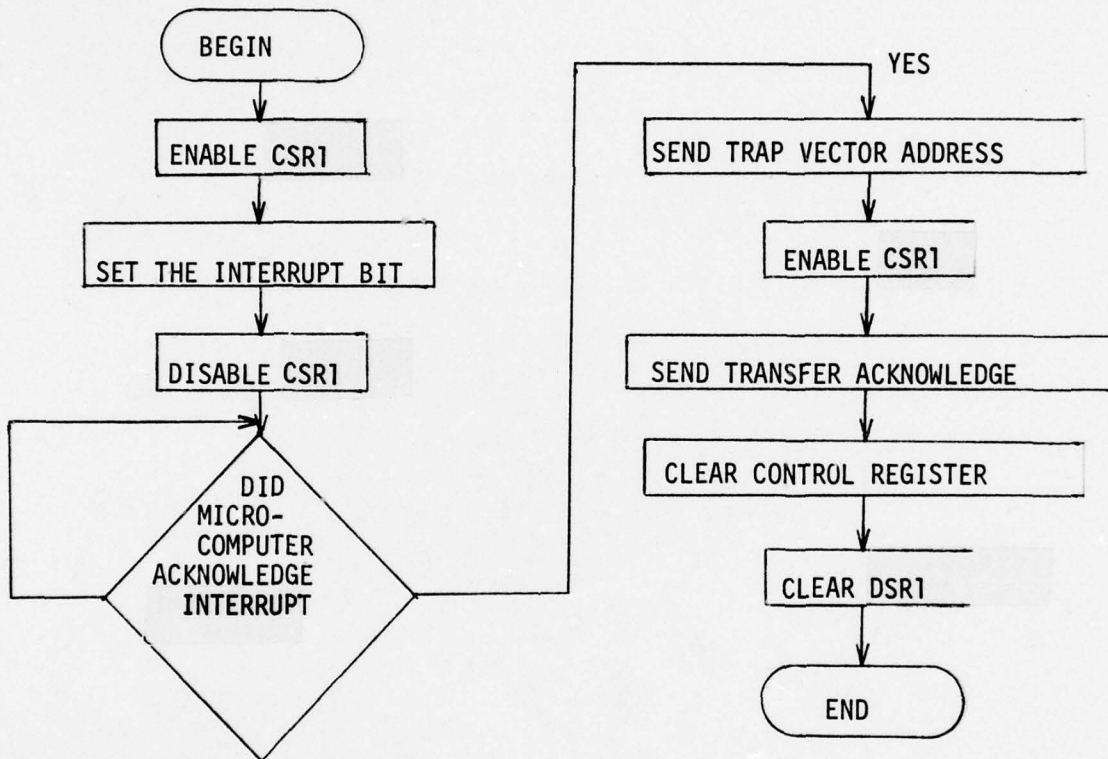


Figure 11  
Start-Up Interrupt Flowchart



### 3.1.1 Program For Start-Up Interrupt.

	MOV #2 DRCSR	Enable DSR1
	MOV #11030, DROUTBUF	Cause start-up interrupt
	MOV #0, DROUTBUF	Clear DROUTBUF
	CLR DRCSR	Disable DSR1
Loop:	MOV DRCSR, R1	Read Control & Status register
	BIC #77777, R1	Check if REQ B came in
	BGE LOOP	No
	MOV # ADDR, DROUTBUF	Set up control register
	MOV #2, DRCSR	Enable DSR1
	MOV #1040, DROUTBUF:	Set control register to enable CRTAKI, CLEAR REQUEST B
	MOV #100040, DROUTBUF	Set enable bit
	MOV #30, DROUTBUF	Clear CR and stop data from being written in DOR
	CLR DRCSR	Disable DSR1

### 3.1.2 Program Description.

Inst. 1 - The first instruction opens the data path for the DROUTBUE of the DRIIC to the interface control register.

Inst. 2 - Loads the control registers with 11030<sub>0</sub>. This bit configuration causes the interface to generate and IRQ (interrupt request) signal.

Inst. 3 - This instruction clears the control register. It is a good practice to always clear this register after an operation since overwriting data might cause a momentary race condition which will lead to improper interface operation. Note that 00 in the write bit causes the next word to be written into the DR OUTBUF to be loaded into the interface DATA OUT Register as long as DRCSR remains cleared.

Inst. 4 - In order to prevent further data from being written into the control register it is necessary to disable DSR1

Inst. 5., 6., & 7. - Contents of the DRCSR get interrogated to check if a request B (interrupt acknowledge) occurred. The program stays in a loop until the request B goes hi.

Inst. 8 - The trap vector location gets written into DROUTBUF and as soon as the NDR pulse arrives it gets latched into the DATA OUT Register.

Inst. 9 - Enable DSR1 and get ready to write into the control register.

Inst. 10 - The value is now written into the control register. This data places a "1" on the D input of the flip-flop which causes the transfer acknowledge signal for interrupts.

Inst 11. - The Mov #100040, DROUTBUF instruction writes the value 100040 into the control register which causes the flip flop FF5 to get loaded with the value on the 'D' input. Since that value is a '1', the interface generates a transfer acknowledge signal (TACK).

Inst. 12. - This instruction clears the control register. Note that the value 3 in the write bits is not used to direct the data from DROUTBUF to any interface register.

Inst. 13. - Disables DSR1. The start-up interrupt procedure is now complete.

### 3.2 Loading The Interface Registers

#### 3.2.1 Control Register.

To load the control register takes two steps. First a Mov #2, DRCSR operation is required which opens up the data path from the DR11C

buffer to the control register. The second instruction is a MOV #XXXXX, DROUTBUF. The execution of this instruction places the value XXXXX into the control register.

### 3.2.2 Upper Break Point Register

Loading this register takes two more steps than in the case addressed in 3.2.1. The first instruction is again a MOV #2, DRCSR. Rather than sending the actual data, the write bits in the control register must first be set so the next data written into the DR11-C output data register will get clocked into the upper break point register. This is accomplished with a MOV # 020, DROUTBUF. Before the correct value can be loaded into the breakpoint register, a CLR DRCSR instruction must be issued. This is necessary since its omission would continuously load the control register with the information written into the DROUTBUF by the program. The final instruction in the loading sequence is MOV #XXXXX, DROUTBUF where XXXXX is the value to be loaded into the upper BPR.

### 3.2.3 Lower Break Point Register

The program sequence is identical to the one described in 3.2.2 except that the second instruction loads a 010 into the control register instead of a 020. The program necessary to load the lower break point register is:

```
MOV #2, DRCSR
MOV #010, DROUTBUF
CLR DRCSR
MOV #XXXXX, DROUTBUF
```

#### 3.2.4 Data Out Register.

The following program is needed to load the DATA OUT register.

```
MOV #2 DRCSR
MOV #0, DROUTBUF (needed if CR did not contain a '0')
CLR DRCSR
MOV # XXXXX, DROUTBUF
```

The only difference between this code and the UPPER/LOWER break point register loading sequence is the value being stored into the control register. In order to route the data from DROUTBUF to the DATA OUT register a "0" is required in the write bits.

#### 3.2.5 Master Clear.

The master clear signal resets all interface registers to zero. It is usually issued when the interface is first turned on or if the operator desires to reset the interface. The code necessary for a master clear consists of one instruction. It is MOV #3, DRCSR. The execution of this instruction causes a one-shot to fire which will then clear all of the interface registers.

#### 3.2.6 Read Status Register.

A status register read operation is necessary in order for the PDP-11 to determine what type of operation the microcomputer desires. The following sequence is required in order to transfer the information from the status register to the PDP-11.

MOV #2 DRCSR	enable control register
MOV # 030, DROUTBUF	set the read bit to 0
CLR DRCSR	prevent further data from being written into CR
MOV DRINBUF, RX	read the contents of the status register into any PDP-11 register.



### 3.2.7 Read Address Lines.

Whenever the microcomputer desires to write into PDP memory or read from PDP memory, the PDP-11 must obtain the address location that the microcomputer wants to access. This is done in the following manner.

MOV #2, DRCSR	enables control register
MOV #1, DROUTBUF	set the read bits to 01
CLR DRCSR	
MOV DRINBUF, RX	read the information on the address lines into any PDP-11 register.

### 3.2.8 Read Data Lines.

This operation is needed when the microcomputer wants to write into memory. In addition to interrogating the address lines, the PDP must also obtain the information on the data lines so that it can perform the write operation. Again, the only difference between this read operation and the others is the value in the "Read" bits.

```
MOV #2 DRCSR
MOV #32, DROUTBUF
CLR DRCSR
MOV DRINBUF, RX
```

### 3.2.9 Read MID.

The MID lines (master identification) tells the PDP whether a processor is requesting service, or some peripheral device such as a bus control interface unit. The sequence of operation is the same with the exception that both read bits are set to "1".

```
MOV #2, DRCSR
MOV #33, DROUTBUF
CLR DRCSR
MOV DRINBUF, RX
```

### 3.3 No Trace Read.

Whenever a microcomputer wants to read data out of memory, then the Processor asserts the transfer request signal (TRQ). At the same time, it sets the DRVC (Data receive) and the IOSL (Input/output select) lines to indicate a read operation and places the address on the address lines. The transition of TRQ causes the address to get clocked into the interface address register and also generates the request A signal which tells the PDP-11 that the microcomputer needs servicing. The request A signal sets a flip flop which, in turn, sets bit number 7 in the DRCSR register. The PDP-11 program continuously monitors the status word to see if the request A bit gets set. The following program will perform a read operation. (See Figure 12).

```
LOOPA: MOV DRCSR, R0          read interface status register
        BIC #177577, R0       mask off all bits except request A
        BEQ LOOPA             if not set, then wait
        MOV #1, DRCSR         clear request A
        CLR DRCSR             reset DSRO
```

NOTE: The MOV #1, DRCSR instruction causes a '1' to be written into bit 0 of the status register (CSRO). This bit is connected through an inverter to the CLEAR input of the REQUEST A flip flop. The execution of the MOV instruction clears the flip flop and prevents it from being reset until the CLR DRCSR instruction has been issued.

The PDP-11 code, up to this point detects a request A condition and resets the status bit. The next operation is to determine what type of operation is to be performed. This is done in the following manner:

```
MOV #2, DRCSR          enable control register
MOV #30, DROUTBUF      get ready to read the status register
MOV DRINBUF, R0        place the status register contents into R0
CLR DRCSR
```

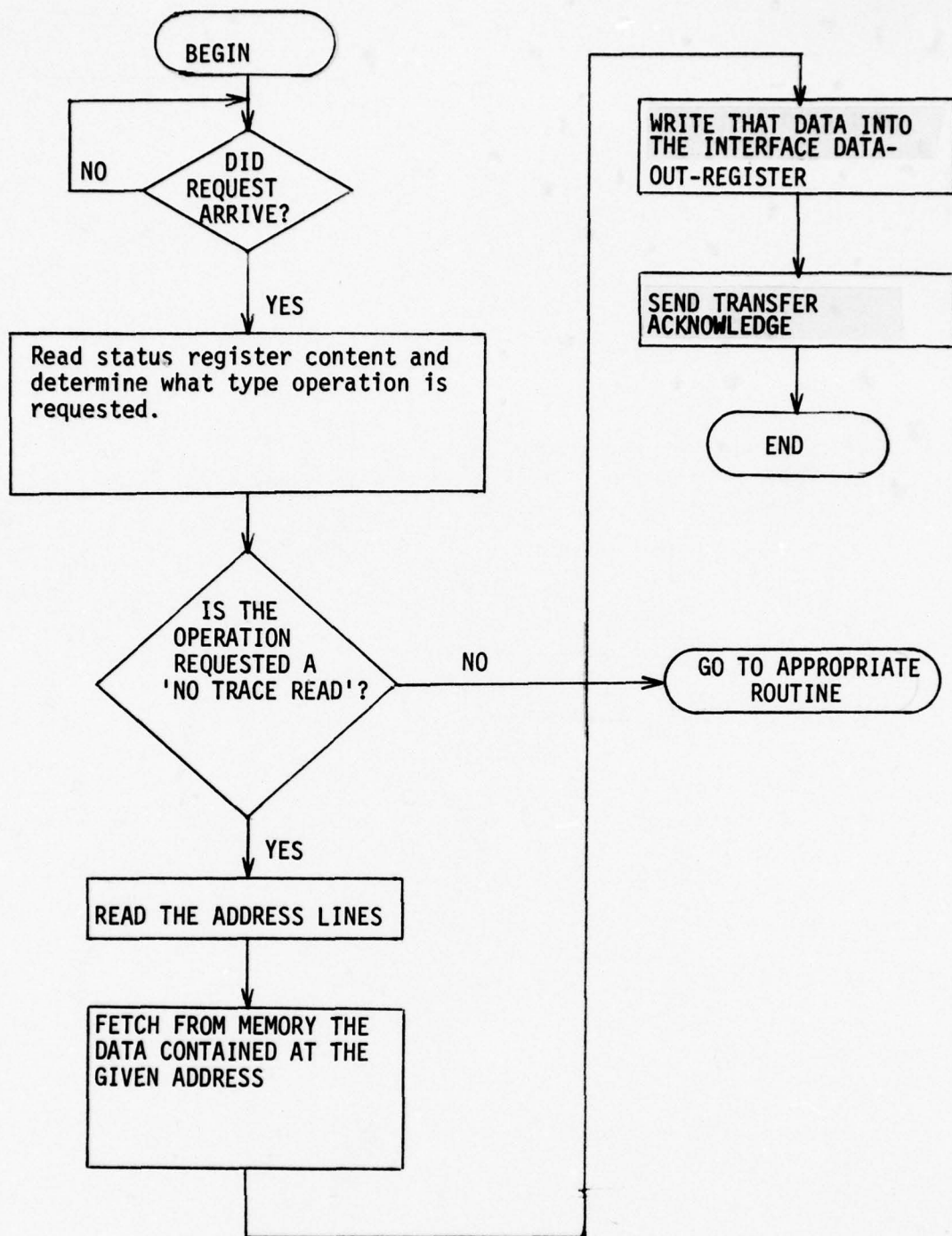


Figure 12  
No-Trace Read Flowchart

The PDP-11 code, up to this point detects a 'request A' condition and resets the status bit. The next operation is to determine what type of operation is to be performed. This is done in the following manner:

MOV #2, DRCSR	enable control register
MOV #31, DROUTBUF	get ready to read status register
MOV DRINBUF, R0	place the status register contents into R0
CLR DRCSR	

The next procedure is to determine if a valid master is requesting service. Then the PDP checks if a trace or no trace operations is desired. The IOSL signal is checked to see if a memory or I/O process is desired. Finally, the state of DRVC indicates whether the operation is a 'read' or a write. Once it has been determined that a 'no-trace read' operation is to be performed the PDP must read the address register. This is accomplished in the following manner:

MOV #2, DRCSR	enable CSR1
MOV #31, DROUTBUF	get ready to read Address
CLR DRCSR	disable DSR1
MOV DRINBUF, R1	read the address into register 1
MOV (R1), R2	read whatever is at the address pointed to by R1 and place it into R3
MOV #2, DRCSR	enable DSR1
MOV #0, DROUTBUF	get ready to write into Data out register
CLR DRCSR	disable DSR1
MOV R2, DROUTBUF	write the data into Data-out register
MOV #2, DRCSR	get ready to write into CR
MOV #100004, DROUTBUF	send transfer acknowledge
MOV #30, DROUTBUF	clear DROUTBUF
CLR DRCSR	disable DSR1



The MOV #100004, DROUTBUF instruction causes the transfer acknowledge signal to be asserted which tells the microcomputer that the data is on the data lines. The microcomputer will take that information off the data lines and then remove the transfer request signal which, in turn, removes the TACK and completes the read cycle.

### 3.4 No Trace Write

The procedure for this operation is very similar to the previous one. This time, the PDP must read the data-line in addition to the address line but it does not place any data in the Data out register like it did previously. It only sends the TACK signal to indicate that the write operation has been performed.

LOOPB: MOV DRCSR, R0	read interface status register
BIC #177577, R0	mask off all bits except request A
BEQ LOOPB	if request A not set, then wait
MOV #1, DRCSR	clear request A
CLR DRCSR	disable DSR1
MOV #2, DRCSR	enable DSR1
MOV #31, DROUTBUF	set up CR to read Address lines
CLR DRCSR	disable DSR1
MOV DRINBUF, R1	place the address into R1
MOV #2, DRCSR	enable DSR1
MOV #32, DROUTBUF	set up CR to read Data lines
MOV DRINBUF, R2	read the data
CLR DRCSR	disable DSR1
MOV R2, (R1)	perform the 'write' operation
MOV #2, DRCSR	enable DSR1
MOV #100004, DROUTBUF	send TACK
MOV #30, DROUTBUF	clear DROUTBUF
CLR DRCSR	disable DSR1

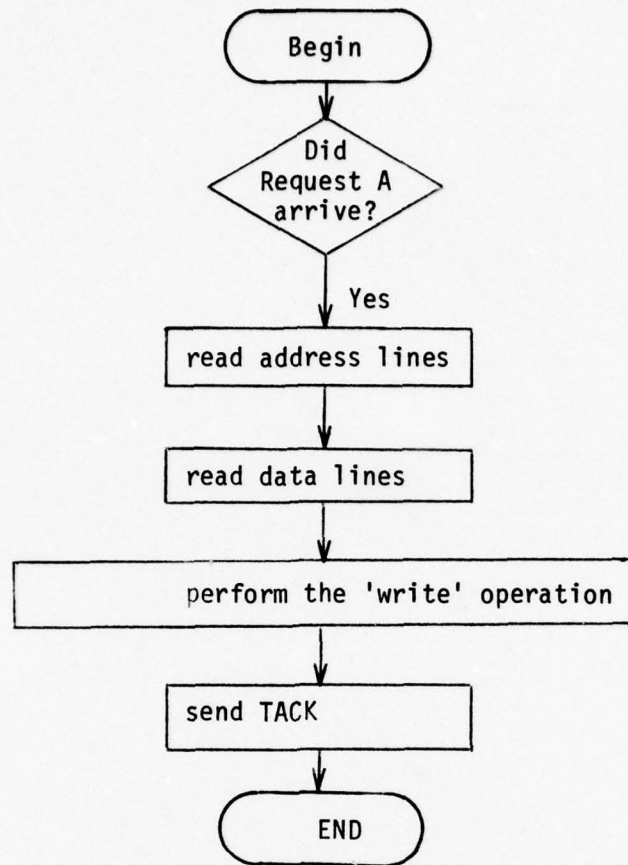


Figure 13  
No-Trace Write Flowchart

Note that the fourth instruction (MOV #1, DRCSR) enables the DSR1 bit and the fifth instruction enables the DSR2 bit. Theoretically the fifth instruction should just overwrite the third without having to do the clear operation. The problem is, however, that if the 'clear' is not issued that for a small fraction of time both, bit 1 and bit 2 are on thereby causing a master clear to be unintentionally issued. The programmer should always clear the DRCSR register before setting a different bit in it. The same is also true for the interface control-register.

### 3.5 Trace Read

Whenever the trace feature is enabled then the interface generates an interrupt to the microcomputer. The microcomputer will complete the ongoing transaction (in this case a read operation) and then honor the interrupt. The PDP program is identical to the normal 'read' operation except that when the PDP detects that the trace bit is set, when it reads the status register, it calls a subroutine, which interrogates the Request B bit. This bit gets set whenever the microcomputer honors an interrupt request.

```
LOOPC:  MOV DRCSR, R0
        BIC #177577, R0
        BEQ LOOPC
        MOV #1, DRCSR
        CLR DRCSR
        MOV #2, DRCSR
        MOV #31, DROUTBUF
        CLR DRCSR
        MOV DRINBUF, R1
        MOV (R1), R2
        MOV #2, DRCSR
        MOV #0, DROUTBUF
        CLR DRCSR
        MOV #2, DRCSR
        MOV #100004, DROUTBUF
        MOV #30, DROUTBUF
        CLR DRCSR
        JMP LOOP2
```

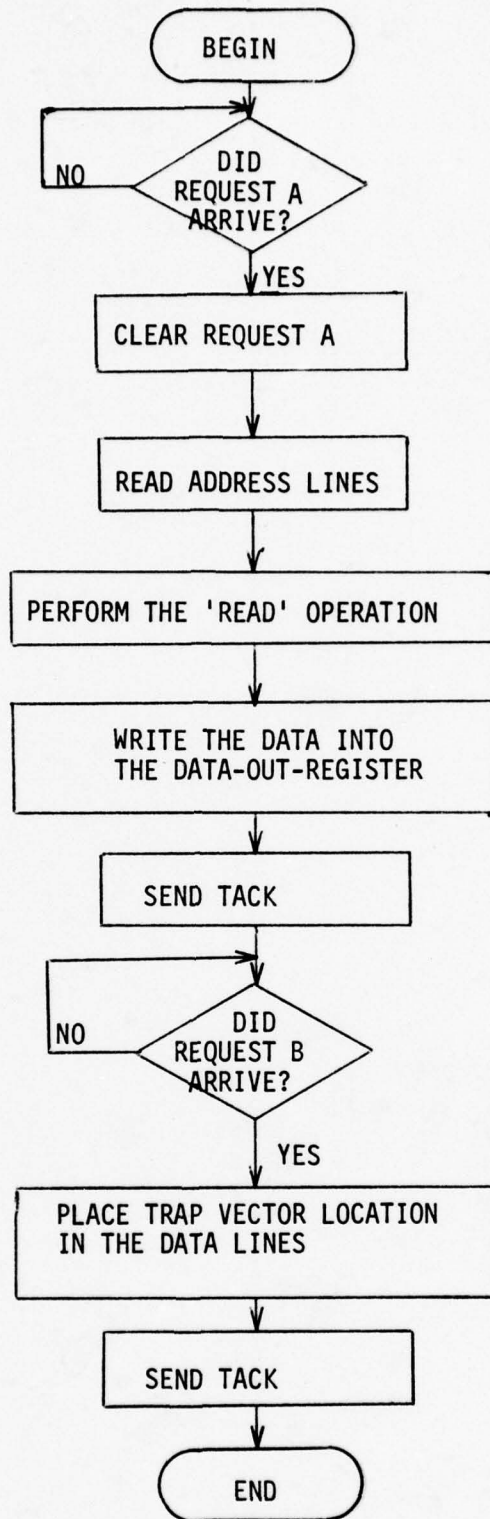


Figure 14

Trace Read Flowchart



LOOP2: MOV DRCSR, R1	read PDP status register
BIC 77777, R1	mask off all bits except request B
BGE LOOP2	if not set, then wait
MOV #2, DRCSR	get ready to write into CR
MOV #0, DROUTBUF	set up bit to write into Data out register
CLR DRCSR	disable DSR1
MOV #Addr, DROUTBUF	write the Address on the data lines
MOV #2, DRCSR	enable DSR1
MOV #1040, DROUTBUF	enable CRTACKI
MOV #100040, DROUTBUF	send the TACK signal
MOV #30, DROUTBUF	clear CR
CLR DRCSR	disable DSR1

### 3.6 Trace Write

This feature is very similar in operation to the trace read. The normal write operation is performed first and then, just like in the 'trace read' case, the interrupt is serviced. The following program is used for the 'trace write' operation.

LOOPD: MOV DRCSR, R0	read DR11C status register
BIC #177577, R0	mask off all bits except for request A
BEQ LOOPD	wait until request A comes in
MOV #1, DRCSR	clear request A
CLR DRCSR	disable DSR1
MOV #2, DRCSR	enable DSR1
MOV #31, DROUTBUF	get ready to read the address
CLR DRCSR	disable DSR1
MOV DRINBUF, R1	read the address into R1
MOV #2, DRCSR	enable DSR1
MOV #32, DROUTBUF	get ready to read the data
MOV DRINBUF, R2	read the data into R2
CLR DRCSR	disable DSR1

MOV R2, (R1)	perform the write operation
MOV #2, DRCSR	enable DSR1
MOV #100004, DROUTBUF	send TACK
MOV #30, DROUTBUF	clear DROUTBUT
CLR DRCSR	disable DSR1
JMP LOOP2	
LOOP2: MOV DRCSR, R1	read DR11-C status register
BIC #77777, R1	mask off all bits except request B.
BGE LOOP2	if not set, then wait
MOV #2, DRCSR	get ready to write into CR
MOV #0, DROUTBUF	set up CR bit to write into data register
CLR DRCSR	disable DSR1
MOV #Addr DROUTBUF	write the address on the data lines
MOV #2, DRCSR	enable DSR1
MOV #1040, DROUTBUF	enable CRTACKI
MOV #100040, DROUTBUF	send the TACK signal
MOV #30, DROUTBUF	clear CR
CLR DRCSR	disable DSR1

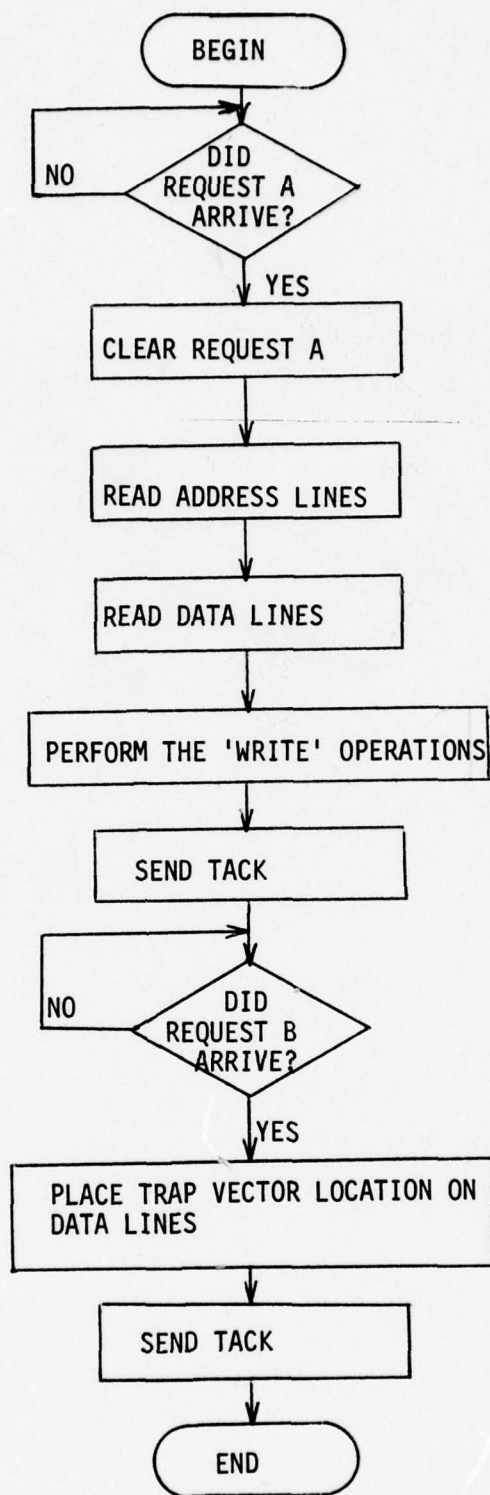


Figure 15

Trace Write Flowchart

## SECTION IV

### CONCLUSION

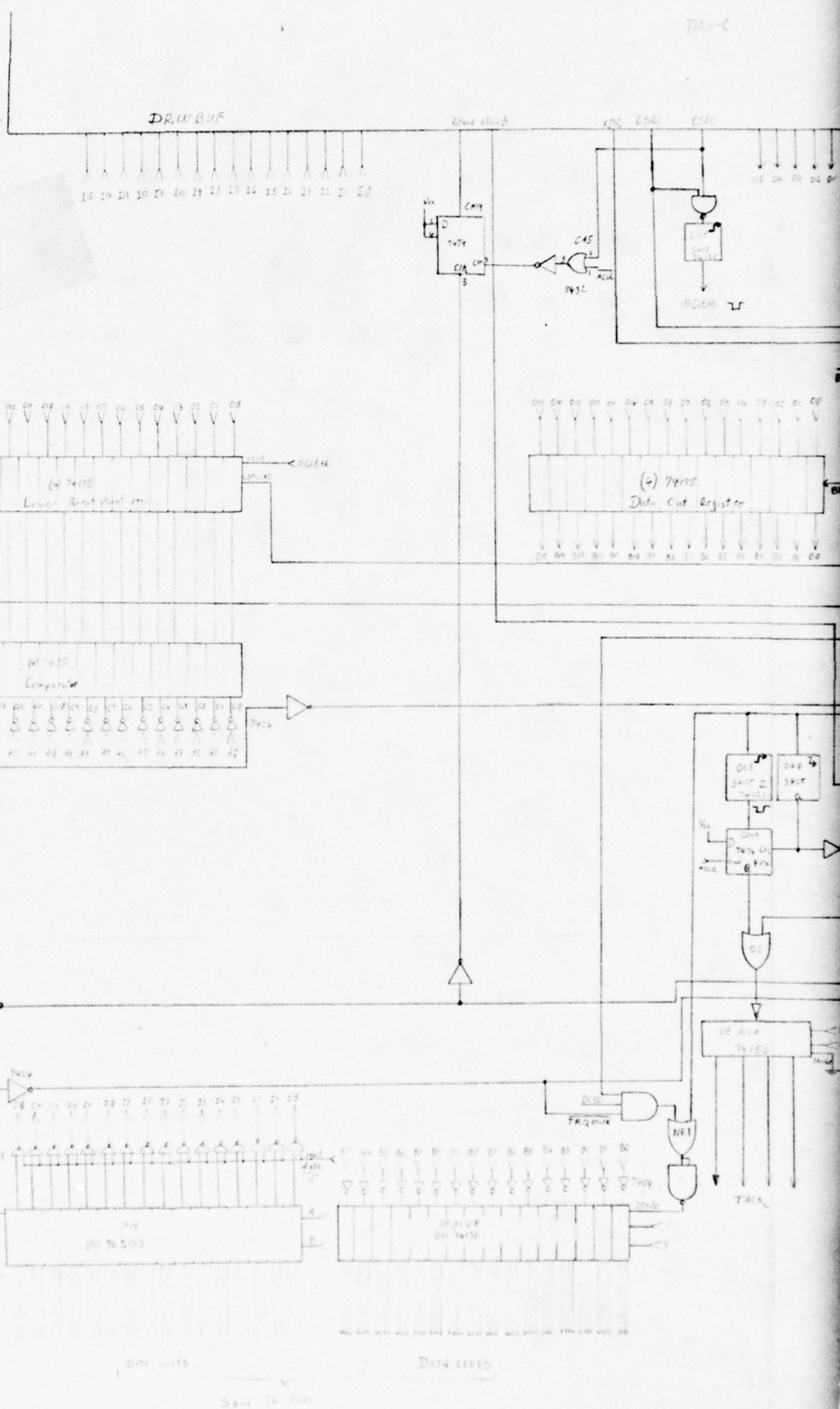
The unique feature of this interface is the fact that the microcomputers execute their programs directly out of PDP-11 memory. Whenever a microcomputer wants to perform a memory read or write operation, the interface notifies the PDP-11 what type of transfer is to be performed. The PDP software will perform the desired task and then notify the microcomputer when it is completed. By doing the traditional 'control panel' features in software, rather than hardware, speed is sacrificed but flexibility is gained. By simply changing the PDP-11 program, the interface can be made to perform many different tasks whereas similar hardware features would necessitate major design changes in the hardware. Using this design also keeps the cost relatively low, since the number of hardware components is kept at a minimum.



## APPENDIX A

### SCHEMATIC OF INTERFACE

THIS PAGE IS BEST QUALITY PRACTICABLE  
FROM COPY FURNISHED TO DDC



THIS PAGE IS BEST QUALITY PRACTICABLE  
FROM COPY FURNISHED TO DDC

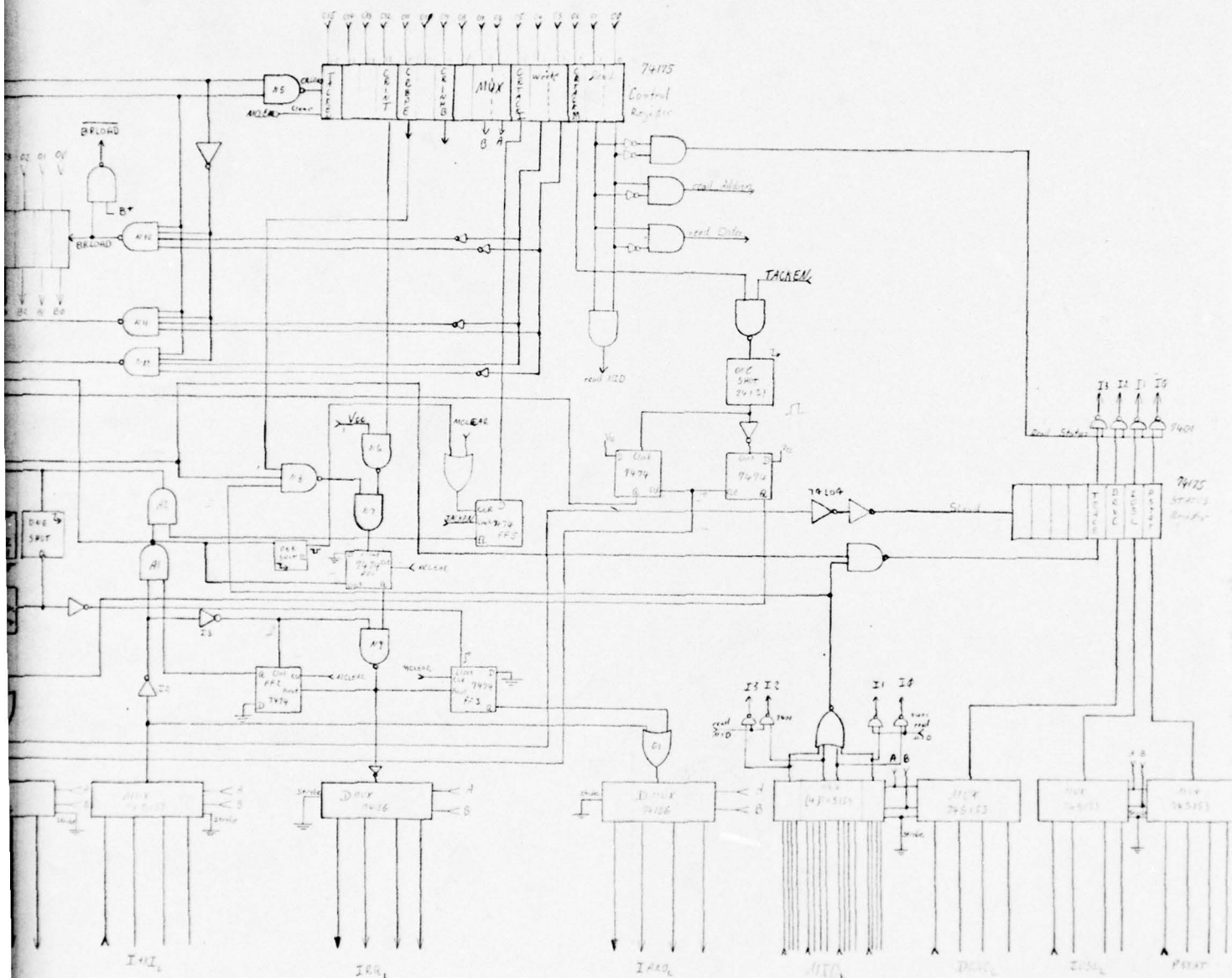


Figure 16  
Schematic of Interface

APPENDIX B

MICROCOMPUTER

INTERNAL BUS

(I-BUS)

SPECIFICATIONS



The I-Bus is a 16-bit data transfer bus and associated control lines which serve to transfer data between all subsystems within the microcomputer (MC). These subsystems include the CPU, the memory, the I/O, and the local and global serial bus interface units (L-BIU and G-BIU).

The I-Bus is asynchronous. The speed of data transfers over the I-Bus is determined by the speed of the devices interfaced to it.

Devices interfaced to the I-Bus compete for access on a priority basis. High-speed peripherals are usually assigned highest priority and the processor is assigned the lowest. In operation, cycle-stealing action occurs with the new bus assignment resolution overlapped with the previous data transfer.

There are two classes of devices which interface to the I-Bus: master devices, which control data transfers; and slave devices, which generate or accept data in response to some master. Data transfers in either direction always occur between one master and one slave. The processor is an example of a master device and a memory module is an example of a slave device. All slave devices recognize and are activated by specific addresses. For example, a memory is activated when some master device does a memory (MEM) data receive (DRCV) to an address within the boundaries of that memory module.

When an autonomous transfer controller (ATC) is started by the processor, it transfers data between memory and the peripheral device by cycle-stealing with the processor and any other master devices which may be active. When the ATC needs to transfer a word of data over the I-Bus, the master device part of the ATC must gain to the I-Bus and then may address a slave (such as a memory module) and read from it, write to it, or perform a read-modify-write with it.

The requirements are broken down into two major subsections. This specification covers the definition of the interface signals and their relations to each other.

## B.1 SIGNAL DEFINITIONS

The next two subsections cover I-Bus block diagram and the interface definitions. The convention is that a high voltage is a logic "1" (TRUE) and that a low voltage (ground) is a logic "0" (FALSE). Some of the signals are wired - or functions to allow the use of open collector outputs to drive these signals.

All signals are transmitted as complements.

### B.1.1 Block Diagram.

Figure 1 shows the various signals associated with the I-Bus to an MC module interconnection.

### B.1.2 Interface Definitions.

The subsections which follow define the signal lines which comprise the I-Bus. These signals are described in five groups according to their function. The signals associated with data transfer operations are defined in Section B.1.2.1. Those associated with bus acquisition are defined in Section B.1.2.2. Interrupt control signals are defined in Section B.1.2.3. Facilities such as MC system control, clocks and power are defined in Section B.1.2.4. Miscellaneous signals which serve special-purpose functions are defined in Section B.1.2.5.

#### B.1.2.1 Data Transfer Operations.

Data transfers on the I-Bus are handled as a demand/response sequence between a bus master (such as the processor) and a bus slave (such as a memory or an I/O device).

The following 37 signals are utilized exclusively for data transfer operations on the I-Bus. Thirty-two of these signals are 16-bits of address (ADD) and 16 bits of data (DATA) while the remaining signals (I/O select (IOSL), data receive (DRCV), transfer request (TRQ), transfer acknowledge (TACK), and transfer timeout (TTO) ) are used to control the actual transfer operations. All signals are transmitted and received between a I-Bus master device and an I-Bus slave device as defined in the following paragraphs.

#### B.1.2.1.1 Master to Slave Send Cycle

When an I-Bus master device has access to the I-Bus, it shall accomplish a send cycle through the following action. The master after gaining I-Bus control asserts transfer request (TRQ). At the same time the master shall assert a send command by pulling data receive (DRCV) low. The master shall also at this time specify whether it is a memory or I/O operation (IOSL), supply a valid 16 bits of data (DATA) and a valid 16-bit address (ADD). IOSL = TRUE implies a memory operation.

All slave devices interfaced to the I-Bus shall receive the transfer request (TRQ) transmitted from the master. The slave devices shall decode the address (ADD) to determine which slave is being addressed. The slave device shall internally delay the transfer request (TRQ) for a sufficient time to account for the worst case address decode time and the worst case I-Bus skew. Thus, each slave shall generate an internally delayed transfer request (TRQ) and use it to strobe a valid address decoder. The case of a memory module, the internally delayed transfer request (TRQ) and a valid address decode would generate a memory start signal. When the slave device has decoded the address as valid, it shall then assert transfer acknowledge (TACK). At the time the slave device asserts transfer acknowledge, it shall either clock the data (DATA), address (ADD), data receive (DRCV), and I/O select (IOSL) signals from the I-Bus into registers, or in the case of a memory, delay the transfer acknowledge (TACK) until the memory write cycle is complete.



When the I-Bus master device receives the asserted transfer acknowledge (TACK), it shall release transfer request (TRQ).

The master then looks for the release of transfer request (TRQ) before releasing data receive (DRCV), I/O select (IOSL), address (ADD), and data (DATA). This allows a slower third party device, such as a maintenance panel, to delay the transfer while it latches the data and/or address for monitor purposes.

When the slave device receives the release of transfer request (TRQ), it shall release transfer acknowledge (TACK). This is shown as "TIME4" of Figure 2.

When the master device receives the released transfer acknowledge (TACK), it may begin a new cycle or it may relinquish the I-Bus to another master device.

#### B.1.2.1.2 I-Bus Master From Slave Receive Cycle

When an I-Bus master device has access to the I-Bus, it shall accomplish a receive cycle through the following action. The master asserts transfer request (TRQ), and supplies a valid 16-bit address (ADD). At the same time the master specifies whether it is a memory or I/O operation (IOSL).

All slave devices interfaced to the I-Bus shall receive the transfer request (TRQ) transmitted by the master. The slave devices shall internally delay request and decode the address as for a send cycle (B.1.2.1.1). Each slave device shall internally delay the transfer request (TRQ) for a sufficient time to account for the worst case I-Bus skew and worst case address decode time. When this is done and the address is decoded as valid, the slave device will begin to send data. In the case of a memory module this means starting a read cycle. When the data (DATA) is valid, the slave device shall assert transfer acknowledge (TACK).



When the I-Bus master device receives the transfer acknowledge (TACK), it shall internally delay it to account for the worst case I-Bus skew and then release transfer request (TRO). As the master device releases transfer request (TRQ), it shall clock the data (DATA) from the I-Bus into its internal register.

The master then looks for the release of transfer request (TRQ) before releasing data receive (DRCV), I/O select (IOSL), and address (ADD). This allows a slower third party device such as maintenance panel to delay the transfer while it latches the data and/or address.

When the slave device receives the released transfer request (TRQ), it shall release transfer acknowledge (TACK) and data (DATA).

When the master device receives the released transfer acknowledge (TACK), it may begin a new cycle or it may relinquish the I-Bus to another master device.

#### B.1.2.1.3 Data Transfer Special Cases

In addition to this normal transfer request/acknowledge sequence there is one way in which a transfer can be terminated abnormally. A transfer time out (TTO) is generated by a watchdog timer on a memory controller. This time out is used to prevent a master device from locking up the I-Bus by attempting to address either nonexistent memory or I/O devices. When the time out occurs, the master uses the time out signal in the same manner as a transfer acknowledge signal, except it sets its own time out error flag and proceeds. In the case of the processor, this will correspond to an I/O acknowledge time out or a memory error.

#### B.1.2.2 Bus Control

There are three control lines and a set of four identification lines associated with bus master control. These three control lines are used to control the assignment of the bus to a bus master. The assignment is

performed on a first-come-first-serve basis, with hardwired priority used to resolve simultaneous requests. The four identification lines are associated with the maintenance function. The master that is in control of the bus places its identification code on these lines (Processor = 0; the ID default value). This identification is especially useful during "The Address Breakpoint" maintenance function where it is important to be able to identify who issued the breakpoint address. The use of four lines allows the unique identification of up to 15 bus masters in addition to the processor.

#### B.1.2.2.1 Normal Bus Requests

The three bus master control signals are bus request (BRQ), bus release (BREL) and bus grant (BGR). The bus request line (BRQ) is used to initiate a bus master assignment, while the bus release line (BREL) is used to terminate the assignment. The bus grant line (BGR) is threaded through each module and is used to resolve conflicts when two or more masters request the bus simultaneously. A master device can request the I-Bus only when bus grant (BGRI) is FALSE. When the bus request line (BRQ) is asserted, the bus grant (BGR) signal starts propagating through the modules on the I-Bus. When the signal reaches the highest priority requesting module, it activates that bus master, which blocks the bus grant (BGRO) signal from propagating to any lower priority module which may also be requesting the bus. When the master issues TRQ, it also issues a bus release (BREL). This command forces all bus masters, including itself, to remove their bus requests (BRQ). When the master sees BGRI go FALSE, it releases BREL and unblocks BGRO. Block transfers are accomplished by immediate re-requests of the bus. A device doing a block transfer can always be preempted between transfers by a higher priority device requesting the bus when BREL goes FALSE. A new bus master will wait until the bus is quiescent (transfer request (TRQ) and transfer acknowledge (TACK) have ended) and start its transfer cycle(s). This sequence allows bus mastership resolving to be overlapped with the data transfers, thus minimizing or partially eliminating the overhead associated with bus mastership assignment.

#### B.1.2.3 Interrupt Control

There are three I-Bus control lines associated with interrupt control. They are interrupt request (IRQ); interrupt inhibit (INHB); and interrupt acknowledge (IAK). Interrupts are serviced on a first-come-first-serve basis with priority resolution if there are simultaneous interrupts. The interrupt request (IRQ) is issued by any device that wishes to interrupt the processor (see Figure 6). There may or may not be an interrupt mask associated with (and internal to) the device which will inhibit the device from issuing an interrupt request (IRQ). In addition, there is an interrupt inhibit (INHB) line which is used by the maintenance panel to inhibit all devices from interrupting the processor when it is operating in the maintenance panel mode. When the processor completes the execution of the current instruction it honors any interrupt by asserting interrupt acknowledge (IAK). The interrupt acknowledge line is threaded through all modules and activates the highest priority interrupting device. This device inhibits the interrupt acknowledge (IAK) signal from propagating to lower priority modules. When the device receives its interrupt acknowledge (IAK) it places the address of its memory interrupt trap vector location onto the data lines. It then indicates that it has placed this address on the lines by asserting transfer acknowledge (TACK) after a 150ns. The processor latches the address into an internal register and completes the cycle by removing the interrupt acknowledge (IAK). The device waits for IAK to go away and then removes the address from the data lines, and terminates TACK.

#### B.1.2.4 General Facilities

The general bus facilities are composed of a free running clock (FCLK), master clock (MCLK), a master stop (MSTP), a clear/reset (CLR) signal, power and ground. The free running clock (FCLK) can be used by any device that needs a clock signal. It does not necessarily have any particular timing relationship with respect to any of the other bus signals. The master clock (MCLK) is similar to the freerunning clock except that it is controlled by the master stop (MSTP) signal while, the free running clock (FCLK) is not. In general, the free running clock (FCLK) will be used by slave devices, while, the master clock (MCLK) will be used by master devices.



This choice allows slave memories which must refresh themselves to continue even though the master stop (MSTP) signal is present. Special care must be exercised to make sure that each data transfer is allowed to go to completion before master stop (MSTP) is asserted. Otherwise, a dynamic memory may be inhibited from doing its normal refresh operation. The clear/reset (CLR) signal is used to initialize the processor and all other devices. The bus becomes quiescent when the clear/reset (CLR) is used. When the clear/reset issuing device issues CLR all activity on the bus aborts, and all signals, except BREL, go FALSE; BREL stays TRUE until CLR becomes FALSE. The CPU clears all general registers, sets the Program Counter to 0100 (HEX) and resets internal interrupt flags (for overflow). All other devices initialize their registers and disable (disarm) their interrupts. The power and ground lines are used to distribute power and ground to each device. A large number of pins are used to distribute power and ground over the connector to minimize any ground and/or power supply induced transients. The standard voltages have been selected ( $\pm 15$ ,  $\pm 12$ ,  $\pm 5$ ). Each voltage is assigned to two pins. This distributes the power and allows pins to be assigned in such a way that the card can be inadvertently plugged in backward without misconnecting the power pins.

#### B.1.2.5 Processor Status

There is a single signal associated with the processor status (PSTAT). It, in conjunction with the data transfer signals, indicates the state that the processor is in and the nature of the data (DATA) and address (ADD). The processor status (PSTAT) is only TRUE when the processor is making a transfer.

##### a. Program Counter and Instruction Registers

If the processor is receiving data (DRCV is TRUE) and the processor status (PSTAT) is TRUE, then the address (ADD) is the program counter and when the transfer acknowledge (TACK) comes TRUE, the data (DATA) is the first word of the instruction. A maintenance panel can use this information to display the program counter and instruction register during program execution. If the maintenance panel needs to, it can inhibit the



completion of the transfer by holding the transfer request (TRQ) TRUE until it has had a chance to record the program counter and instruction register values. This prevents the slave from dropping transfer acknowledge (TACK), which delays the completion of the cycle. This may be necessary if the processor is faster than the maintenance panel.

#### b. Interrupt State

If the processor is sending data (DRCV is FALSE) and the processor status (PSTAT) is TRUE, Then the processor is saying its program counter and status word. The address (ADD) is the interrupt stack pointer (ISP) and the first word sent, when the processor status (PSTAT) and the transfer request (TRQ) comes TRUE the first time, is the old program counter (PC). The second word sent, when the processor status (PSTAT) and the transfer request (TRQ) comes TRUE the second time, is the old status word (SW). Note that there will be an I/O device receive with a CAW = 00 (HEX) between the sending of the old program counter (PC) and the old status word (SW). The processor performs this I/O transfer to obtain those bits of the status word (SW) which are external to it. If the maintenance panel needs to delay the transfer of the old program counter (PC) and/or the old status word (SW) it can. To inhibit the completion of the transfer, the transfer request (TRQ) is held TRUE, as in "A" above.

#### B.1.2.6 Memory Control

Memory timing, address decoding, parity and write protection are controlled by a memory control circuit associated with each memory module. Each memory module is 17 bits wide, 16 for data storage and one for parity. A transfer time out circuit in the lowest (address) memory module generates the TTO signal when an I-Bus time out error occurs. The time out mechanism uses MCLK for timing. A unique interrupt trap location is provided for each memory module. A status register within each module is the concatenation of the write protect bit and the parity bit. The status register is assigned a CAW address, and is reset when it is read. The interrupt error mask bit is a part of the microcomputer status word, and is accessed through CAW = 0.

B.1.2.6.1 Memory parity is checked during each read, and calculated during each write. If it is not valid and if the interrupt bit is set, then the memory controller issues an interrupt and the parity bit is set. The read is allowed to complete.

B.1.2.6.2 The controller delays writes when the write protect bit is set to check whether the write is addressing a read-write portion of memory. If the addressed word is in a write protect area, the write is aborted and the write protect bit is set. If the write protect bit is set, and the interrupt error mask bit is set, the controller causes an interrupt.

#### B.1.2.7 Special Cases

##### B.1.2.7.1 Interrupts

BREL is generated by the bus master (always the CPU) during interrupt servicing at a different time than BREL is usually generated. BREL must be delayed until TACK is asserted by the interrupting device.

##### B.1.2.7.2 External Status Bits

Interrupts are masked by mask bits at each device. These mask bits are represented by 10 bits in the CPU status word. Reading or writing into the status word implies all devices must be accessed at once on the I-Bus for accessing each particular mask bit. This is accomplished by a broadcast mode I-Bus transfer that is designated by a CAW = 0. All devices must recognize CAW = 0. All devices must hold TRQ TRUE until they are ready to acknowledge.

SIGNAL TYPE	SIGNAL	SYMBOL	NO. OF WIRES
DATA TRANSFER AND CONTROL	DATA	(DATA)	16
	ADDRESS	(ADDR)	16
	I/O SELECT	(IOSL)	1
	+DATA RECEIVE	(DRCV)	1
	+TRANSFER REQUEST	(TRQ)	1
	TRANSFER TIME OUT	(TTO)	1
	TRANSFER ACKNOWLEDGE	(TACK)	1
BUS MASTER CONTROL	+BUS REQUEST	(BRQ)	1
	+BUS RELEASE	(BREL)	1
	BUS GRANT IN	(BGRI)	1
	BUS GRANT OUT	(BGRO)	1
	MASTER ID	(BMID)	4
INTERRUPT CONTROL	+INTERRUPT REQUEST	(IRQ)	1
	+INTERRUPT INHIBIT	(INHB)	1
	INTERRUPT ACKNOWLEDGE IN	(IAKI)	1
	INTERRUPT ACKNOWLEDGE OUT	(IAKO)	1

+ Denotes a wired or signal.  
All signals are low true.

Fig B.1 I-Bus Signal List  
51

SIGNAL TYPE	SIGNAL	SYMBOL	NO. OF WIRES
GENERAL BUS FACILITIES	FREE RUNNING CLOCK	(FCLK)	1
	+MASTER CLOCK	(MCLK)	1
	+CLEAR/RESET	(CLR)	1
	POWER/GROUND		20
SPECIAL FUNCTIONS	PROCESSOR STATUS	(PSTAT)	1

Power & Ground; 2 pins each

+15

+12

+5

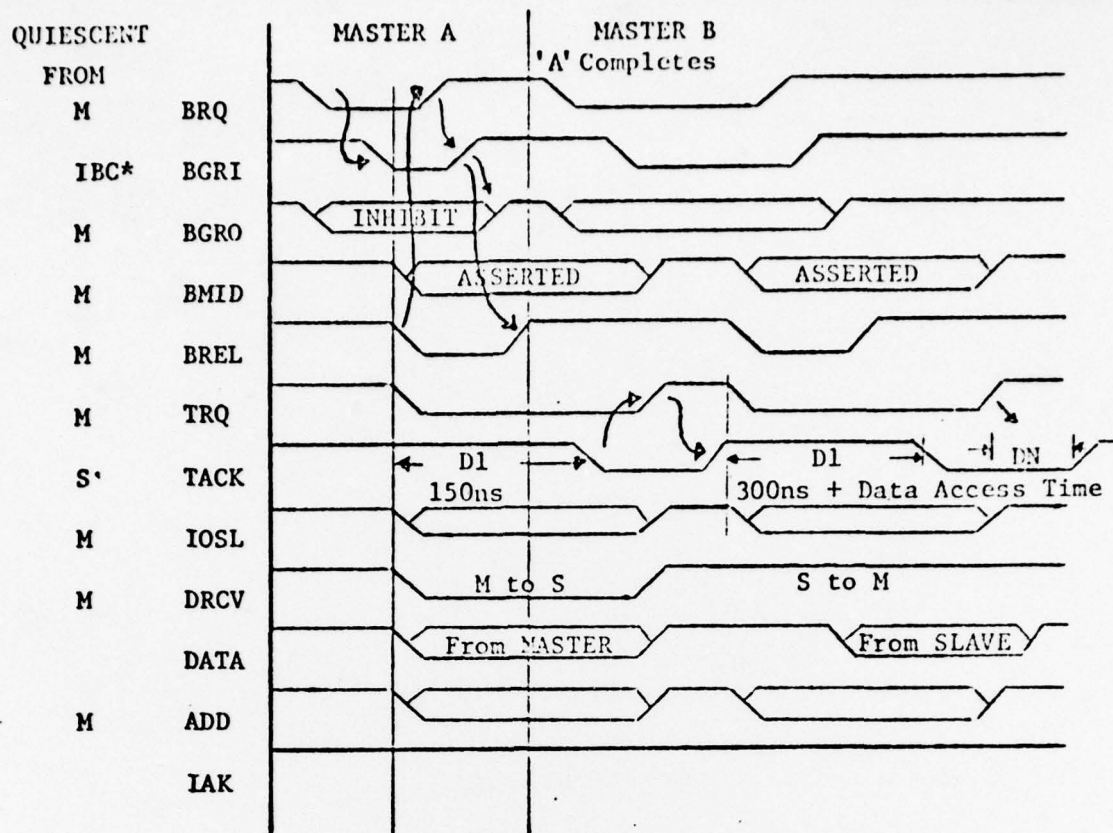
8 Pins for Ground

+Denotes a wired or signal.

All signals are low true.

Figure B.1 I-Bus Signal List (Continued)





This diagram shows two bus master control cases; the first transfer starts when the bus is quiescent. The second transfer overlaps bus acquisition with the completion of the previous transfer.

D1 is a delay induced by the slave device. The slave delays 150ns for bus slew and address decoding. For transfers from the slave to the master, the slave delays another 150ns for bus slew for sending data back to the master.

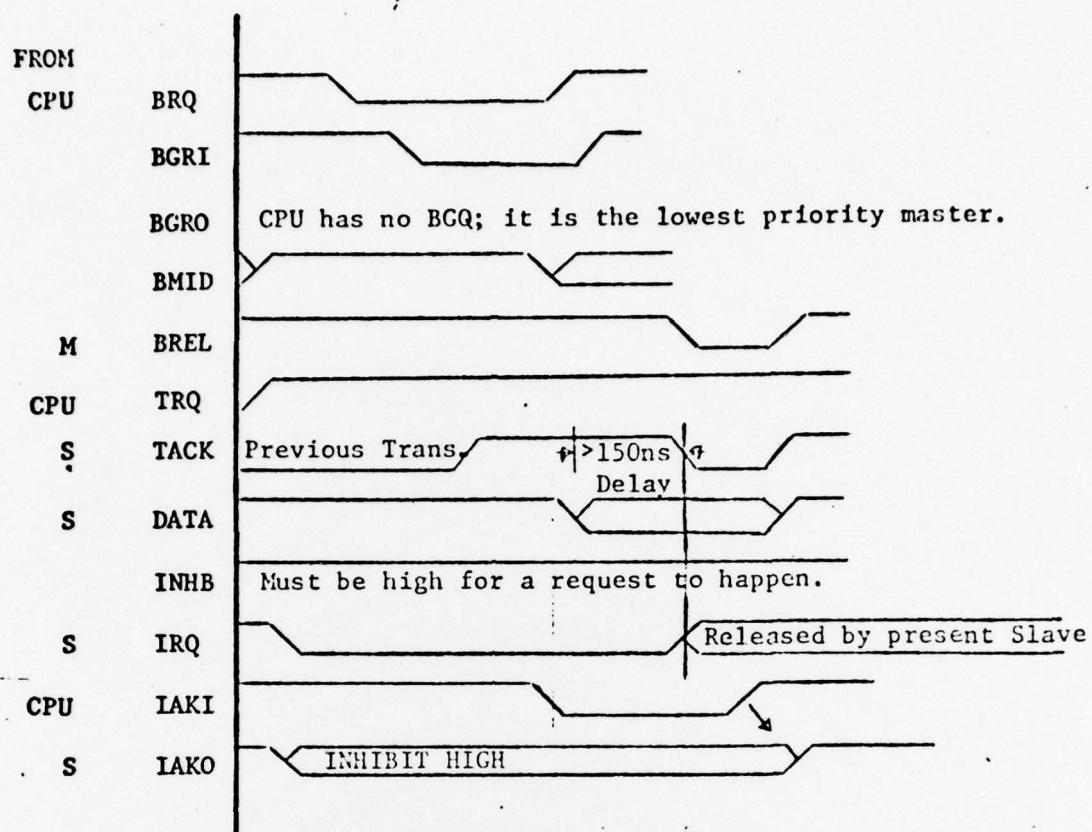
DN: Nominal delay (20ns) to insure that the slave and next master do not overlap assertions on the data lines.

Note that TRQ cannot be asserted by the master until TACK, TRQ, and IAK are false.

The BREL pulse is a nominal length,  $\approx 50$ ns.

\*IBC = I-Bus Control Logic

Fig B.2 I-Bus Control & Data Transfer Timing



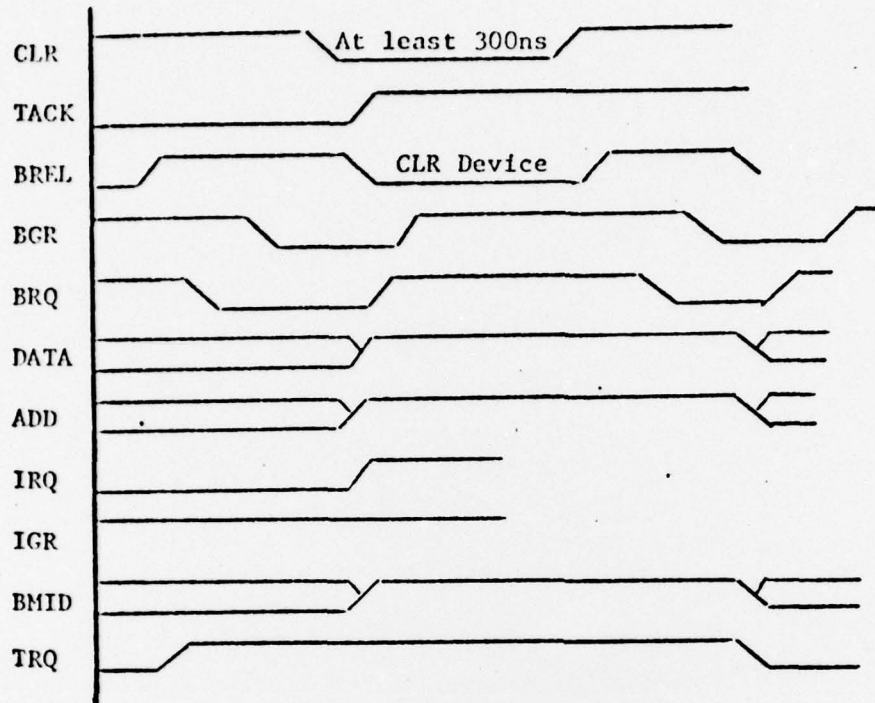
The IRQ can be asserted when IAKI is false.

Note that the CPU does not issue a BREL pulse until TACK and IAKI are both true.

Fig B.3 I-Bus Interrupt Interactions with Normal Bus Control and Data Transfers.

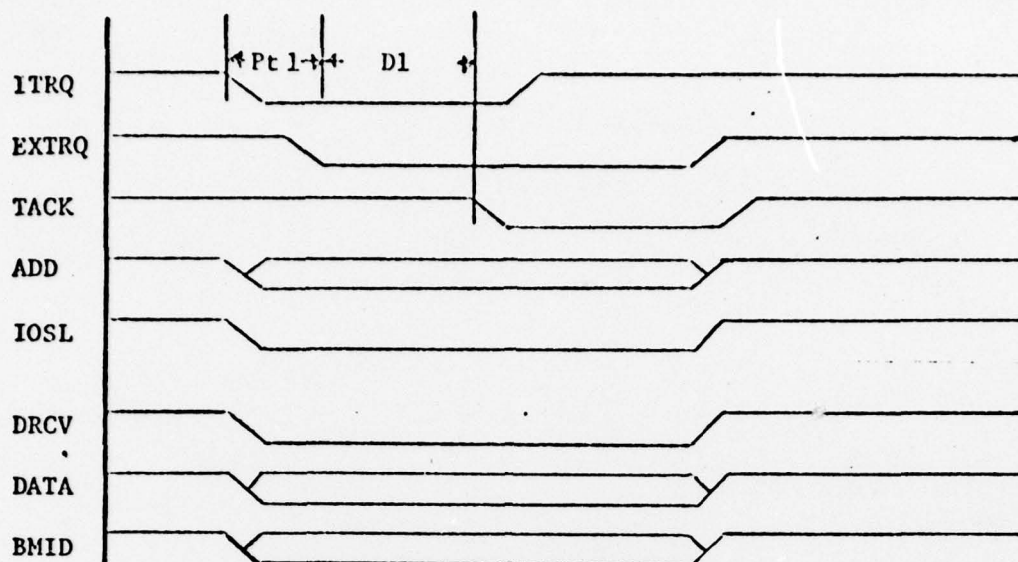
From

CLR Device



The CLR signal causes bus quiescence by forcing all asynchronous signals to false (or disable, where applicable) except BREL, which stays true when CLR is true. An arbitrary minimum signal length has been selected to be 300ns. or greater to guarantee that all Masters and Slaves have ceased activity on the bus.

Fig B.4 CLEAR-RESET Timing



#### DETAIL OF TRQ INTERACTION

ITRQ = Internal to master transfer request applied to TRQ wire.

Pt1 = Propagation delay time for open collector bus driver.

EXTRQ = wired or signal that actually exists on the TRQ wire.

D1 = Total delay time for address decoding and slew.

Note that ADDR, IOSL, DRCV, BMID, and DATA are not removed until the TRQ, sensed on the TRQ wire, is false. The TRQ may be held high by slow devices during multi-device accesses (i.e., CAW = 0), until they send back TACK.

Fig B.5 TRQ Detail