THE UNIVERSITY OF MICHIGAN

Technical Report 31

DEC PDP-7/IBM 1800 HIGH SPEED INTERFACE REFERENCE MANUAL

J. L. Foy, Jr.
R. F. Brender
D. R. Frantz
J. A. Miller

CONCOMP: Research in Conversational Use of Computers
F.H. Westervelt, Project Director
ORA Project 07449

supported by:

ADVANCED RESEARCH PROJECTS AGENCY
DEPARTMENT OF DEFENSE
WASHINGTON, D.C.

CONTRACT NO. DA-49-083 OSA-3050
ARPA ORDER NO. 716

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

August 1970

Engn
UMR
1502

ABSTRACT

This report describes an interface between an IBM 1800 computer and a DEC PDP-7 computer. It has the following features: 1) it allows the transfer of blocks of data directly from the memory of one computer to the memory of the other, at up to 125,000 words per second, in parallel with program execution; 2) it allows a program running on one machine to interact asynchronously with one running on the other through a system of "attention" interrupts; 3) it is relatively simple to control, being symmetric to both computers; 4) it compensates automatically, in either of two modes, for the difference in word length between the PDP-7 and the 1800.

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

The work described here was conducted at the Logic of Computers Group, a research unit within the Department of Computer and Communication Sciences of The University of Michigan, under the direction of Professors A. W. Burks and J. H. Holland. In addition to individual dissertation projects, two major, long-range projects are being developed on the computer facilities located at Logic.

The first is a simulation system for investigation of cellular, or iterative array, models. Members of the Group have designed and implemented a computer language and programming system specifically oriented toward this class of models. Two investigations currently under way on the system are a neural network simulation and a simulation of a biological cell population. Additional work is planned to explore models exhibiting adaptive or learning behavior.

The second project, under the direction of Professor M. H. O'Malley of the Phonetics Laboratory of the Department, is implementing an on-line speech analysis and synthesis system for both teaching and research. The special equipment for this work is interfaced to the IBM 1800. Capabilities are being created to input, store, analyze, edit, and synthesize speech and/or coded speech data.

The hardware facilities present a number of awkward problems not often found elsewhere. With its display, the PDP-7 has the greater potential for interactive control and, of course, for graphic presentation of the speech analysis. On the other hand, the 1800 has the bulk storage, more

core storage, and in general a more elaborate operating system (i.e., a full batch monitor); hence program development tends to be easier on the 1800. The general result is that most large programs tend to be divided functionally between the two computers with the PDP-7/display used for man-machine interaction and the 1800 serving as the primary computation or simulation processor.

A slow interface (about 8000 baud) has been available since 1967 but certainly was too slow for many of our projected uses. A block-transfer interface was set as an objective to achieve high data rates.

It was an unfortunate fact that the bulk storage device was on the computer with the smaller-sized word (1800 word has 16 bits; PDP-7 word has 18 bits). Had it been the other way, the data path could have provided word-for-word transfers and the overhead of unused bits on the disk would probably have been accepted without much protest. However, the larger word machine was at the end of the chain and we definitely wanted to load programs into the PDP-7 from the 1800 disk. We also anticipated that much of our simulation data would be integer valued and should be movable from one machine to the other on a word-for-word basis. We rejected a scheme for storing one PDP-7 word in two 1800 words as being excessively wasteful of storage space. This eventually led to the data path currently used.

We also felt it was very important that the Interface be easy to set up and control, the small core memory available being too valuable to consume with elaborate device support programs. It was also thought desirable that either computer be able to control the Interface by itself.

In setting forth these requirements, we were quite aware that their reasonableness depends on the fact that we regularly use our computers

as a single system. We wanted maximum flexibility rather than protection from errors in the other computer.

This Interface has fulfilled those goals. It provides what we believe to be the most tightly coupled two-computer system possible without shared memory.

The functional design was developed by R. F. Brender and J. L. Foy, Jr. The Interface was built by Logic, Inc., of Detroit, Michigan. J. Miller, of that company, did the logic design in close cooperation with Foy. Acknowledgment is also due G. Cooper for helpful advice. Brender and Foy developed the diagnostic software; Foy and D. R. Frantz designed and implemented the device support software.

The authors of various sections of this report are:

Preface	Brender
2. Functional Description	Brender and Foy
3. Engineering Description	Foy and Miller
4. Programming Support	Frantz and Foy
5. Evaluation and Conclusion	Brender
Appendix A. Interface Control Panel	Foy
Appendix B. Command Summaries	Foy
Appendix C. Test Programs	Brender and Foy

1. INTRODUCTION

This report describes interface equipment specially designed for the Logic of Computers Group of the Computer and Communication Sciences Department at The University of Michigan.

The Group's computer facilities support two major research projects: one involves an interactive simulation system oriented toward cellular models, the other is concerned with on-line analysis and synthesis of speech. The hardware consists of an IBM 1800 system with 1.5 million words of disk storage, to which the speech analysis and synthesis equipment is attached, and a DEC PDP-7 equipped with a modified 338 display (see Figures 1.1 - 1.3). The two central processors are interconnected by a low-speed interface capable of transferring data one word at a time under program control; thus each machine appears to the other as a medium-speed I/O device. The shortcomings of this arrangement spurred the design of the interface described herein, which has the following features: 1) it allows the transfer of blocks of data directly from the memory of one computer to the memory of the other, at up to 125,000 words per second, in parallel with program execution; 2) it allows a program running on one machine to interact asynchronously with one running on the other through a system of "attention" interrupts; 3) it is relatively simple to control, being symmetric to both computers; 4) it compensates automatically, in either of two modes, for the difference in word length between the PDP-7 and the 1800.

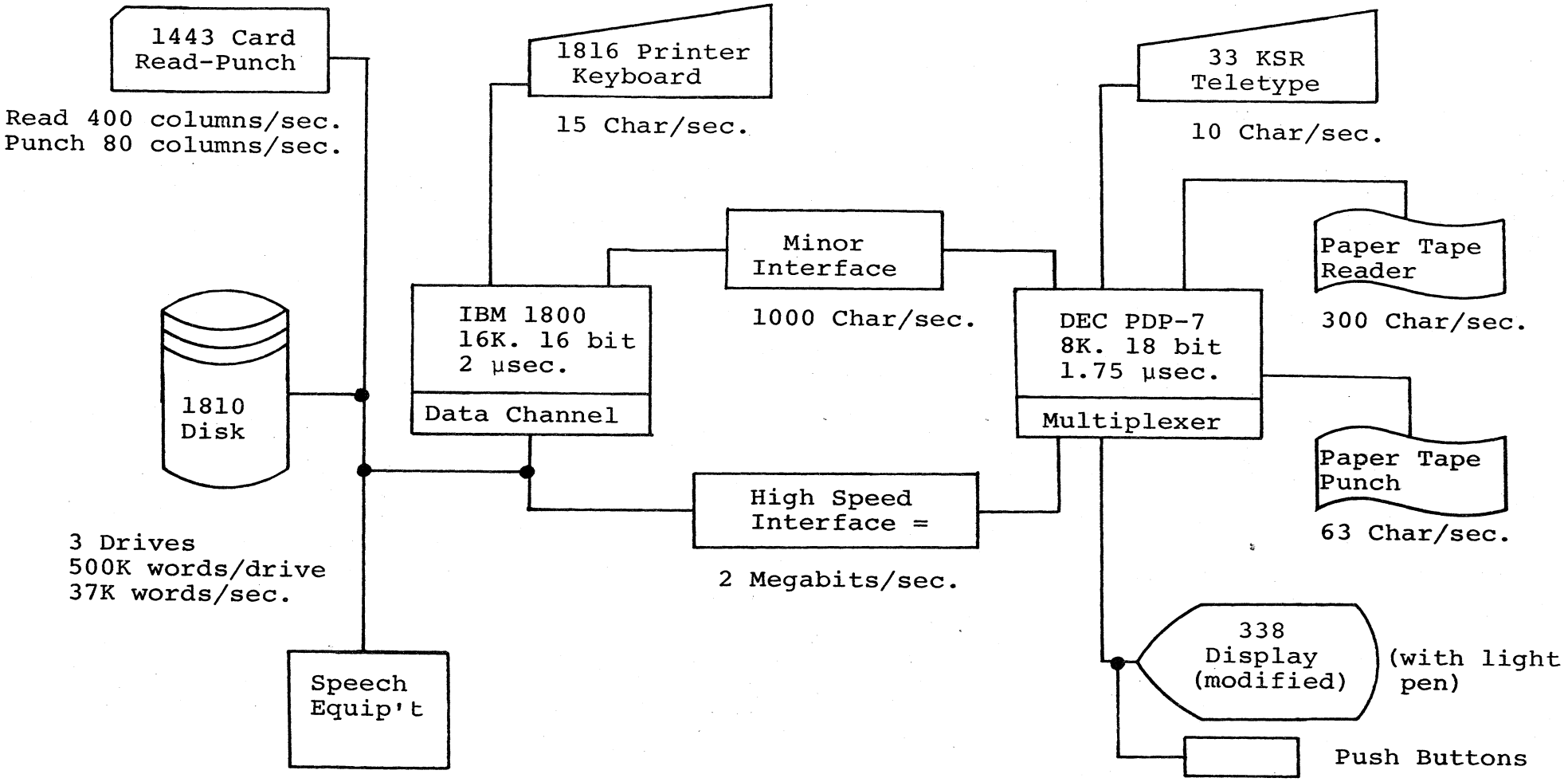


Figure 1.1 Logic of Computers Group Computer Complex

CPU (1801C2)

16K of 2μ sec core

16 bits/word + parity and storage protection

priority interrupt system (12 levels)

3 index registers

1- and 2-word instruction formats

6 data channels

Keyboard-Printer (1816)

15 characters/second + hardware tabbing

Card Read-Punch (1442)

read 300 cards/minute

punch 60 cards/minute

Disk (1810A3)

3 independent drives

movable heads

interchangeable cartridges (1815)

512,000 words per cartridge

Figure 1.2 Logic of Computers Group 1800 System

CPU

8K of 1.75μsec core

18 bits/word

hardware interrupt

Teleprinter (33KSR)

10 characters per second

Paper Tape Reader

8-channel

300 characters per second

Paper Tape Punch

8-channel

63 characters per second

CRT Display (Modified 338)

A display consisting of a DEC 338, less the PDP-8 portion of the 338, is interfaced to the PDP-7. This is locally known as a 337 and is the prototype for the DEC 339. The display operates asynchronously from instruction files in the PDP-7 memory. It provides point, increment, short vector, vector, and character plotting modes, and is capable of branches and subroutining as well as conditional branches, depending on the state of user-controlled switches.

Figure 1.3 Logic of Computers Group PDP-7 System

2. FUNCTIONAL DESCRIPTION

The structure of the Interface is the result of two major design decisions: the basic data path would consist of a circular shift register providing variable frame or word sizes, and the major control registers would be fully accessible to *both* computers making it possible for *either* computer to initiate a data transfer without cooperation of the other. In addition, the Interface has carefully been kept as symmetric as possible with respect to the control capabilities of the two computers.

2.1 Functional Organization

This section discusses the functional organization of the Interface. The later parts of this chapter will present the commands available to the two computers. It may be useful to refer to Figure 2.1 during the discussion.

2.1.1 Control Registers

The Interface consists of seven major registers:

1. Data Register
2. 1800 Shift Counter
3. PDP-7 Shift Counter
4. 1800 Address Register
5. PDP-7 Address Register
6. Unit Count Register
7. Status Register

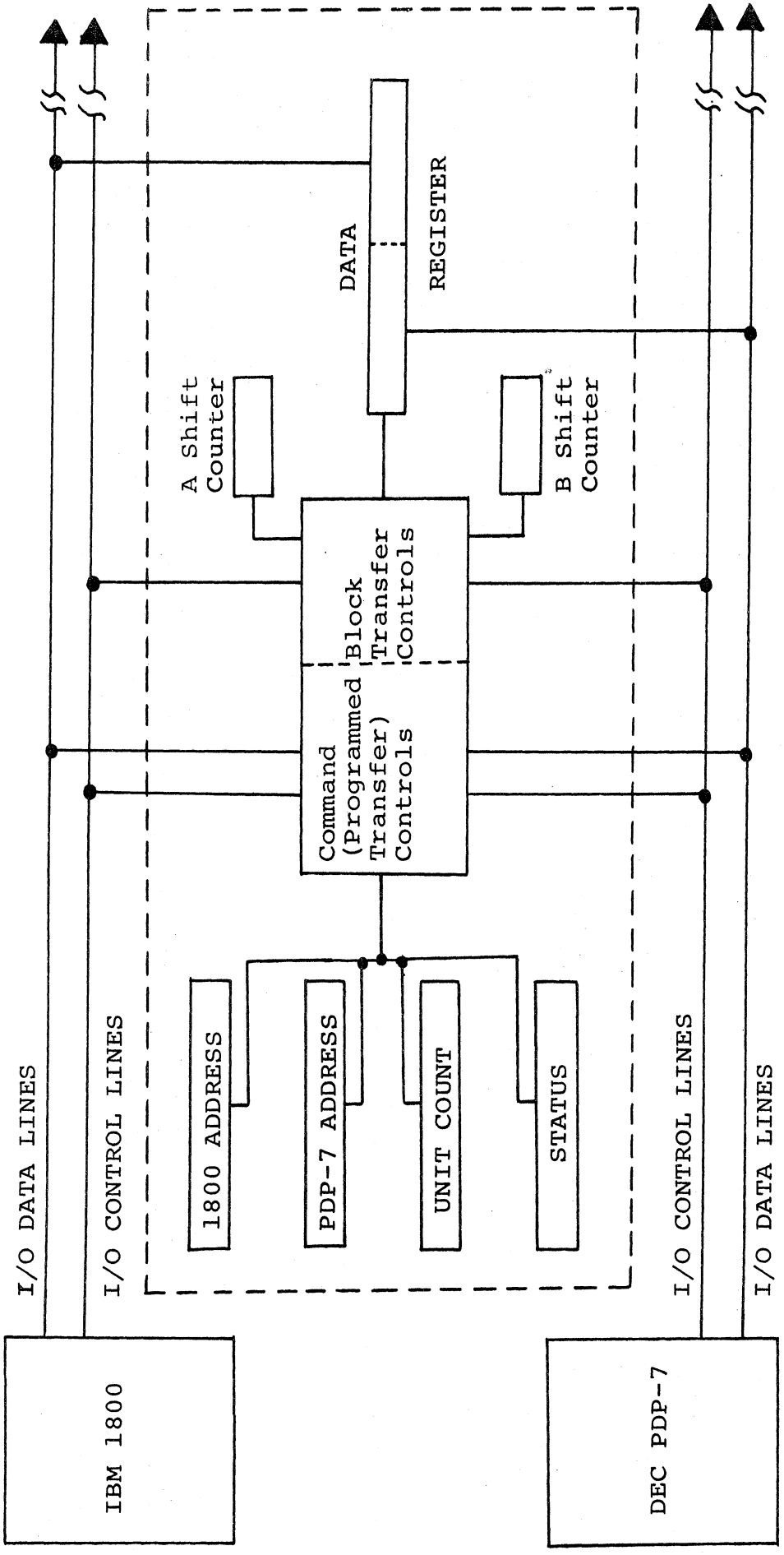


Figure 2.1 Interface Block Diagram

The first three registers are not directly addressable by the programmer; they are used by the Interface itself during block transfers of data. The Data Register communicates directly with the core storage of each computer by means of the direct memory access facilities, known as "data break" on the PDP-7 and as "cycle steal" on the 1800. The Shift Counters control the special functions which compensate for the difference in memory word size between the PDP-7 and 1800 (see Section 2.1.2.).

The last four registers are directly controllable by the user and may be read and written from either computer. The Interface will respond correctly to simultaneous Reads and/or Writes from both computers; it is the programmer's responsibility to ensure that such operations make sense, since simultaneous Writes to the same Interface register will produce indeterminate results.

The 1800 address register specifies the 1800 core address from (to) which data is obtained (stored). The PDP-7 address register specifies the PDP-7 core address from (to) which data is obtained (stored). The unit count register counts the number of units[1] transferred and causes a transfer to terminate when it reaches the value zero.

Although neither computer currently requires more than 14 bits to specify an address, these registers are each 16 bits in size. Thus, when not used for data-break transfers, they are readily useable as a full-duplex direct-program-controlled data path. When used in this manner two of the registers are used for data and the third for control flags.

---

1 A "unit" of data will be defined later. For the moment it is sufficient to consider a unit as being one word.

The Status Register is a 16-bit entity through which the programmer may exchange control information both with the Interface itself and with the programmer of the opposite computer. The eleven high-order bits of this register consist of various flip-flops which indicate and/or determine the internal condition of the Interface; all may be read by the programmer, and some may be individually set, cleared, or both. The function of each bit is detailed below.

The five low-order bits of the Status Register are actually two separate registers: one receives data from the PDP-7 (whenever the PDP-7 issues a Write addressed to the Status Register) and presents data to the 1800 (whenever the 1800 issues a Read addressing the Status Register), while the other receives from the 1800 and presents it to the PDP-7. In this way, the two computers can simultaneously present status information to each other. Because they are used in conjunction with the "attention" interrupt facility, these five-bit sections are referred to as the "attention codes".

In order to set up a data transfer, one of the computers seizes the Interface for its use according to a convention described below. Then it loads the address registers with appropriate addresses, loads the Unit Count register with the length of the data block, and sets the bits of the Status Register to indicate direction of transmission and handling of operation-complete interruption. Then the computer issues a Start command and the transfer commences.

2.1.2 Data Path

The data path consists of a Data Register (also called the Shift Register) and two Shift Counters. The 34-bit Data Register is composed

of a 16-bit (A) and an 18-bit (B) section; the 16-bit section can be loaded in parallel from the 16-bit output bus of the 1800, and the 18-bit section can be loaded in parallel from the 18-bit I/O bus of the PDP-7. Similarly, the 16-bit section of the Data Register can be gated to the input bus of the 1800 while its 18-bit section can be gated to the I/O bus of the PDP-7. The two sections of the Data Register, then, are capable of independent, parallel transfers of data to and from their associated computers, while the Data Register as a whole can be rotated right. Serial and parallel operations are performed alternately: data are parallel-transferred from the sending computer to its section of the Data Register, the latter then shifts right as necessary to move the data into the other section, and then the data are parallel-transferred from that section to the receiving computer.

The following detailed description of the events that take place during a transfer from the PDP-7 to the 1800 in the "18/16" mode illustrates the functions of the Data Register and the two Shift Counters.

When the Start command is given, the 1800 Shift Counter is set to 16 and the PDP-7 Shift Counter to 18. A word is loaded from the PDP-7 into its portion of the Data Register. Shifting is begun, and both shift count registers are decremented for each bit position shifted. Shifting continues until one of the shift count registers reaches zero. In this case, this will occur after 16 shifts, which have moved the low-order 16 bits of the PDP-7 word into the 1800 portion of the data register. Shifting is suspended until the 1800 part of the data register can be stored in the 1800 memory; then the 1800 Shift Counter is reloaded with 16, and shifting continues. After two more shifts, the PDP-7

Shift Counter will go to zero. At this point, the two high-order bits of the PDP-7 data word are in the 1800 portion of the data register. Shifting is suspended while the next PDP-7 memory word is loaded into the PDP-7 portion of the Data Register. Then the PDP-7 Shift Counter is reloaded with 18 and shifting resumed. After 14 more shifts, the 1800 Shift Counter will go to zero. The 1800 portion of the Data Register, which now contains two bits of data from one PDP-7 word and 14 bits from the next, will be transferred into the 1800's memory. This process continues in the same manner until both Shift Count Registers go to zero simultaneously. This will occur when an integral number of words has been moved in each machine, and is considered one Unit Operation. The Unit Count Register is then decremented. The entire sequence is repeated until the Unit Count Register has become zero, which signals the end of the transfer operation.

When the user wants to transmit data from the PDP-7 to the 1800 for processing, he uses the "16/16" mode. This mode is very similar to that described above, except that the PDP-7 Shift Counter is loaded with 16, so that after every right-shift of 16 bits the PDP-7 section of the Data Register is reloaded with another 18-bit word while the 16-bit 1800 section is being written into the 1800. Reloading the PDP-7 section after a shift of only 16 places destroys the two high-order bits which remain there; this information is simply lost. However, it is the purpose of this mode to transfer the low-order 16 bits of each word of the PDP-7 to corresponding words in the 1800, preserving the "word" relationships necessary for efficient processing. (Often the data will fit naturally into a 16-bit frame anyway, as when two 8-bit characters are packed together, so that no useful data are lost at all by transmitting only

16 bits.)

As a consequence of this organization, a unit of data is dependent on the mode used. In the "16/16" mode, one unit corresponds to one word in each machine. In the "18/16" mode, one unit corresponds to nine words on the 1800 and eight words on the PDP-7.

The inverse transfers are performed very similarly. Bits are shifted out of the low-order part of the 1800 section into the high-order part of the PDP-7 section. Note that in the "16/16" mode from 1800 to PDP-7 the bits are not shifted into the very left of the PDP-7 section but rather into a point 16 bits from the low end of the word. Thus after 16 shifts a word is properly aligned in the low-order part of the word (right justified).

To simplify exchanges of integer valued data in the 16/16 mode, the high-order bit of data entering the PDP-7 is fanned out to the three highest order bits in the PDP-7. Since twos complement arithmetic is used by both machines, signed integers are treated appropriately. On 16/16 mode transfers to 1800, the three high-orders bits should all be the same to avoid errors introduced by truncation. This is *not* checked by the hardware.

2.1.3 Special Operations

The Status Register contains two bits, known variously as the "test-and-set", "seize" or "resolution" bits, which (among others) cannot be set directly. Their purpose is to allow the resolution of contention situations, in which both computers attempt to perform conflicting operations on the Interface at the same time. One bit is assigned to mean that the PDP-7 is controlling the Interface, the other that the

1800 has control. A computer "seizes" the Interface by issuing the Test & Set command, which turns on the corresponding resolution bit if and only if the resolution bit for the other computer is off. The Test & Set instruction always resolves unambiguously: one and only one bit will be turned on even if the command is issued simultaneously by both computers. The resolution bits are provided purely for the use of the software in each computer; they have no effect whatever on the Interface (except, of course, for the Test & Set command). By programming convention, each computer "seizes" the interface before using it and releases it by clearing its resolution bit, when finished. "Attention" signals passed solely through the Status Register constitute the only exception to this rule.

The Test & Set command has two forms: immediate and pending. The first takes immediate effect, and either succeeds or fails depending on the state of the other machine's resolution bit. The second is "remembered" by the Interface and "seizes" it as soon as the other machine's resolution bit is zero, causing an interrupt to the issuing computer. The programmer can thereby acquire a busy Interface as soon as it becomes available without having to wait in a test loop.

A general reset of the Interface will occur whenever power is turned on for the PDP-7 or 1800, and also whenever the RESET buttons on the Interface console or the 1800 console are pressed. This same function is available to the programmer as the Blast instruction, which clears the interface and resets all control circuits to the idle state. Any ongoing data transfer will be halted immediately (but not necessarily correctly), and any pending "seize" will be cleared.

The Halt command will stop any data transfer at the end of the block (1 word or 8/9 words, depending on mode) in which the Halt was received by the Interface. The Address, Count, and Status Registers will be in the proper states such that the transfer may be resumed by issuance of the Start command. If these registers are logged out, the Interface can be used for any other operations before the registers are reloaded and the Start issued. This allows high-priority transfers to interrupt lower-priority ones whenever desirable.

2.1.4 Attentions and Errors

Interrupts from the Interface may be divided into two groups: expected and unexpected. Expected interrupts occur at the completion of a data transfer or the success of a pending Test & Set (explained in 2.3). Unexpected interrupts result from explicit action of the opposite computer or from errors detected during data-break transfers. In this Interface unexpected interrupts are called "attentions" and special provisions are made for handling these events.

An attention is initiated by one computer loading the attention code and setting the attention bit for the opposite computer. When the interrupt is serviced by the receiving computer, it reads the attention code to determine the action to be taken. When ready to accept another attention the computer must clear its attention bit.

The attention code is useful for providing control information between the two computers independent of and even in parallel with ongoing data transfers. In addition, it provides a means to initiate a "cooperative" data transfer as in this example:

The PDP-7 has data to be sent to the disk storage device attached to the 1800. After seizing the Interface, the PDP-7 loads the PDP-7 address and count registers and also loads the desired disk memory

address into the 1800 address register. It then gives an attention command to the 1800. The attention register provides a means to specify to the 1800 that a particular interpretation is to be given to the content of the 1800 address register. The 1800 must save the 1800 address content for later use, then load it with the memory address of an available buffer region and start the data transfer. The total operation is completed by the 1800 writing the data onto the disk at the previously designated location.

The use of a separate register as an attention code register makes cooperative initialization such as this quite simple, although the basic registers can be used if more elaborate control sequences are employed. Such an additional register provides a signal path for synchronizing intricate control sequences between the two computers. It also provides a separate signal path that can be used in parallel with on-going data-transfers.

The Interface is capable of detecting three kinds of errors during data transfers: a parity error or storage-protect violation on the 1800 (those features are not installed on the PDP-7), or an attempt by either computer to alter the Count or Address Registers while a data transfer is actually in progress. When an error is detected, the following events occur:

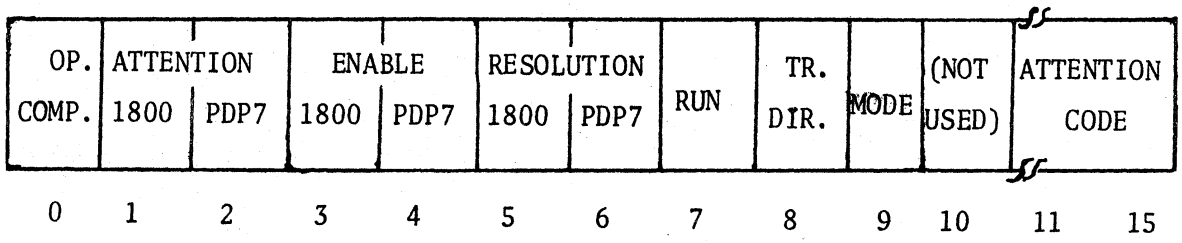
1. The transfer is aborted immediately if the error was parity or storage-protect; if an "illegal write", the data transfer continues normally but the illegal operation is suppressed.
2. The code 2 is loaded into the attention codes of both computers -- replacing any existing code present at the time. Thus, code 2 should not be used by the software for communication purposes. (Sometimes the hardware loads code 3 instead, so it too should be avoided by the software.)
3. Both attention bits are set, thereby interrupting both computers for notification of the error. The error condition within the Interface can only be cleared by a Blast command - presumably, in most cases the computers should suspend Interface operations and await human intervention.

2.2 Interface Programming

The programmer interacts with the Interface through the Status Register, whose bits can be variously set, cleared, or tested, and through the command repertoire, which is essentially symmetric to both computers with some variations to take advantage of special features.

The Interface Registers consist of 16 bits, numbered for reference 0 to 15. These bits correspond in the obvious way to the 16 bits of an 1800 word, and correspond to the right-hand 16 bits of the 18-bit PDP-7 word. (Thus, Interface register bit 0 corresponds to PDP-7 bit 2, bit 15 to bit 17.) Interface registers communicate with the accumulator on the PDP-7, and with both the accumulator and core storage on the 1800. The two high-order bits of the accumulator are set to zero whenever the PDP-7 reads an Interface register, and are ignored whenever it writes one.

2.2.1 Status Register



Notes:

* = interrupt-causing condition
S = may be individually set by programmer
R = may be individually reset by programmer
C = is reset by Clear Status command

Bit Ø - Operation-Complete (*RC)

Is set by Interface whenever a block transfer terminates normally, or whenever a Test & Set (Pending) takes effect. Causes an interrupt to 1800 and/or PDP-7 if corresponding Enable bit(s) are on. Is not set when a block transfer is stopped by the Halt command, or when stopped due to parity error or storage-protect violation.

Bit 1 - Enable 1800 (SRC)

When set (1), allows the Operation-Complete bit to present an interrupt to the 1800; if not set (0), 1800 will not be interrupted by Operation-Complete.

Bit 2 - Enable PDP-7 (SRC)

Allows Operation-Complete to interrupt the PDP-7 (analogous to Bit 1).

Bit 3 - Attention 1800 (*SRC)

When set, presents an interrupt to the 1800.

Bit 4 - Attention PDP-7 (*SRC)

When set, presents an interrupt to the PDP-7.

Bit 5 - 1800 Resolution (R)

Is set only by Test & Set command from the 1800, and only if Bit 6 is not set. It is a signal to the software that the 1800 has "logical control" of the Interface, but has no effect on the Interface

other than, if set, to prevent the setting of Bit 6.

Bit 6 - PDP-7 Resolution (R)

Signal that the PDP-7 is controlling the Interface; can be set only by Test & Set command from the PDP-7, and only if bit 5 is not set.

Bit 7 - Run (-)

Is turned on by the Interface whenever a block transfer is in progress.

Bit 8 - Transfer Direction (SRC)

Controls the direction of data movement during block transfers; 0 => PDP-7 to 1800, 1 => 1800 to PDP-7.

Bit 9 - Mode (SRC)

Determines whether block transfers operate in the 16/16 or 18/16 mode, i.e., whether word-for-word or packed mode is used.
0 => 16/16, 1 => 18/16.

Bit 10 - Unused (SRC)

This bit is unassigned.

Bit 11-15 - Attention Code (SR)

Each of these "bits" is actually represented by two flip-flops, one of which is set by the 1800 and read by the PDP-7, the other set by the

PDP-7 and read by the 1800. This section of the Status Register is therefore "full-duplex". Each computer can set or reset the bits it presents to the other computer, but can only read the bits presented by the other computer to it. The Clear Attention Code command resets the "outbound" Attention Code bits of the computer issuing said command.

2.2.2 Commands

Because of the symmetric design of the Interface, the following command descriptions apply to both the 1800 and the PDP-7, except as noted. Each command is represented by an IOT instruction on the PDP-7, and by the right-hand half of an IOCC on the 1800. READ commands cause data to be transferred from the Interface to the accumulator on the PDP-7, or to core storage at the address specified in the left-hand half of an IOCC on the 1800. WRITE commands are the inverse of READ, i.e., data goes to the Interface register(s) from either the PDP-7 accumulator or the 1800 core storage. The SENSE command exists on the 1800 only; it is identical to READ, except that data is routed to the accumulator instead of core storage, and the left-hand side of the IOCC is consequently ignored.

The commands which access the Interface registers are termed "programmed transfers", to distinguish them from "block transfers" which, once initiated, require no programmed intervention from either computer.

For compactness, the following abbreviations will be used:

A - 1800 Address Register

B - PDP-7 Address Register

C - Unit Count Register

D - Status Register

Y - The accumulator, for commands from the PDP-7; the core storage word addressed by the left half of the IOCC, for commands from the 1800.

CLEAR A
CLEAR B
CLEAR C

The designated register is cleared to zero.

CLEAR D

The Status Register is cleared to zero except for bits 5,6, and 7, which are unaffected.

SET A
SET B
SET C

The designated register is logically OR'ed with Y; the result appears in the register.

SET D

The Status Register is logically OR'ed with Y; the result appears in the Status Register, except for bits 0,5,6, and 7, which are unaffected.

RESET D

The Status Register is logically AND'ed with $\bar{Y}$ (the complement of Y); the result appears in the Status Register, except for bits 7 and 11 - 15, which are unaffected.

Note that Set D and Reset D are defined in such a way that a single mask can be used both to set and to reset bit(s) of the Status Register.

WRITE A
WRITE B
WRITE C
WRITE D

These commands are obtained by coding the bits for both Clear and Set in the same command; they take advantage of the fact that both Clear and Set can be performed in one Interface cycle, with the Clear operation preceding the Set.

READ A
READ B
READ C
READ D

The contents of the designated Interface register are stored in Y; the register is unaffected.

SENSE A
SENSE B
SENSE C
SENSE D

(1800 only) The contents of the designated Interface register are loaded into the 1800 accumulator.

CLEAR ATTENTION

The Attention Code (bits 11-15 of the Status Register) presented to the other computer is cleared to zero; the code read by the computer issuing the command is not affected. (Recall the dual nature of the Attention Code (see Section 2.1.1.))

TEST & SET

(PDP-7) Sets the PDP-7's resolution bit (bit 6 of the Status Register) if and only if the 1800's bit (bit 5) is not 1. If successful, causes the PDP-7 to skip the next instruction.

(1800) Sets the 1800's resolution bit (bit 5 of the Status Register) if and only if the PDP-7's bit (bit 6) is not 1. This command includes Sense D, so that the contents of the Status Register are automatically placed in the accumulator immediately after the test/set is performed. The programmer can then by shifting or masking determine whether the Test & Set was successful.

TEST & SET (PENDING)

(PDP-7) When, subsequent to the execution of this command, the 1800's resolution bit (bit 5 of the Status Register) becomes zero, the following bits of the Status Register are set: bit 0 (Op-Complete), bit 4 (Enable PDP-7), and bit 6 (PDP-7 Resolution). This effectively "seizes" the

Interface for the PDP-7 and notifies the latter by an interrupt. If bit 5 is zero when this command is issued, then bits 0,4, and 6 are set immediately and, in addition, the PDP-7 skips the next instruction.

(1800) When, subsequent to the execution of this command, the PDP-7's resolution bit (bit 6 of the Status Register) becomes zero, the following bits of the Status Register are set: bit 0 (Op-Complete), bit 3 (Enable 1800), and bit 5 (1800 Resolution). This effectively "seizes" the Interface for the 1800 and notifies the latter by an interrupt. If bit 6 is zero when this command is issued, bits 0,3, and 5 are set immediately.

START

Causes block transfers to commence as specified by the contents of the four Interface control registers.

HALT

Causes block transfers to stop the next time the Unit Count Register is decremented. The control registers will reflect the status of the transfer correctly, so that it may be resumed by a Start. The Operation-Complete bit is not set when a transfer is stopped in this way.

BLAST

Resets all Interface registers and control circuits to the zero or idle state. Any data transfer in progress will be halted immediately (and not necessarily correctly), and any Test & Set (Pending) will be cleared.

SET ATTENTION

This command is obtained by coding the bits for both Clear Attention and Set D in the same command. It allows the Attention Code to be set independently of the rest of the Status Register. (Note that the Attention bit can be set simultaneously, if desired.)

SKIP ON OP-COMPLETE

(PDP-7 only) Causes the PDP-7 to skip the next instruction if the Op-Complete bit (bit 0 of the Status Register) is set.

SKIP ON ATTENTION

(PDP-7 only) Causes the PDP-7 skip the next instruction if the Attention PDP-7 bit (bit 2 of the Status Register) is set.

SENSE INTERRUPT LEVEL

(1800 only) When issued from interrupt level 1 (recognition of which is wired into the Interface), this command causes the Interface to present interrupt-identification data to the 1800; this data appears in the accumulator as part of the Interrupt Level Status Word. The Interface presents the constant 4000_{16} if the Attention 1800 bit is on, 2000_{16} if the Op-Complete bit is on, 6000_{16} if both are on, and 0000 if neither is on. This data is OR'ed by the 1800 with similar data from other devices (if any) on interrupt level 1 to form the ILSW.

3. ENGINEERING DESCRIPTION

This section describes the Interface from the logic designer's point of view. A general introduction to the function of most important circuits in the Interface, it is intended primarily for maintenance personnel. However, the detailed execution sequences of some commands may occasionally be of help to the programmer. More detailed discussion, specifications, and timing information will be found in the Interface Maintenance Manual, published separately.

Logically, the Interface consists of four programmer-addressable registers (1800 Address, PDP-7 Address, Unit Count, Status) and one large Data or Shift Register, plus associated control circuits. Physically, the Interface contains 12 flip-flop registers: the five mentioned above, plus four Data Buffers (one in each direction for both the 1800 and PDP-7), one Command Buffer (for the 1800), and two Counters (to control the Shift Register). Digital Equipment Corporation M Series logic modules are used for these, and for all control and timing logic. Some W Series modules are used as level converters to the PDP-7; specially fabricated modules using DEC boards and IBM SLT Series circuits serve as level converters to the 1800. Most of the logic is supplied by internal power supplies, but small amounts of power are also drawn from each computer.

3.1 Programmed Transfers

Because the Interface, as an I/O device, is addressable by both the 1800 and the PDP-7, special timing problems arise. The two computers run asynchronous1 to each other, and it is not possible for an external device to stop the

clock of either machine. Since there is no way for a program running in either computer to determine whether the Interface is ready to respond to commands except by the issuance of a command, it follows that there must be at least one command to which the Interface can respond properly when issued by both computers simultaneously or in arbitrary time relation.

This requirement was met by designing the Interface to run asynchronously to both computers, under control of its own internal clock. Data movement within the Interface is timed by this clock; data flow between the Interface and either computer is largely timed by signals from the computer. Some of the Interface's logic circuit are dedicated to either the 1800 or the PDP-7, but common circuits are multiplexed during command execution so that they appear to be available to both computers simultaneously. As a result, the Interface can respond properly to simultaneous issuance of not only the "minimal" command (Read/Sense), but also any other pair of commands not inherently contradictory (e.g., two Write commands addressing the same register.)

3.1.1 Internal Timing

The Interface's basic clock is an 8MHz. free-running oscillator, which is started by manual depression of the panel START switch with the clock switch in the RUN position. The clock output complements a PHASE flip-flop every 125 ns., whose outputs are in turn connected to two pulse amplifiers. Each transition of the PHASE FF triggers one or the other PA, producing pulses of 110 ns. duration. These pulses, which stand in complementary phase relation to each other, are designated A PHASE and B PHASE. Thus, a 110ns A PHASE pulse occurs every 250ns.; a similar B PHASE pulse also occurs every 250 ns., but delayed 125ns. from A PHASE.

The outputs of the PHASE FF are also connected to two one-shots

each of which thereby produces a 60ns. pulse every 250ns. These pulses, delayed by 50ns. so as to fall squarely within the corresponding PHASE pulses, are designated A and B STROBE (see Figure 3.1). The A and B PHASE and A and B STROBE signals are distributed throughout the Interface through parallel-fed power NAND gates to preserve synchrony, waveform, and drive capability.

The PHASE and STROBE pulses are the basic multiplexing mechanism whereby the Interface executes both 1800 and PDP-7 commands simultaneously. During A PHASE, segments of 1800 commands are executed, while B PHASE is reserved for segments of PDP-7 commands. The overall rate is fast enough that even a multi-step command can always be executed within the "window" allowed by the issuing computer. Typically, an operation is set up by the PHASE signal (which acts as a level, rather than a pulse, relative to the STROBE) and clocked by the STROBE.

(The Block-Transfer circuits also make use of the PHASE and STROBE signals, but simply for timing rather than multiplexing. Those operations involve different circuitry, and proceed sometimes in parallel with the execution of programmed commands.)

Sequencing of operations during a particular 1800 or PDP-7 command is controlled by the 1800 or PDP-7 Period Counter, respectively. These "counters" are actually shift registers whose left-most bit is set to 1 upon receipt of a command from the corresponding computer. Every 250ns. thereafter the counter shifts the bit one position to the right, providing a sequence of distinct periods during the course of each command. (The use, if any, to which these periods are put depends, of course, on the command.) Eventually the bit shifts out the right-hand end of the register, which then remains zero until the start of the next command. The transition

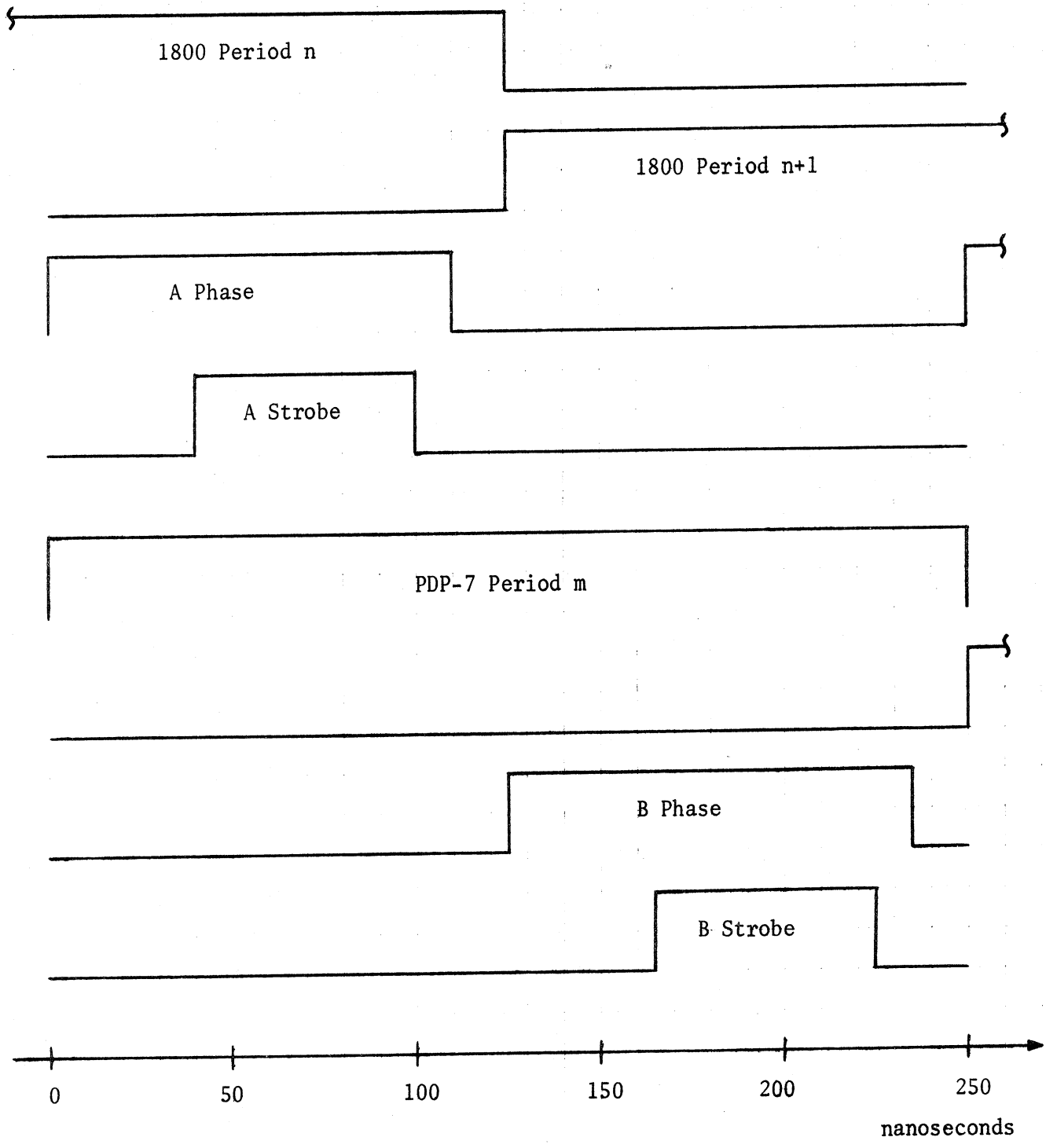


Figure 3.1: Schematic representation of Period, Phase, and Strobe timing relationships

from one period to the next is triggered by the "opposite" PHASE pulse, e.g., the 1800 Period Counter shifts during B PHASE, thereby ensuring that all gates enabled by the new period will have settled by the time the corresponding PHASE/STROBE come along.

3.1.2 External Timing: PDP-7

The PDP-7 addresses the Interface by means of "slow-cycle" IOT instructions of 3.75μsec. duration*. The Interface decodes the instruction directly from the Memory Buffer whenever the IOT signal line is asserted. (This signal, from the PDP-7's own instruction-decoding circuits, is not normally brought out to I/O devices.) Device selection codes 5X (i.e., 50 to 57) octal have been assigned to the Interface; therefore, any PDP-7 instruction of the form 7Y5XXX (octal), where Y = 0,1,2, or 3 and X = any digit, will be interpreted as an Interface command. (See Appendix B for bit assignments.) The IOP lines from the PDP-7 are ignored.

The conditions IOT·(device 5X)·A PHASE set the "PDP-7 Instruction" (PDP7-I) FF, which then causes the PDP-7 Period Counter to be initialized. Period Ø is not used; Periods 1,2,3, and 4 are used as required by the command. Successive A PHASE pulses clock the transitions from one period to the next; within a period, events happen on B PHASE and/or B STROBE. The fall of the IOT signal from the PDP-7 resets the PDP7-I FF, ending the execution sequence. Note: because the PDP-7 and Interface are asynchronous, Period 1 may begin anywhere between 150 and 400ns. after the assertion of IOT, depending on whether the latter just caught or just missed an A PHASE pulse. Logically, the PDP-7 communicates with the Interface control registers

---

* Normal-cycle IOT's, of 1.75μsec. duration, would work also. The PDP-7 is held permanently in the slow-cycle mode (for IOT's) by other attached equipment.

through its accumulator. Physically, the Data Buffers are interposed to guarantee sufficient settling time for the signal levels. On transfers to the PDP-7, the selected register is gated to the Interface-PDP-7 Data Buffer (I-PDP7DB) via the Interface Out Bus (IOB); the output of the I-PDP7DB is presented to the Data Collector bits 2-17 through level converters. After a sufficient settling interval (during which the 1800 may be accessing the Interface) the Interface asserts the STROBE AC IN line, causing the levels at the Data Collector to be set into the accumulator.

For transfers from the PDP-7, the low-order 16 bits of the Data Distributor are continuously connected to the PDP7-Interface Data Buffer (PDP7-IDB) through level converters. During Period 1 this data is clocked into the PDP7-IDB; on subsequent periods the data is transferred to the selected register(s) via the Interface In Bus (IIB).

3.1.3 External Timing: 1800

The 1800 addresses the Interface by means of the XIO instruction. Within the 1800, this instruction refers to a two-word Input-Output Control Command (IOCC), of which the right-hand word contains control information and the left-hand word a memory address (sometimes not used). Not counting fetch time, the instruction takes at least two memory cycles of 2μsec. duration each. During the first cycle, called Control Cycle, the control word (right half of IOCC) is placed on the 1800 OUT BUS while the CONTROL CYCLE line is asserted. The control word includes a device selection code along with other information specifying the operation to be performed (See Appendix B for bit assignments). The second cycle, Data Cycle, does not necessarily occur immediately after Control Cycle. When it does occur, the DATA CYCLE line is asserted and data is either placed on the OUT BUS or expected on the IN BUS, depending on the command. It is the responsibility

of the device to retain enough information from Control Cycle to respond properly during Data Cycle. (Note that this situation differs from the PDP-7, where the IOT instruction remains available on the MB throughout execution of the command - but at a considerable increase in cabling requirements.)

During Control Cycle, bits 0-4 of the OUT BUS contain the device selection code ("Area code" to IBM); the Interface has been assigned code 10100_2. Consequently, the Interface samples these bits whenever the condition CONTROL CYCLE·B PHASE·T4 is true. Recognition of the Interface selection code causes the "1800 Instruction" (1800-I) FF to be set, which then causes bits 5-15 of the OUT BUS to be strobed into the 11-bit Command Buffer (CB). In addition, this causes the 3-bit 1800 Period Counter to be initialized. Period Ø is not used; successive B PHASE pulses will advance (shift) the counter through periods 1 and 2, then clear it by shifting the bit out the right end.

For commands involving bit Set or Reset (e.g., Write A, Set D, Reset D), the 1800 Period Counter is re-initialized during Data Cycle. Thus, there can be as many as four distinct periods for command execution: periods 1 and 2 of Control Cycle, and periods 1 and 2 of Data Cycle.

The eight time pulses, TØ through T7, are generated as required by the Interface from the 1800's TIME PULSE A, B, and C lines. Synchronization between the 1800 and the Interface is generally controlled by these pulses.

The Interface registers communicate with the 1800 IN BUS and OUT BUS through the Data Buffers. For transfers to the 1800, contents of the selected register are transferred to the I-1800DB via the Interface Out Bus and the I-1800DB gated to the 1800 IN BUS, all during Data Cycle. Simultaneously, correct parity is generated and placed on the PARITY lines. For transfers from the 1800, the 1800 OUT BUS is strobed into the 1800-IDB; the

contents of the latter are then gated to the selected register(s) via the Interface In Bus during period 1 or 2, depending on function. At time T6 of Data Cycle, the 1800-I FF is reset, terminating the command sequence.

3.1.4 Read/Write: PDP-7

Read

The condition MB5(0)·MB9(0) is decoded as a Read command during period 1. This condition is AND'ed with the register selection bit (MB10, MB11, MB12, MB13 for A, B, C, and D registers respectively) to produce a level which enables the out-gating of the selected register. The next following B PHASE gates this data to the Interface Out Bus, where it is applied to the inputs of both I-1800DB and I-PDP7DB. However, B STROBE is applied to I-PDP7DB only, thereby clocking the register data into the Data Buffer.

Period 2 triggers a one-shot of 1.1μsec. duration which gates I-PDP7DB to the Data Collector, allowing plenty of time for the levels to stabilize. During period 4 a second one-shot of 260ns. duration is fired to pulse the STROBE AC IN line; this signal triggers a pulse amplifier (type W640, located physically within the PDP-7) which causes the Data Collector to set the accumulator. (Recall that bit 14, when set, causes the PDP-7 to clear the accumulator at the beginning of the IOT instruction - approximately period 1.)

If bit 4 is set on a Read command, the Attention PDP-7 FF (bit 2 of the Status Register) is reset during period 2.

Write

The commands Clear, bit Set, and bit Reset are all performed as part of the Write function. (As normally used by the programmer, Write includes both Clear and Set.) The Interface decodes MB5(0)·MB9(1) as a Write command during period 1, and if MB15 (Reset) or MB16 (Set) is on, also uses this period to strobe accumulator data from the Data Distributor into PDP7-IDB.

If MB17 (Clear) is set, the selected register(s) (as determined by MB10-13) are cleared during period 1. Note that any or all of the four control registers may be selected at the same time for Clear and/or Set functions, except that registers A, B, and C also require the condition RUN(0) for selection on a Write command. That is, Write operations on registers A, B, and C are suppressed while block transfers are in progress. The Reset function applies to the D(Status) register only, and in part accounts for differences in gating and flip-flop type between the D register and the others.

Bit Set and Reset operations are performed during periods 2 and 3, respectively. In-gating for the selected register(s) and out-gating for the PDP7-IDB are enabled; B PHASE places the contents of PDP7-IDB on the registers' inputs via the Interface In Bus, and B STROBE clocks the data into the register(s).

3.1.5 Read/Write: 1800

Read and Sense

These commands are treated similarly by the Interface, and in general are similar to the Read command from the PDP-7. The Interface decodes CB5(Ø), CB6(1), CB7(Ø) as Read and CB5(1), CB6(1), CB7(1) as Sense. CB8-11 provide register selection (see Appendix B). The contents of the selected register are gated to the I-1800DB via the Interface Out Bus at T1 of Data Cycle. During TØ-T6 of Data Cycle the contents of I-1800DB are placed on the 1800 IN BUS and the PARITY line for each byte asserted as appropriate. (Because the 1800 samples its IN BUS at T3, the transfer proceeds correctly even though the initial contents of I-1800DB, at TØ, are incorrect. This strange - to say the least - timing is the result of a design change to correct another problem, and is accepted purely for pragmatic reasons.)

Write

As with the PDP-7, the 1800 commands Clear, Set, and Reset are all part of the basic Write function. The Interface decodes CB5(0), CB6(0), CB7(1) as Write, while CB12-15 specify the sub-function. The Clear (CB15) and Clear Attention Code (CB12) operations are done during period 2 of Control Cycle. Data is available on the 1800 OUT BUS at T4 of Data Cycle, and is strobed into the 1800-IDB by a pulse delayed some 75-225ns. from T4. If Set (CB14) or Reset (CB13) are specified, the I-1800 DATA FF is set at T4 of Data Cycle, causing the Period Counter to be re-initialized. Bit Set operations are done during period 1 of Data Cycle, bit Reset during period 2. The PERIOD, A PHASE, and A STROBE signals are used analogously to PERIOD, B PHASE and B STROBE for the PDP-7. The restrictions mentioned for PDP-7 Write operations also hold for 1800 Write; of course, Reset applies to the D register only, and the A, B, and C registers are protected against Write commands when the RUN FF is on.

3.1.6 Miscellaneous

Status Register

The Status ("D") Register can be considered to consist of one 11-bit register and two 5-bit registers, of which either computer can access at most 16 bits at a time. The 11-bit section is a collection of control flip-flops, all of which can be read but only some of which can be set and/or cleared under program control. See Appendix B for specifics.

Each of the 5-bit "Attention Code" sections can be cleared and set only by one computer, and read only by the other. When the 1800 writes or clears the Status Register, it alters the PDP-7 Attention Code Register; but when it reads the Status Register, bits 11-15 come from the 1800 Attention Code Register, and conversely for the PDP-7. The Attention Code Registers

provide two one-way communication paths.

The construction and gating of the Status Register are different from the A, B, and C Registers because

1) its bits can be individually set and/or reset;

2) its bits are used in various ways to control other circuits;

3) it can be accessed (i.e., altered) not only by both computers but also by the Interface itself.

Resolution Circuits

The PDP-7 Test & Set command is decoded from MB5(1)·MB13(1) and is performed during period 1. The signal B PHASE is AND'ed with the condition DR5(0)·DR6(0), i.e., both resolution bits off, and if the result is true then B STROBE sets DR6. Simultaneously the PDP-7 SKIP line is pulsed, performing the "skip on success" part of the command.

Test & Set from the 1800 is accomplished similarly. The command is decoded as an option on the Sense D command (CB13(1)) and is performed during 1800 period 1 of Control Cycle. The A PHASE signal is AND'ed with DR5(0)·DR6(0), and if true, A STROBE sets DR5. Since the Sense command gates the selected register to the I-1800DB during Data Cycle, the result of the Test & Set operation is immediately available to the 1800 programmer.

Because the 1800 tests and (maybe) sets only during A PHASE, and the PDP-7 only during B PHASE, the operation is always resolved unambiguously in favor of one machine or the other.

The Test & Set (Pending) command is performed whenever the regular Test & Set is decoded (as above) with, in addition, MB12 (PDP-7) or CB14 (1800) set. From the PDP-7 the PDP7 T&S PENDING FF is set during period 1, any phase. This FF is AND'ed with DR5(1) whenever the condition is true, i.e., 1800 resolution bit is off, a pulse amplifier is fired which

sets DRØ, DR4, and DR6 (Op-Complete, Enable PDP-7, and PDP-7 Resolution). A second pulse amplifier then resets the PDP7 T&S PENDING FF.

Similarly, this command from the 1800 sets the 1800 T&S PENDING FF during period 1 of Control Cycle, any phase. The FF is AND'ed with DR6(0) to produce a condition which sets DRØ, DR3, and DR5 (Op-Complete, Enable 1800, and 1800 Resolution); the FF is then reset as above.

Since the immediate form of the command is included within Test & Set (Pending), there is no ambiguity even if the latter command is issued simultaneously by both computers. Only one computer "seizes" the Interface and is interrupted. The loser's request remains pending. Should this command be issued by the PDP-7 when the 1800 Resolution bit is not on, the skip will occur in addition to the interrupt.

Blast

The Blast command, decoded from MB5(1)·MB16(1) on the PDP-7 or as a "control" function (CB5(1), CB6(Ø), CB7(Ø)) with CB15 set on the 1800, causes the Interface RESET line to be pulsed. All registers and counters are set to zero and all flip-flops are set to their initial state. (The Op-Complete bit, DRØ, is normally set by the clearing of the Run FF, DR7, at the conclusion of a block-transfer. Circuit delays here outlast the Blast-induced reset pulses, so that if Blast is issued while Run is on, Op-Complete will be set. A second Blast takes care of this problem.) The same RESET line pulsed by Blast is also pulsed from the PDP-7 POWER CLEAR line (activated by a Clear All Flags IOT instruction or the power-on sequence), the 1800 D.C. RESET line (activated by the power-on sequence or depression of the Reset button on the 1800 panel), and the Reset lever on the Interface panel. A Reset never stops the Interface basic clock, however.

The Blast is done during PDP-7 period 1 or 1800 period 1 of Control Cycle. This command self-destructs.

Start and Halt

These commands are discussed in Section 3.2.

Skip

The two PDP-7 skip commands are decoded from MB5(1) and either MB10(1) (Skip on Op-Complete) or MB11(1) (Skip on Attention). During period 1 MB10(1)·DRØ(1)+MB11(1)·DR2(1) causes the SKIP line to the PDP-7 to be pulsed, incrementing the PDP-7 Program Counter.

Interrupt

Interrupting the PDP-7 is simple: the condition DRØ(1)·DR4(1)+DR2(1) causes the INTERRUPT line to the PDP-7 to be asserted.

1800 devices can present interrupt requests only in response to a polling signal from the CPU. Moreover, a device must be prepared to provide some identification information in addition to the simple interrupt request; this frees the 1800 - unlike the PDP-7 - from the need to interrogate each device in turn.

The Interface has been assigned to interrupt on priority level 1; therefore, the Interface AND's the condition DRØ(1)·DR3(1)+DR1(1) with the INTERRUPT POLL line from the 1800, and if the result is true, asserts 1800 IN BUS 1. (Level Ø devices would assert IBØ, level 2 IB2, and so forth.) Eventually the CPU will recognize the interrupt request, and the interrupt-handling subroutine will issue a Sense Interrupt Level Status Word (Sense ILSW) command. This command is addressed to all devices on the currently recognized priority level, and hence has no device-selection code; it is recognizable only by virtue of its "function" code (bits 5,6, and 7), which is $3(011_2)$. Since the devices have no way of knowing which priority level the CPU is currently

honoring, that information is also coded in bits 11-15 of the command (all this appears on the OUT BUS during Control Cycle).

Just how, one might ask, is this information coded? A very good question, one which moves the authors to acknowledge the admirable cleverness with which IBM has concealed this information in the (ir-) relevant publications - if it is there at all. An empirically derived best guess is that the code is the same as the address of the memory location through which the interrupt forced-branch occurs. That is, each priority level is assigned a word in a low-core transfer vector; the word for level 1 is at address $000C_{16}$, and a Sense ILSW from this level produces 01100_2 in bits 11-15 of the control word.

The Interface, therefore, decodes these bits as a sort of secondary device-selection, during T4 of Control Cycle. The OUT BUS is strobed into the Command Buffer for display only; and the Period Counter is not initialized. Instead, a SILSW FF is set. During TØ-T5 of Data Cycle the Interface asserts IN BUS bit 1 if the Attention condition (DR1(1)) is true, and/or bit 2 if the Op-Complete condition (DRØ(1)·DR3(1)) is true. These two bits are OR'ed into the Interrupt Level Status Word for level 1 by the 1800 CPU; the operating system, TSX, interprets these as though from two separate devices. The SILSW FF is reset at T6 of Data Cycle, ending the command sequence.

3.2 Block Transfers

In block-transfer operations the Interface alternately requests cycle-steals from the 1800 and data-breaks from the PDP-7 (or vice versa, depending on direction of transfer). Each data word is read (stored)

at an address specified by the 1800 Address Register or the PDP-7 Address Register, and passes through the Interface via the Shift Register. The total number of words transferred is controlled by the Unit Count Register. The procedure is described in detail on the following pages.

The reader should be aware that the core storage location addressed during an 1800 cycle-steal is controlled by the Channel Address Register (CAR). (Each cycle-steal priority level - not to be confused with interrupt priority level - has its own CAR. The Interface uses cycle-steal priority level zero, interrupt priority level one.) During the initialization sequence the contents of the 1800 Address Register are transferred to the CAR; both are incremented (AR by the Interface, CAR by the 1800) on every cycle-steal thereafter, so that they "track" each other. But the actual addressing of memory is done by the CAR.

A list of abbreviations used in the following descriptions will be found at the end of Section 3.

3.2.1 Initialization

1. Start (GO) sequence

A PDP-7 Start command is decoded from MB5(1)·MB15(1); the 1800 Start is a "control" function with bit 13 set. Either command sets the GO FF during its respective period 1. As soon as the Interface is not busy with a command from either machine, i.e., when the condition GO(1)·$\overline{\text{PDP7-I}}$·$\overline{\text{1800-I}}$ obtains, the GO' FF is set, producing the GO pulse via a pulse amplifier. (The GO' FF is used to prevent redundant GO pulses; logic á GO-GO doesn't work too well...)

The GO pulse does the following:

a) gates the 1800 Address Register to the Interface-1800 Data Buffer;
b) clears the A and B Counters;
c) clears the Shift Register;
d) requests a cycle-steal from the 1800 (by setting the CS REQ FF, which asserts the CS REQ line).

GO => AR → I-1800DB

0 → A CNTR, 0 → B CNTR

0 → SR

1 → CS REQ FF

2. When the 1800 grants the cycle-steal (in response to the CS REQ), it asserts the CS ACK line. ACK signals the Interface to start the LOAD CAR sequence, in which the contents of the I-1800DB (loaded from AR at step 1(a) above) are gated to the 1800 Channel Address Register assigned to the Interface. This transfer is accomplished as follows:

a) ACK•GO(1) places the contents of the I-1800DB on the 1800 IN BUS, and asserts CS CONTROL 1 and 2 during T1-T6 of the CS cycle. This signals the 1800 to load the CAR from the IN BUS.

I-1800DB → 1800IB

1 → CS 1, 1 → CS 2

1800IB → CAR (within 1800CPU)

b) During this same CS cycle the A or B Shift Counter is loaded (both were set to zero at step 1(b) above) as determined by the transfer direction and mode:

ACK•GO(1)•T1•TD(0) => 16 → A CNTR

ACK•GO(1)•T1•TD(1)•MODE(0) => 16 → B CNTR

ACK•GO(1)•T1•TD(1)•MODE(1) => 18 → B CNTR

At time T1 of this and every CS cycle, CS REQ is dropped to prevent occurrence of a second CS cycle.

ACK•T1 => 0 → CS REQ FF

3. At the beginning of T3 of the GO sequence CS cycle, the RUN FF is set. The trailing edge of T5 resets the GO and GO' FF's, thus ending the GO sequence.

ACK•GO(1)•T3 => 1 → RUN FF

ACK•GO(1)•T6 => 0 → GO, 0 → GO' FF's

3.2.2 Serial/Parallel Data Flow

1. Regular Direct Memory Access (DMA) operations commence with the setting of the RUN FF at the end of the GO sequence. Data transfers between the Interface and the 1800 and between the Interface and the PDP-7 occur in parallel, with path widths of 16 and 18 bits respectively. Within the Interface, data move serially through the Shift Register under control of the A and B Counters. The important events during this process can be concisely expressed as follows:

while RUN = 1

A CNTR = 0 => 1 → 1800 CS REQ FF

16 → A CNTR

B CNTR = 0 => 1 → PDP7 BRK REQ FF

n → B CNTR

(n = 16 or 18, depending on MODE)

(A CNTR = 0)•(B CNTR = 0) => decrement CR

CR = 0 => 0 → RUN FF, 1 → OP-COMPLETE FF

Additional logic, not shown above, is employed to synchronize the shifting and DMA operations and to handle initiation and termination special cases. More detailed operation of these circuits can probably best be illustrated by example, so the following discussion assumes a transfer from the 1800 to the PDP-7 in 18/16 mode.

2. During the GO sequence, the RUN FF is set and the A CNTR cleared. RUN(1)•(A CNTR = 0) places a level on the input to the CS REQ ENB FF, which is clocked by the trailing edge of the next following B STROBE. The setting of this FF produces two pulses which

a) load the A CNTR with 16

b) set the 1800 CS REQ FF
16 → A CNTR
1 → 1800 CS REQ FF

3. The CS REQ FF asserts the CS REQ line to the 1800, which replies with CS ACK. This ACK signals the Interface to begin a CS Write Cycle (i.e., out of the 1800). At T1 the CS REQ FF is cleared; at T4' the 1800 OUT BUS (available at T4-T7) is gated to the 1800 side of SR. The leading edge of T5 clocks the OB data into SR_{18-33}. After a 300ns. delay to allow for settling, an 1800-Interface Data Completed pulse (from the trailing edge of T5) sets the 1800 DATA EX OVER FF, which increments the AR and places a level on the "reset" input of the 1800 CS REQ ENB FF. The trailing edge of the next B STROBE' resets this FF, which in turn resets the 1800 DATA EX OVER FF. This completes the transfer of the first data word from the 1800 to the Interface Shift Register.

$$ACK \cdot \overline{GO} \cdot TD(1) \cdot (T4') \Rightarrow OB \rightarrow SR_{18-33}$$

4. Shift pulses from the 4 MHz basic clock (A STROBE) are enabled by the conditions (1800 CS REQ ENB FF = 0)·(PDP7 BRK REQ ENB FF = 0) and RUN·$\overline{GO}$. In addition, the MODE FF determines whether bits shifted out of position 33 of SR will be shifted in at position 0 or 2 (recall that SR is actually a circular register); MODE(1) enables shifting into SR bit 0, while MODE(0) enables shifts to SR bit 2. Approximately 100ns. after being enabled, the first shift pulse reaches the SR. (Note that the RUN·$\overline{GO}$ condition is present during almost all of a block transfer; therefore the determining factors with respect to shifting are the CS/BRK REQ ENB FF's, which are reset at the completion of every cycle-steal or data-break.) Each shift pulse (A STROBE') causes the entire SR to shift to the right one bit. Corresponding to each shift pulse is a decrement-counters pulse (A STROBE') which decrements both A and B Counters.

During shifting,

A STROBE' => $SR_i \rightarrow SR_{i+1}$, $SR_{33} \rightarrow SR_0$ or SR_2 depending on MODE

B STROBE' => decrement A CNTR, B CNTR

5. After 16 shifts, the A CNTR will have been decremented to 0, the B CNTR to 2 (recall that with the assumed TD and MODE, the B CNTR was initialized to 18 during the GO sequence). The condition (A CNTR = 0) places a level on the "set" input of the CS REQ ENB FF; the latter is clocked by B STROBE' whereupon further shifting is inhibited and the CS REQ FF set.

(A CNTR = 0)·B STROBE' => 1 → CS REQ ENB FF

1 → CS REQ FF

6. The 1800 responds to CS REQ with CS ACK and the second data word. (Each cycle-steal automatically increments the CAR, which determines the memory location to be accessed during the cycle-steal cycle.) This second data word is loaded into SR_{18-33} and the AR incremented as described in paragraph 3 above. Completion of the cycle-steal cycle allows shifting to resume, and after two shifts the B CNTR contains 0, the A CNTR 14.

7. The condition (B CNTR = 0) sets the 7 BRK REQ ENB FF and, after a 250ns. delay to allow settling of the SR_{0-17} data at the input of the PDP-7 multiplexer, sets the 7 BRK REQ FF and reloads the B CNTR. In contrast to the situation with the 1800, the contents of SR_{0-17} and BR are continuously presented (through level converters) to the PDP-7 Data Multiplexer. During the break cycle granted in response to the assertion of BRK REQ, the multiplexer samples its DATA IN lines and places their contents in the Memory Buffer; it also samples its ADDRESS IN lines and gates their contents to the Memory Address Register. The Interface TD FF (bit 8 of the Status Register), which in this example is set, causes the PDP-7 multiplexer's TRANSFER DIRECTION line to be asserted, so that during the break cycle the contents of the Memory Buffer are stored at the location specified by the Memory Address Register.

(B CNTR = 0) => 1 → 7 BRK REQ ENB FF

(7 BRK REQ ENB FF = 1) => 1 → 7 BRK REQ FF

n → B CNTR

where n = 16 or 18, depending on MODE

8. The Data Multiplexer returns a DATA ACCEPTED pulse to the Interface while strobing the SR data into the Memory Buffer. If the DATA ACCEPTED pulse is taken as the signal to resume shifting, errors occur due to skew. Consequently, DATA ACCEPTED sets a DATA ACC FF within the Interface, which is AND'ed with PDP-7 timing pulse T6 to produce a pulse which

a) increments the BR, and

b) sets the 7 DATA EX OVER FF

DATA ACCEPTED => 1 → DATA ACC FF

DATA ACC(1)•T6 => increment BR

1 → 7 DATA EX OVER FF

This guarantees that the SR remains undisturbed for some 500ns. after receipt of DATA ACCEPTED.

9. The setting of the 7 DATA EX OVER FF resets the 7 BRK REQ ENB FF which, with the condition (1800 CS REQ ENB FF = 0), allows shifting to resume. After 14 shifts, the A CNTR will have been decremented to zero, whereupon another 1800 cycle-steal occurs as described in paragraphs 5 and 6 above. This sequence continues, with 1800 cycle-steal occurring every time the A CNTR reaches zero and PDP-7 data breaks occurring whenever the B CNTR becomes zero.

10. Whenever both Counters are zero simultaneously (after 9 1800 words, 8 PDP-7 words), the Unit Count Register (CR) is decremented and the entire process repeated.

(A CNTR = 0)•(B CNTR = 0) => decrement CR.

11. When the Unit Count Register reaches zero, the RUN FF is reset and the Op-complete bit (bit 0 of the Status Register) is set. Because this happens during the last data break, special provision is made to ensure that the latter is completed properly.

(CR = 0) => 0 → RUN FF

1 → DRØ

12. A transfer in the other direction, from the PDP-7 to the 1800, is handled in much the same fashion. The GO sequence initializes the A CNTR to 16, the B CNTR to 0, at 1800 time T1; at T3 the RUN FF is set, whereupon the condition RUN(1)·(B CNTR = 0) causes the 7 BRK REQ ENB FF to be set and the B CNTR reloaded. During the ensuing data break cycle, the PDP-7 loads its Memory Buffer with the contents of the location addressed by the BR and returns a DATA READY pulse to the Interface. The DATA READY pulse sets a DATA READY FF, which gates the Memory Buffer output to the SR_{0-17}; at PDP-7 time T6 this data is clocked into the SR. This use of T6 for clocking provides about 500ns. settling time after the receipt of DATA READY. Note that this sequence differs from that of paragraph 7 (above) because the condition TD(0) causes the Data Multiplexer TRANSFER DIRECTION line not to be asserted.

13. The clearing of the 7 BRK REQ ENB FF at the end of the break cycle allows shifting to begin again. After the SR has shifted right 16 positions, the A CNTR becomes zero, thereby causing the 1800 CS REQ FF to be set. When the 1800 responds to CS REQ with CS ACK, the Interface places the contents of SR_{18-33} on the 1800 IN BUS during time T1-T5. The Interface also computes and sends to the 1800 two parity bits (one for each byte of data), and asserts CS CONTROL line 2. The latter signals the 1800 to perform a CS Read cycle, in which the IN BUS data is stored at the memory location addressed by CAR, after which the CAR is incremented.

14. The 16/16 mode is identical to the 18/16 mode, except that the B CNTR is always loaded with 16 instead of 18. This means that both A and B Counters decrement to zero simultaneously, with the result that

a) 1800 cycle steals and PDP-7 data breaks occur at approximately the same time, and

b) The Unit Count Register (CR) is decremented once per 16-bit word rather than once per 144-bit block.

As a special feature during 16/16 mode transfers into the PDP-7, the high-order bit of the 16-bit word - bit 2 of the SR - is fanned out to bits 1 and 0, thereby "propagating the sign" so that twos complement negative numbers retain their value when mapped from 16 bits to 18 bits.

3.2.3 Abnormal Termination

Halt

The Halt command is decoded from MB5(1)·MB17(1) on the PDP-7, or from a "control" function with bit 14 set on the 1800. Either command sets the HALT PENDING FF during its respective period 1. HALT PENDING(1) is AND'ed with the next succeeding DECREMENT CR pulse; the resulting pulse resets the RUN FF, stopping further block-transfer activity. The A,B,C, and Shift Registers are left in the proper state, however, for the block-transfer to be resumed correctly by a START command. The resetting of the RUN FF in turn resets the HALT PENDING FF.

Fault

A fault condition exists when either

a) A parity error or storage-protection violation occurs on the 1800 during Interface operations, or

b) either computer attempts a Write (i.e., Set and/or Clear) command on the A,B, or C registers while the RUN FF is on.

Occurrence of any of the above conditions sets one of four error FF's within the Interface (PARITY ERROR, STOR PROT, PDP-7 ILLEGAL WRITE, 1800 ILLEGAL WRITE); the output of these FF's is OR'ed together and, as soon as the Interface is not executing a command from either computer (i.e., $\overline{\text{PDP7I}}\cdot\overline{\text{1800I}}$), sets both Attention bits in the Status Register and forces a code of 2 or 3 into both Attention Code Registers. If the parity or store-protect error occurred during a block-transfer operation (the most likely situation), the RUN FF is reset, stopping the transfer immediately. Since these faults

generally require human intervention, the fault FF's can be displayed on the Interface panel; but they can be reset only by a Blast command or manual reset.

Table 3.2: Signal Cables

Signal Names	No. of Lines
I. PDP-7 → Interface	
Buffered Accumulator 2-17 (0,1 not used)	16
Buffered Memory Buffer 0-17	18
T6, DATA ACCEPTED, DATA READY, POWER CLEAR, IOT	5
	39
II. Interface → PDP-7	
Data Collector 2-17 (to accumulator)	16
Data Address 3-17 (to multiplexer)	15
Data In 0-17 (to multiplexer)	18
SKIP, INTERRUPT, BREAK REQUEST, TRANSFER DIRECTION, STROBE AC IN	5
	54
PDP-7 Total:	93
III. 1800 → Interface	
OUT BUS 0-15	16
DATA CYCLE, CONTROL CYCLE, TIME PULSE A, TIME PULSE B, TIME PULSE C, PARITY ERROR, INTERRUPT POLL A, STORAGE PROTECT VIOLATION, CYCLE STEAL ACKNOWLEDGE, D.C. RESET	10
	26
IV. Interface → 1800	
IN BUS 0-15	16
PARITY 0-7, PARITY 8-15, CYCLE STEAL CONTROL Ø, CYCLE STEAL CONTROL 1, CYCLE STEAL CONTROL 2, CYCLE STEAL REQUEST	6
	22
1800 Total:	48

Abbreviations used in Section 3:

1. General

FF = Flip-flop
REG = register
CNTR = counter
DMA = Direct Memory Access
CS = cycle-steal (1800 DMA)
BRK = (data) break (PDP-7 DMA)
REQ = request
ENB = enable
A = the 1800, or pertaining thereto
B = the PDP-7, or pertaining thereto

2. Registers

AR = 1800 Address Register
BR = PDP-7 Address Register
CR = Unit Count Register
DR = Status Register
SR = Shift (Data) Register
MB = PDP-7 Memory Buffer
CAR = 1800 Channel Address Register
I-1800DB = Interface - 1800 Data Buffer
I-PDP7DB = Interface - PDP7 Data Buffer
1800-IDB = 1800 - Interface Data Buffer
PDP7-IDB = PDP7 - Interface Data Buffer
CB = Command Buffer
A CNTR = A Shift Counter
B CNTR = B Shift Counter
Xn = the nth bit of register X (where bit Ø is the leftmost)

3. Logic

P•Q = event - P and event - Q
P+Q = event - P or event - Q
$\overline{P}$ = not P (complement of P)
P => Q = event - P causes event Q
X → Y = contents of register or signal line(s) X are gated to register or signal line(s) Y
n → Y = the constant n is gated to register or signal line(s) Y

4. Signal Lines

X(1) = the "1" output of flip-flop X
X(Ø) = the "Ø" output of flip-flop X
Xn = the n^{th} line of a set of signal lines
IB = 1800 IN BUS
OB = 1800 OUT BUS
IIB = Interface IN BUS
IOB = Interface OUT BUS
ACK = (cycle-steal) Acknowledge

5. Flip-flops

MODE = DR9 (16/16 or 18/16 bit)
TD = DR8 (transfer direction)
RUN = DR7 (block-transfer in progress)

6. Timing

Signals sightly advanced or delayed from "standard" timing signals are denoted by a prime, e.g., T4' is related to, but not identical with, T4. Details will be found in the maintenance manual.

4. PROGRAMMING SUPPORT

4.1 Philosophy and Structure

There are three basic kinds of operations that a program may wish to perform using the Interface: block transfer, issuing or handling attention interrupts, and seizure of control over the Interface prior to the performance of an extended control sequence. These three functions are common to both machines; either may initiate any of the control sequences at any time. Since the hardware was designed to be controlled almost exclusively by software (rather than containing hardware checks to lock out certain sequences), programs must adhere to a consistent set of conventions in order to allow each computer equal access without fear of interference by the other.

To this end, all Interface operations are performed through system subroutines.

Section 4.1.1 describes the basic conventions used, the structure behind the system routines, and some of the lowest level routines. Section 4.1.2 describes the features common to the routines in both machines for performing the three basic operations. Sections 4.2 and 4.3 describe the implementation of the programming support for each machine separately, including detailed calling sequences for the routines described in 4.1.1 and 4.1.2.

4.1.1 Basic conventions and low level routines

There are two conventions necessary to maintain control over the Interface. The first, already mentioned, is necessary so that the system will know at all times the state of the Interface: all Interface operations are performed through system subroutines. The second is: neither machine will alter any of the Interface registers (except for attention code processing) unless it has logical control of the Interface. This ensures that Interface operations will not have to contend with interference from the other machine. The second convention imposes the following standard operating procedure. First, if the Interface is under control of machine A, machine B must wait for A to relinquish control (and vice versa) before it (B) may do anything with the registers. Secondly, the controlling machine has the responsibility of releasing control as soon as it is finished, to allow the other machine access.

Both user and system routines are expected to operate under these basic conventions. If the user performs operations only through the system routines, according to the rules below, he will implicitly obey these conventions. Failure to observe them will certainly cause incorrect results, and may even cause full scale system disaster, such as destroying the contents of a disk.

In order to provide the maximum amount of processing power and Interface control at the same time, Interface operations are performed under interrupt control. This allows operations such as block transfers to be performed concurrently with other program operations.

Interrupts are of two types: attention and op-complete. Attention interrupts are handled in a straightforward manner. Issuing an attention

may be performed immediately and completely asynchronously with all other operations since there are bits in the Interface Status Register used only for attentions. Receiving attentions consists merely in reading the appropriate part of the Status Register. Section 4.1.2 discusses the routines for performing these functions. No deep structure is necessary.

Op-complete interrupts may result from two different operations: as the result of the completion of a transfer, and as the result of a Test & Set (Pending) instruction when control is finally assigned to the requesting computer. These cases correspond to user requests for block transfers or for Interface control. After the user program has made its request, either of these operations may have to wait for completion if the other machine has control or if there are other operations already being performed by the same machine.

To allow concurrent processing when an operation may not be performed immediately, the system maintains a queue of unfilled requests. When the system receives a request which it can not execute immediately, it puts the request on the Interface Function Queue (IFQ). The IFQ consists of a list of Interface Control Blocks (ICBs). The ICB for an operation contains all the information needed to perform that operation. Once the machine gains control of the interface, it takes the first ICB from the IFQ and starts the operation defined by that ICB. When the operation is finished, the next operation on the IFQ is started according to the information contained in its ICB, and so forth. When the end of the IFQ is reached, the machine relinquishes Interface control.

There are two kinds of ICBs (although the structure is general), corresponding to requests for transfers and for control. The user calls the appropriate routine with one parameter, the address of the ICB

containing the pertinent information. The system routine adds additional information to the ICB, calls another routine which enters it on the IFQ, and then returns control to the user's program. Part of the information in the ICB is status information maintained by the system. The user may inquire about the status of the operation by calling a "test" routine which interprets the status information of the ICB given to it as a parameter. The "test" routine then returns an indication to the user's program.

Thus, normal user operating procedure for either transfers or for control requests is to call the appropriate subroutine, and then to wait for completion of the operation by repeated calls on the "test" subroutine.

The ICB for a control request consists of two words, and the ICB for a transfer consists of seven words. Both words of the ICB for control requests and the first two words of the transfer ICB are used for system information. Thus, for control requests, the user merely supplies the address of a two-word block the system can use. For transfers, the user fills in the last five words of the ICB with the information specifying the transfer before he calls the transfer subroutine. These last five words are described in Section 4.1.2.

The first word of the ICB is set to a special value when it is inserted on the IFQ, indicating that the function requested has not yet been completed. When the function is completed, the first word is set to another special value. (This word is also used as a link pointer to the next ICB on the queue, if any.) The second word of an ICB is set by the first level routine called by the user; i.e., it is set by either the transfer routine or by the control request routine. The second word contains the address of a subroutine to be called to initiate the operation represented by the ICB. Thus, at some later time when the ICB rises to the top of the queue, the

IFQ processor knows what routine to call to start the operation.

The action of this initiation subroutine is slightly different for the two kinds of requests. For transfers, the initiation subroutine actually starts the transfer based on the information contained in the rest of the ICB. The Status Register is set so that when the transfer is completed, an op-complete interrupt will occur. The IFQ processor will then be reentered, the ICB of the transfer just completed with be marked as finished, the next ICB taken from the list, and the process repeated.

For Interface control requests, however, the initiation subroutine does not start any operation, and, as a result, no op-complete interrupt occurs. (The IFQ processor must be restarted as specified below.) The "initiation" routine merely marks the ICB for the control request as finished and returns. At this point in time, Interface control still resides with the machine, but no Interface operation is being performed. The user's program, which has been waiting for the indication that the operation is finished, will now receive that confirmation. The net result is that Interface control is in the hands of the user. He may then direct operation of the Interface (within the limits defined in Section 4.1.2). When the user is finished with his control sequence, he must call a system routine to relinquish control. This subroutine then restarts the IFQ processor as if an operation-complete interrupt had occurred. The whole process is then repeated for the next ICB on the queue.

This takeover of Interface control by a program allows for a more extensive communication sequence than the simple functions of block transfer and attention interrupt. For example, once a program has gained control, it may load any of the control registers A, B, and C, and then issue an attention to the other machine. This allows a transfer of more information than just the attention code, and also allows an acknowledgment or other transfer of information from the other machine.

4.1.2 Programmable operations

The previous section described the general outline of the Interface control protocol. It also described in general terms the process of putting an Interface Control Block on the Interface Function Queue and the method for determining the status of the operation represented by the ICB. Specific descriptions of these two routines are found in Sections 4.2 and 4.3.

This section amplifies the descriptions for routines to accomplish block transfers, attention interrupts, and control seizures. A table at the end of the section supplies the equivalences between the functions described here and the actual names of routines on the two machines to accomplish the functions.

ATTENTION INTERRUPT

An attention interrupt is sent for the purpose of transferring a small amount of information (about five bits) to the other machine, immediately. It may be issued at any time, regardless of location of control or of the current operation being performed. The subroutines for issuing the attention are simple, taking one parameter, the number to be sent. Return is made immediately after the attention is sent.

The complementary function, receiving an attention interrupt, is only slightly more complex. A program which wishes to handle an interrupt calls a system routine with two parameters: the number of the attention, and the address of an interrupt time subroutine which is to be called when the attention is received. The subroutine thus called may perform only those actions allowed an interrupt routine on its machine. The method of return from this subroutine is specified in the detailed descriptions of 4.2 and 4.3. There are also methods for resetting the attention handling

for one or all attention interrupt numbers when a program no longer wants to handle that number or numbers.

The system routines maintain a table of all possible attention interrupt numbers. When the user declares that he will take an interrupt, an entry is made in the table to that effect. When an attention actually occurs, control is dispatched through the table. If there is no processor for an attention number, a message is printed on the printer and the attention ignored (1800), or the machine halts (PDP-7).

BLOCK TRANSFER

As stated in Section 4.1.1, block transfers are initiated by calling a system subroutine with one parameter, the address of a seven-word Interface Control Block. This block contains all the information for specifying the transfer. No verification is made of the suitability of the user-specified addresses in the ICB; it is the responsibility of the initiating program to ensure the appropriateness of the transfer. The system routines will safeguard the interface conventions by first seizing control of the interface before starting the transfer. The system routines will also set the first two words of the ICB and enter the block on the IFQ.

The user must set words 3-7 of the ICB. These words should contain:

3. The address of an op-complete subroutine.
4. The address of the 1800 data area.
5. The address of the PDP-7 data area.
6. The block count (ie., the number of words if 16/16 mode transfer, or the number of 8/9 blocks if 18/16 mode).

7. An encoding of the direction of transfer and the mode of the transfer.

Words 4-7 simply specify the actual transfer operation. As stated in Section 4.1.1, the normal method for determining whether a transfer has been completed is to call a "test" subroutine with the ICB of the transfer as a parameter. The user, however, may wish to perform some operation as soon as the operation is completed, at interrupt time. If this is the case, he may specify the address of a subroutine to be called in word three of the ICB. If he does not wish to do this, he must set this word to zero. The subroutine thus called is an interrupt routine and may perform only those operations allowed an interrupt routine on the respective machines. It may even set up another block transfer, perhaps using the same ICB.

Note that several transfer requests may be outstanding, since a queue of requests is kept by the system. The only restrictions are that each request have its own ICB. No ICB or data area should be modified until the operation associated with it is completed.

(The block transfer subroutine described above is the one which should be used in most cases since it provides the necessary seizure of control beforehand and operates within the system conventions explicitly. There are lower level routines used by the system which are described in the following sections and which may be of use to the user for special applications. These routines require Interface control to have been obtained beforehand, do no queueing, and no checking. A program may use these after it has directly seized interface control.)

SEIZURE OF CONTROL

As explained in Section 4.1.1, some programs may require a greater degree of control over the Interface than is permitted through simple block transfers and attention interrupts. A request for such control is initiated by calling the appropriate routine with one parameter, the address of a two-word ICB. When the operation is completed (as indicated by the "test" routine), the user has control of the Interface to operate as he sees fit within the following constraints.

1. No operation may be started which will result in an op-complete interrupt. (Block transfers may be initiated only by calling the lowest level routines, but the "enable interrupt" bit must not be set in the Status Register.)
2. Control of the Interface must not be relinquished directly by the program (i.e., the machine's resolution bit must not be reset).
3. The user must relinquish his control by calling a system subroutine to do so.

Operation of the interface is restricted when it is seized in this manner by a program. Queued block transfers will not take place, and the other machine will be completely locked out. For these reasons, the user should restrict his use of this function to short periods of time so that the normal operation of the Interface may take place unhindered.

ADDITIONAL FUNCTIONS

Sections 4.2 and 4.3 contain descriptions of subroutines which are not described here. These routines are for initialization and for access to the Interface from higher level languages.

	FUNCTION	SUBROUTINE NAMES		
		PDP-7	1800 ASSEMBLER	1800 FORTRAN
User entries	Issue an Attention Interrupt	GIVATN	ATTN7	ATTN7
	Receive an Attention Interrupt	TAKATN	ATNSU	ATNSU
	Reset the handling for one Attention	TAKATN	ATCLR	ATCLR
	Reset the handling for all Attentions	ATTNI	@CLR	---
	Block transfer request (queued)	BLKTRQ	DATA7	IDTA7
	Seize Interface control (queued)	SEZINT	SEIZE	SEIZE
	Release Interface control	RELINT	NEXTQ	NEXTQ
	Test an ICB for ready (for transfers and control)	TESTCB	TEST7	ITST7
Lower level routine available for special cases	Initialize all Interface handling	INTERI	INIT7	INIT7
	Enter an ICB on the queue	INTQUE	ENQUE	ENQUE
	Block transfer (immediate, not queued)	LODNGO	ILOAD,XSTRT	ILOAD,XSTRT

Table 4.1 Interface Function-Subroutine Names

4.2 1800 Interface Support under TSX

Programming support for the Interface is generally divisible into two areas: attention processing and data transfer. Although these functions certainly interact in everyday operation, they are handled largely by distinct sets of subroutines. Moreover, attention and op-complete interrupts appear to TSX as though they came from separate devices, so the distinction is functionall as well as conceptually convenient.

The subroutines described below are stored in the relocatable program area of the 1800 disk. ATTN7, @CLR, ENQUE, NEXTQ, and possibly others, will normally be included in the system skeleton. ATTN7, ATNSU, ATCLR, DATA7, TEST7, SEIZE, NEXTQ, ENQUE, ILOAD, and XSTRT are reentrant.

4.2.1 Attention Interrupts

ATTN7

Purpose: To generate an attention interrupt, with a given attention code, at the PDP-7.

Calling Sequence:

```
(assembler)     CALL    ATTN7
                DC      ='n'

(FORTRAN)       CALL    ATTN7(n)
```

where n = attention code number ($0 \leq n \leq 31$)

Description: ATTN7 will always wait until the PDP-7 has processed the previous attention before issuing the new one. The latter, however, will not necessarily have been processed by the PDP-7 when ATTN7 returns. ATTN7 can be called regardless of which machine has logical control of the Interface; but if the user wishes to transmit additional information through the control registers along with the attention, he must call SEIZE first to acquire the Interface. Code numbers 0 to 10 are reserved for system use.

ATNSU

Purpose: To establish a subroutine as an attention-handler.

Calling sequence:

```
(assembler)     CALL    ATNSU
                CALL    SUB
                DC      ='n'
```

where SUB is the name of the subroutine and n is the attention code number it handles.

Description: When an attention interrupt with code n is received, SUB is called at the interrupt level. The call to SUB invalidates it for further attention-handling; it can be restored by calling ATNSU again. If ATNSU is called from within SUB, the latter must return directly to TSX by a "B I 90" instruction, rather than "B I SUB". SUB cannot be written in FORTRAN.

ATCLR

Purpose: To undo the effect of ATNSU.

Calling sequence:

```
(assembler)     CALL    ATCLR
                DC      ='n'

(FORTRAN)       CALL    ATCLR(n)
```

where n = attention code number.

Description: ATCLR makes attention code n (perhaps temporarily) invalid. Such attentions will be ignored until a new handler is established via ATNSU.

4.2.2 Block Transfers

DATA7

Purpose: To initiate a block-transfer of data to or from the PDP-7.

Calling sequence:

```
(assembler)     CALL    DATA7
                DC      ICB
```

where ICB is the label of a 7-word block of core storage which will become the Interface Control Block for the transfer.

Description: DATA7 fills in words 1 and 2 of the ICB with system information and links it into the Interface Function Queue. Words 3 through 7 of the ICB must be set up by the user before calling DATA7, as follows:

word 3: address of an op-complete subroutine, to be executed at the interrupt level when the transfer is completed. If there is no such subroutine, this word must be zero.

word 4: starting address of the 1800 data area, i.e., contents of 1800 Address Register.

word 5: starting address of the PDP-7 data area i.e., contents of PDP-7 Address Register.

word 6: word count (in 16/16 mode) or block count (in 18/16 mode) for transfer; to be loaded into Unit Count Register.

word 7: transfer direction and mode, coded as follows:

0 = PDP-7 to 1800, 16/16 mode

1 = PDP-7 to 1800, 18/16 mode

2 = 1800 to PDP-7, 16/16 mode

3 = 1800 to PDP-7, 18/16 mode.

If this ICB is first on the queue, DATA7 will attempt to seize the Interface and initiate it. If other items are queued ahead of it, or if the Interface cannot be seized, DATA7 will return immediately, leaving the ICB on the queue to be initiated in due course. In any event, the user must wait for the operation to be completed and the ICB freed (as determined by calls to TEST7) before using or altering either the data area or the ICB.

An ICB may be re-used as soon as it becomes free. In particular, the op-complete subroutine (if any) may set up another transfer using the same ICB by calling DATA7 again.

4.2.3 Other Operations

SEIZE

Purpose: To "seize" the Interface; i.e., turn on the 1800's resolution bit in the Status Register.

Calling sequence:

```
(assembler)     CALL    SEIZE
                DC      ICB

(FORTRAN)       CALL    SEIZE(ICB(2))
```

where ICB is the label of a two-word block of core storage (assembler) or a two-word integer array (FORTRAN), which will become an Interface Control Block.

Description: SEIZE inserts information in the ICB and queues it on the Interface Function Queue. If this operation is first on the queue and the Interface can be seized, it will be done immediately; if not, it will be processed in due course. In any event, SEIZE returns immediately. The user must call TEST7 to determine when the operation is complete, i.e., the Interface seized.

Having seized the Interface, the user may do whatever operations he wishes on it, as long as they cause no interrupts to the 1800. Note that queue processing is suspended; calls to DATA7 or IDTA7 will queue operations, but not initiate them. XFER7 can be called to initiate operations directly. When he has finished, the user must call NEXTQ to resume queue processing and/or release the Interface.

NEXTQ

Purpose: To resume queue processing and/or release the Interface.

Calling sequence:

(assembler or FORTRAN)	CALL NEXTQ

Description: NEXTQ initiates the operation specified by the ICB currently at the top of the Interface Function Queue. If the IFQ is empty, it releases the Interface (i.e., turns off the 1800's resolution bit). NEXTQ is intended to be called by user programs which have SEIZE'd the Interface and performed some specialized operation(s) thereon; it is also called by the system op-complete interrupt processor.

INIT7

Purpose: To initialize the system Interface support subroutines

Calling sequence:

(assembler or FORTRAN)	CALL	INIT7

Description: INIT7 resets the Interface (via the Blast command), clears the attention-interrupt dispatch table to its default state, and resets the Interface Function Queue to empty. It is normally called when the 1800 is reloaded.

ITST7

Purpose: Same as TEST7

Calling Sequence: (FORTRAN INTEGER FUNCTION)

(FORTRAN) IF(ITST7(ICB(2))) n_1,n_2,n_1 after SEIZE

IF(ITST7(ICB(7))) n_1,n_2,n_1 after IDTA7

Description: Same as TEST7, but FORTRAN-callable. Value zero means operation busy; nonzero means operation complete.

TEST7

Purpose: To determine whether a queued Interface operation has been performed.

Calling sequence:

```
(assembler)        CALL      TEST7
                   DC        ICB
                   return 0 - busy
                   return 1 - not busy
```

where ICB is the label of the Interface Control Block for the operation in question.

Description: TEST7 determines whether the ICB whose address is given as a parameter is still on the Interface Function Queue. If so, the operation is still pending; if not, it is complete. If the operation is complete, TEST7 pseudo-skips on return.

IDTA7

Purpose: Same as DATA7, but FORTRAN-callable.

Calling sequence:

(FORTRAN) CALL IDTA7(ICB(7),AREA(LAST),IADR7,COUNT,MODE)

where ICB(7) is an element of a 7-word integer array.

AREA(LAST) is the last element of an integer array to (from) which data is to be written (read); because FORTRAN arrays are stored backwards in memory, the data transfer will go A(LAST), A(LAST-1), A(LAST-2),...,A(2),A(1).

IADR7 is an integer variable containing the address of the data area in the PDP-7.

COUNT is an integer variable containing the unit (word or block) count.

MODE is an integer variable specifying the transfer direction and mode (see "word 7" under DATA7).

Description: Same as DATA7.

Note: program containing the above call must be compiled using the *ONE WORD INTEGERS option.

The op-complete subroutine option is not available from FORTRAN.

@CLR

Purpose: To reset the attention-interrupt dispatch table.

Calling sequence:

(assembler) CALL @CLR

Description: Resets the attention-interrupt dispatch table to its default state, i.e., with only certain system codes enabled. Called by INIT7.

ILOAD

Purpose: To load the Interface control registers.

Calling sequence:

```
(assembler)     CALL        ILOAD
                DC          LIST

(FORTRAN)       CALL        ILOAD(LIST(4))
```

where LIST is the label of a 4-word block (or name of of an integer array) containing the data to be loaded.

Description: The word at LIST (or LIST(4)) is written into the 1800 Address Register; at LIST+1 (LIST(3)) into the PDP-7 Address Register; at LIST+2 (LIST(2)) into the Unit Count Register; and at LIST+3 (LIST(1)), into the Status Register.

XSTRT

Purpose: To start an Interface block-transfer.

Calling sequence:

(assembler or FORTRAN)	CALL	XSTRT

Description: Issues the Start command, with no checking; assumes the control registers have already been loaded.

ENQUE

Purpose: To append an Interface Control Block to the Interface Function Queue.

Calling sequence:

```
(assembler)        CALL    ENQUE
                   DC      ICB
```

where ICB is the label of the Interface Control Block

Description: ENQUE adds the ICB to the IFQ. If the IFQ was initially empty, ENQUE attempts to seize the Interface and, if successful, initiate the operation. The user will normally have occasion to call ENQUE only for nonstandard operations (i.e., not available through DATA7 or SEIZE).

4.3 PDP-7 Implementation under LOCOSS

The device support in the PDP-7 is implemented in LOCOSS (Logic of Computers Operating System for the Seven) as a set of subroutines permanently resident in the operating system. Knowledge of the LOCOSS conventions is assumed in the descriptions below. These subroutines are accessed by indirect reference through the system communication region. The LOCG assembler for the PDP-7 has appropriate definitions in its permanent symbol table for referencing the routines. For example, absolute location 135 (octal) has the address of the block transfer subroutine. The assembler's definition for BLKTRQ is "JMS* 135", as an operator. Thus, the user calls the block transfer subroutine by putting the symbol BLKTRQ in the operator field of an instruction.

4.3.1 Attention Interrupts

GIVATN

Purpose: To issue an attention interrupt to the 1800.

Calling sequence: (interrupts on or off)

(AC=attention number)

GIVATN

RETURN ------

Description: GIVATN will not disturb the interrupts-enabled status. The attention will not be issued until any previous attentions have been cleared by the 1800. Upon return from the GIVATN, the attention will have been issued, but not necessarily processed by the 1800. Interface control is not a prerequisite for issuing an attention, but if any register transfers are to occur as a result of the attention, the control should be obtained prior to calling GIVATN, through subroutine SEZINT.

Attention numbers 0-10 are reserved for system use; numbers 11-31 may be used by other programs.

No checks are made.

TAKATN

Purpose: To set up handling for attention interrupt reception.

Calling sequence: (interrupts off)

(AC=attention number)

TAKATN

XXX

RETURN ----------

Description: TAKATN sets up the attention interrupt handling for the attention number given in the low-order bits of the AC. This number must be less than 32; numbers 0-10 are reserved for system use. XXX is an instruction which is executed (i.e., XCT'ed) at interrupt time when the attention is received. If the instruction is a subroutine call, the subroutine thus called must return to its caller. If XXX is zero, the attention handling for that number reverts to normal system handling. The handling set up by calling this subroutine remains in effect until LOCOSS routine RESET is called or LOCOSS is restarted.

Normal system attention handling for interrupt numbers which have no service routines declared is to halt with the interrupt number in the AC.

ATTNI

Purpose: To reset handling for all attentions to the default case.

Calling sequence: ATTNI

RETURN --------

Description: Any handling for attention interrupts previously set up by users' programs is restored to the default.case, which is to halt with the interrupt number in the AC. After this subroutine has been called the only attentions recognized are the special system communications attentions.

This subroutine is called by INTERI.

4.3.2 Block Transfers

BLKTRQ

Purpose: To initiate a block transfer between the memories of the two machines (queued).

Calling sequence: (interrupts off)

```
        BLKTRQ
        DC  ICB
RETURN  ----------
```

Description: BLKTRQ must be called with interrupts off so that it may be called from an interrupt routine. Return is made with interrupts off, and is always immediate (i.e., control will not be passed to the task queue processor). Upon return, the transfer may or may not be completed so that subroutine TESTCB should be called to determine the status of the operation. Neither the ICB nor the transfer area should be changed or used until the operation is finished.

The ICB (Interface Control Block), whose address is the parameter to the call, is a seven-word block of core specifying all the information to control the transfer. The last five words are specified by the user, and the first two are used by the system subroutines. The contents of the block are:

1. For system use.
2. For system use.
3. The address of a subroutine to be called when the operation is completed (may be zero).
4. The IBM 1800 address.

5. The PDP-7 address.

6. The block count (Number of words if the transfer is in 16/16 mode, and the number of 8/9 blocks if the transfer is in 18/16 mode).

7. Direction and transfer mode, coded as follows:

   Direction: Bit 8=0, from PDP-7 to 1800
   =1, from 1800 to PDP-7

   Mode: Bit 9=0, 16/16 mode
   =1, 18/16 mode

   Thus, the following are the possible (octal) values this word may take:

   000000=PDP-7 to 1800, 16/16 mode
   001000=1800 to PDP-7, 16/16 mode
   000400=PDP-7 to 1800, 18/16 mode
   001400=1800 to PDP-7, 18/16 mode

The op-complete instruction will be executed when the transfer has been completed. If it is a subroutine call, that subroutine may perform only those actions permitted an interrupt subroutine, and must return to the caller. It may call BLKTRN itself to restart a block transfer operation.

Several transfer requests may be issued, as long as each has its own unique ICB. Requests are filled in the order received.

TESTCB

Purpose: To determine whether an operation has been completed.

Calling sequence:

```
          TESTCB
          DC   ICB
RETURN1   --------
          --------
RETURN2   --------
```

Description: Interrupts-enabled status is not affected by this subroutine. ICB is the address of an Interface Control Block, either transfer or seize. After the respective operation has been requested, the program determines the status of the operation by calling TESTCB with the control block as a parameter. If the operation is not yet finished, control is returned to RETURN1, which is the address of the call plus two. If the operation has been completed (i.e., the transfer is finished or control has been seized), control is returned to RETURN2, which is the address of the call plus four. (Thus, when the operation has finished, TESTCB performs a two word "pseudo-skip".) The user should not change the control block until TESTCB yields a success return.

SEZINT

Purpose: To obtain logical control of the Interface from the task-time level.

Calling sequence:
```
        SEZINT
        DC  ICB
RETURN  -------
```

Description: SEZINT may be called with interrupts on; it returns with interrupts on. ICB is a two-word Interface Control Block for use by the system. Upon return, the Interface may not yet be under control. The user determines when this occurs by calling TESTCB with ICB as a parameter.

When TESTCB indicates that the operation is finished (i.e., control has been obtained), the user may operate the Interface and its registers directly. The user may not perform any operation on the Interface which will lead to an interrupt occurring except by going through the system routines. No block transfers can take place through BLKTRQ while control is in the hands of the user via this subroutine (LODNGQ must be used instead). After the user no longer needs direct control of the Interface he must relinquish control by calling subroutine RELINT.

More than one seize request may be outstanding (e.g., from different tasks), but each request must have its own ICB.

RELINT

Purpose: To relinquish logical control of the Interface from the task-time level.

Calling sequence: RELINT

RETURN ------

Description: RELINT may be called with interrupts on or off; it returns with interrupts on. If the interface is under logical control of the program, the control is relinquished to the system. If the system already has control, nothing happens.

INTQUE

Purpose: To enter a control block onto the Interface Function Queue.

Calling sequence: (interrupts off)

(AC=address of an ICB)

INTQUE

RETURN ------

Description: On return, interrupts will still be off. The second word of the Interface Control Block should be an instruction to be executed when the control block is ready for operation. If the operation started by the control block word two results in an op-complete interrupt, the third word in the control block should be an instruction to be executed at op-complete time. If the operation does not produce an op-complete interrupt, the IFQ processor must be restarted by calling subroutine RELINT.

Upon return from INTQUE, the function may or may not be finished. Subroutine TESTCB must be called, with the control block as parameter, to determine the status of the operation.

INTERI

Purpose: To initialize Interface handling.

Calling sequence: (interrupts off)

INTERI

RETURN ------

Description: INTERI sets up the LOCOSS handling for Interface interrupts, sets all attention interrupt number handlers to the default case (by calling ATTNI), and clears the IFQ of all requests. INTERI is called by LOCOSS routine RESET and at LOCOSS initialization time, so that it is usually not called by user programs.

LODNGO

Purpose: To start a block transfer by loading the Interface registers and issuing the Interface Start command.

Calling sequence: (interrupts off)

(AC=address of a four-word block)

LODNGO

RETURN -------

Description: This routine operates and returns with interrupts-off because it is called at interrupt time by the IFQ processor. The four-word block addressed by the AC contains data to be loaded into the Interface registers in the order: 1800 Address, PDP-7 Address, Count, and Status. After the registers have been loaded, the Interface Start command is issued. The Status Register is loaded as given, so that it is the caller's responsibility to ensure the correct setting of the enable bits for the proper op-complete termination.

5. EVALUATION AND CONCLUSION

Our experience on the Interface has been limited primarily to development of diagnostic routines and normal device support routines. The hardware is just now being opened to general usage. The experience of "novice" users will doubtless affect our evaluation of our wisdom in developing this design - hopefully positively. Our personal experience has been very good, with the exception of some items discussed later in this section.

5.1 Design Variations

There are many ways in which the current design could be varied to suit specific requirements. We shall briefly sketch a few of these.

5.1.1 General Frame Size

In more general applications the effective word size for either or both computers could be made more variable. Consider two additional Frame Size Registers - one controlling the "word size" of each computer. A Frame Size Register would be used in two ways. First, its value would be used to reload the corresponding Shift Counter each time the latter reaches zero. This provides the necessary bit-counting. Second, its value would be decoded and used to select the point at which the data is inserted into the Data Register. This achieves the right-justification of the data at the destination computer. Figure 5.1 illustrates a possible method of constructing such a "variable length" Data Register for two hypothetical computers.

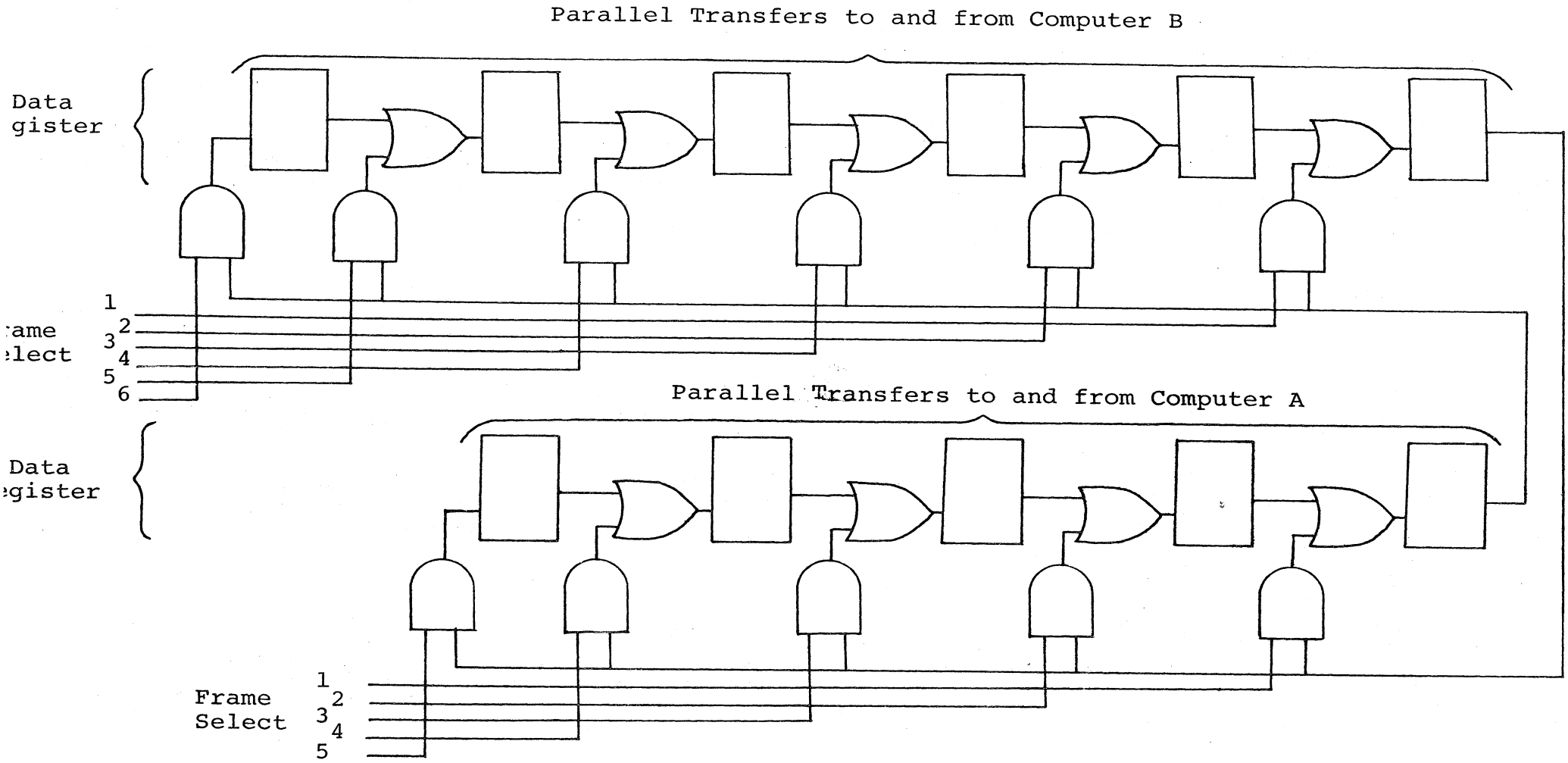


Figure 5.1 Method of Achieving "Variable Length" Data Register
(Timing and Parallel Gating Not Shown)

Of course, not all frame combinations need be provided. And the insertion point could be located to give other than right-justification.

5.1.2 Alternate Data Transfer Method

Observe that the variable-length circular Data Register is necessary to achieve the inverse of a given data transfer provided the conventional technique of starting a transfer at the beginning or lowest-numbered address of a block is used. An alternate organization could use a linear Data Register capable of shifting in both directions and Address Registers that can count both up and down. The inverse of a given data transfer then becomes a true (almost) time-reversed copy of the original transfer: the addresses start at the high end of a block and count down, and the shift direction is opposite the original. Of course, the Count Register still counts down.

5.1.3 Constraints on Control Functions

In some applications it may not be desirable to permit one machine to start a data transfer without the other machine verifying the action. It would be a simple matter to require a Start command from both machines to initiate transfers. Many other interlocks are possible.

5.1.4 Use with Nondirect-Memory-Access Machines

The architecture of this interface is oriented toward machines that provide the ability for an external device to access memory directly at an address of the device's choice. But for the IBM System 360 machines, for example, and for machines using virtual address schemes, such direct access is not possible.

The 360 channel does not permit external devices to provide a physical

storage address or to initiate a data transfer operation without cooperation from the CPU in the form of a channel program. In addition, the logic of virtual addressing does not permit user programs access to physical page information.

In applications where another machine is to be connected to a machine like a 360, the conceptual function of the interface can be maintained by means of a programming convention which would have the other machine load the interface registers with logical addresses and present an attention interrupt to the 360, whereupon the latter could construct an appropriate channel program and initiate the operation.

5.2 "If we had it to do over..."

This section discusses ways in which the interface could have been improved and/or made simpler based on hindsight.

5.2.1 Uniformity of Structure

The original goal of making the interface completely programmable should have been followed more closely; hardware debugging would have been easier if all bits of the status register could have been set or cleared by program. An additional "diagnostic" register, consisting of all control flip-flops not represented in the status register, would have aided hardware debugging also. The attention codes should be half-duplex paths like the other registers. Errors should not be reported as attentions, but rather as ending status available with the operation-complete interrupt.

5.2.2 Simultaneity of Control

Our experience suggests that some simplification could be achieved

if the requirements for simultaneous access to the interface by the two computers were relaxed. It is probably adequate to expect the interface to be seized before any control function is performed (except seize and perhaps attention). This would ease the timing constraints on some functions and should permit more sharing of interface logic between the respective computers.

5.3 Summary

An interface has been provided between an IBM 1800 and a DEC PDP-7 that permits flexible and interactive programming. It provides block transfers of data at high speeds in either of two modes: on a word-for-word basis, or using a pack-unpack arrangement to compensate for the difference in memory word lengths. Control registers are accessible to both computers, allowing the interface to be initialized by either computer acting alone or by both cooperatively. A conflict resolution circuit permits either computer to seize logical control of the interface asynchronously and unambiguously.

BIBLIOGRAPHY

Brender, Ronald F. "A Programming System for the Simulation of Cellular Spaces," Ph.D. Dissertation, The University of Michigan, 1969.

Brender, Ronald F. and Foy, J.L., Jr. "Flexible High-Speed Interface Between IBM 1800 and DEC PDP-7 Computers," Technical Report 12, Concomp Project, The University of Michigan, Ann Arbor, October, 1968.

Digital Equipment Corp. *Interface and Installation Manual*, DEC Publication F-78A.

___, *PDP-7 Users Handbook*, DEC Publication F-75.

Frantz, D.R.; Brender, R.F.; and Foy, J.L.,Jr. "LOCOSS: A Multiprogramming Monitor for the PDP-7," Technical Report 10, Concomp Project, The University of Michigan, Ann ARbor, November 1968.

International Business Machines Corp. *IBM 1800 Functional Characteristics*, IBM Publication A26-5918.

___, *IBM 1801 and 1802 Processor-Controllers Original Equipment Manufacturers' Information*, IBM publication A26-3591.

APPENDIX A: INTERFACE CONTROL PANEL

The Interface Control Panel consists of indicators, toggle switches, rotary switches, and momentary-contact switches. Each of these is labeled with a letter in the schematic diagram (Figure A.1) and described below.

Indicator Displays (a - i, p)

a. 34-bit Shift Register

b. 16-bit 1800 Address Register

c. 16-bit PDP-7 Address Register

d. 16-bit Unit Count Register

e. 5-bit B Shift Counter

f. 5-bit A Shift Counter

g. 16-bit Buffer Registers, selected by switch j.

- j1: 1800-Interface Data Buffer
- j2: PDP7-Interface Data Buffer
- j3: Interface-PDP7 Data Buffer
- j4: Interface-1800 Data Buffer
- j5: 1800 Command Buffer

h. 8-bit Control Flip-flops, selected by switch k.

k1:
1. PDP-7 Instruction (PDP7-I) FF.
2. not used
3. PDP7 T&S PENDING FF
4-8. PDP7 Period Counter 0-4

k2:
1. 1800 Instruction (1800-I) FF
2-4. not used
5. 1800 T&S PENDING FF
6-8. 1800 Period Counter 0-2

k3:
1. PDP7 ILLEGAL WRITE FF
2. 1800 ILLEGAL WRITE FF
3. PARITY ERROR FF
4. STOR PROT VIOLATION FF
5. PDP7 BRK REQ FF
6. 1800 CS REQ FF
7. HALT PENDING FF
8. GO FF

i. 11-bit Status Register, 5-bit 1800 Attention Code Register

i'. 5-bit PDP-7 Attention Code Register

p. Power-on pilot lamp

Rotary Switches (j-k)

j. 5-position Buffer Register display selector

k. 3-position Control Flip-flops display selector

Toggle Switches (l-o)

l. Enable/Disable PDP-7

m. Enable/Disable 1800

n. Clock switch - Run/Single-step

o. Power switch - On/Off

Momentary-Contact Switches (q-r)

q. Clock pulser - Start/Step

r. Interface reset

Notes:

Switches l and m, in the Disable position, prevent the corresponding computer from accessing the Interface and prevent the Interface from interrupting the computer. Block-transfers (if initiated by the other computer) are not inhibited.

With switch n in the Single-step position, each depression of the clock pulser, switch q, causes the Interface basic clock to advance one step (i.e., causes one transition of the PHASE flip-flop).

Settings for normal operationsare:

l,m - Enable

n - Run

o - On

Power-on procedure:

1. Turn POWER (switch o) on.
2. Press RESET (switch r)
3. Press START (switch q)

The PDP-7 and 1800 should both be stopped while Interface power is being turned on.

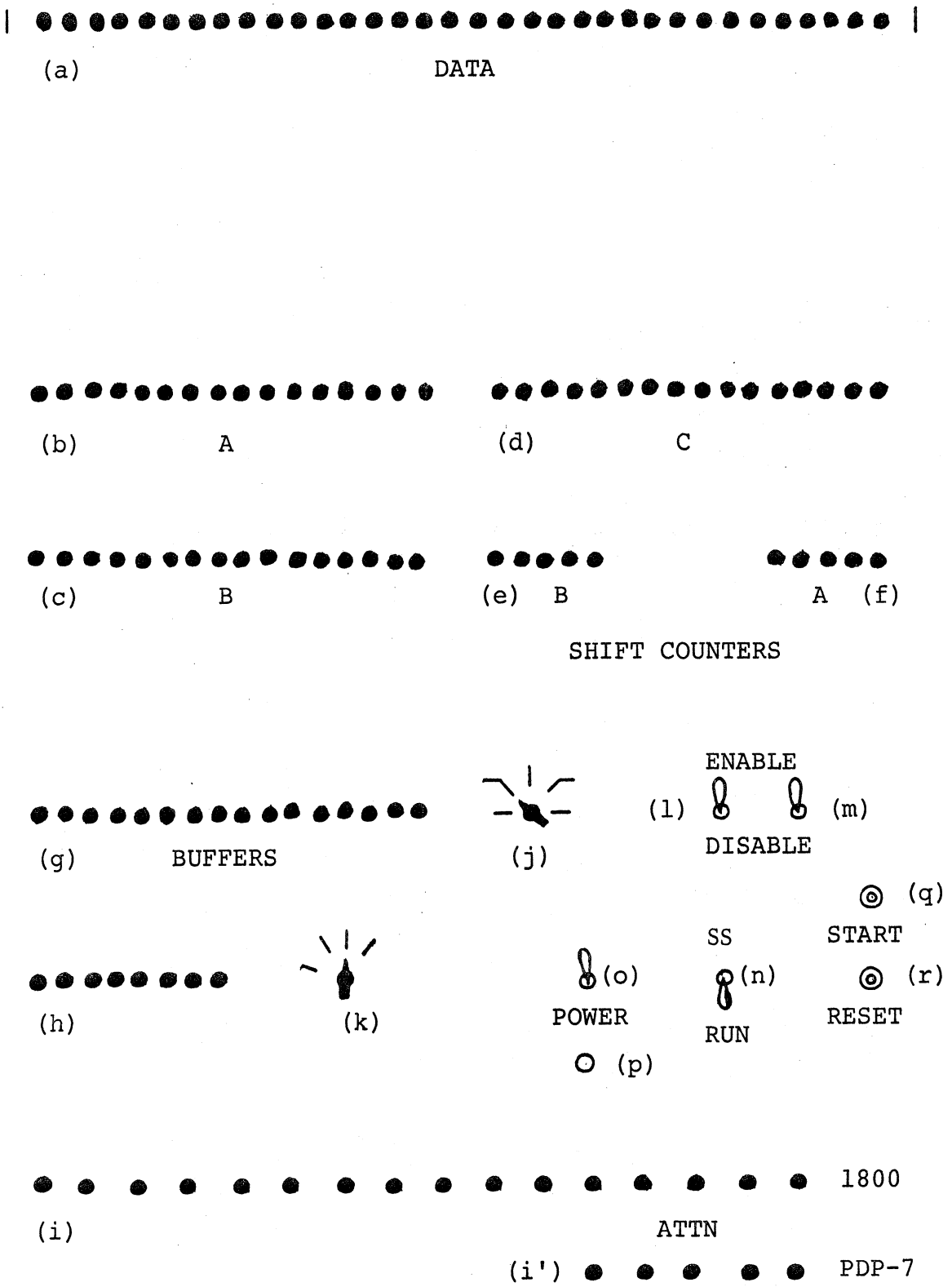


Figure A.1 Interface Control Panel

Appendix B: Command Summaries

1. 1800 Commands

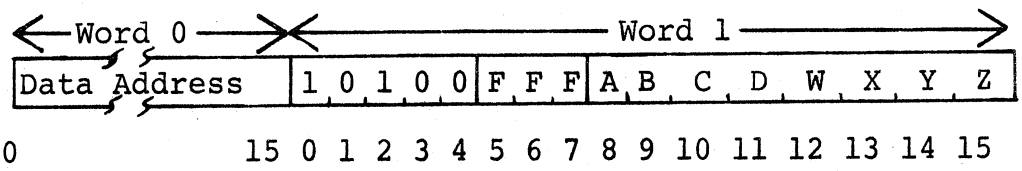
The 1800 issues commands to the Interface by executing the Execute I/O (XIO) instruction, which refers to a two-word I/O Control Command (IOCC). The left-hand word of the IOCC contains the address of a memory location to be used in a Read or Write function; on all other operations concerning the Interface, the left-hand word is ignored. The right-hand word is broken into three sections, as follows:

Area (bits 0-4) - contains a unique code for each device on the system; Interface code is 10100.

Function (bits 5-7) - specifies type of operation; functions relevant to the Interface are Read, Sense, Write, Control, Sense Interrupt Level.

Modifier (bits 8-15)- further specifies certain functions, e.g., which register is desired during a Sense.

IOCC Format for Interface Commands



FFF - function code

001 = WRITE

010 = READ

011 = SENSE INTERRUPT LEVEL

100 = CONTROL

111 = SENSE

ABCD - register-select for READ, WRITE, SENSE

1000 - 1800 Address Register

0100 = PDP-7 Address Register

0010 = Unit Count Register

0001 - Status REgister

Note - more than one register may be selected at the same time. This is reasonable only for WRITE; for READ or SENSE, the selected registers would be OR'ed together.

WXYZ - function sub-select

a) WRITE (FFF = 001)

1000 = Clear Attention Code (to PDP-7)

0100 = Bit Reset (Status Register only)

0010 = Bit Set (selected register(s))

0001 = Clear (selected register(s))

Any or all of the above bits may be on simultaneously.

b) SENSE (FFF = 111)

0100 = Perform Test & Set

0010 = Set "Pending" flip-flop

(Regular Test & Set is 0100; Test & Set (Pending) is 0110.)

c) CONTROL (FFF = 100)

0100 = Start block-transfers

0010 = Halt block-transfer

0001 = Blast - Interface reset

Note: the function WRITE (FFF = 001) is not the same as the command "Write". Several different commands have a function of WRITE (FFF = 001) but different modifiers (ABCDWXYZ): among these are Clear Attention Code, Reset Status, Set Status, Write (register), and Clear (register). The command Write has WXYZ = 0011, i.e., both Clear and Set are selected.

2. PDP-7 Commands

The PDP-7 issues commands to the Interface by executing IOT instructions. An IOT instruction is broken into five fields as follows:

Operation code (bits 0-3)

- 1110 for IOT

Device selection (bits 6-11)

- contains a unique code for each device on the system; Interface code is 5X (octal), where X is any octal digit. (X is used for further specification of Interface commands.)

Sub-device selection (bits 4-5, 12-13)

- available for further specification of device functions

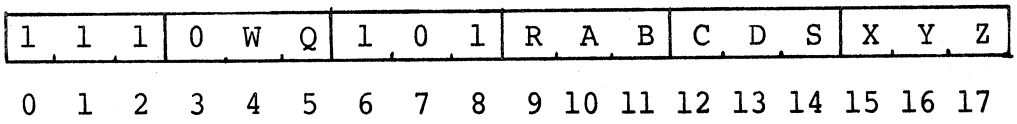
Clear accumulator (bit 14)

- if set, causes the accumulator to be cleared (preparatory to receipt of data from the device)

IOP selection (bits 15-17)

- each bit, if set, causes a pulse to be emitted on a corresponding control line to the device; these bits are used by the Interface, but the corresponding pulses are ignored.

IOT Format for Interface Commands



Q - command class

1 = control

0 = read/write

R - command subclass

1 = write

0 = read

S - clear accumulator

1 = clear accumulator; normally set with Read

0 = do not clear accumulator

ABCD - register selection for Read, Write

1000 = 1800 Address Register

0100 = PDP-7 Address Register

0010 = Unit Count Register

0001 = Status Register

Note - more than one register may be selected at the same time. This is reasonable only for Write; for Read, the selected registers would be ORed together.

ABCD - function sub-selection for Control

1000 = Skip on Op-Complete

0100 = Skip on Attention

0010 = Set "pending" flip-flop

0001 = Perform Test & Set

(Regular Test & Set is 0001; Test & Set (Pending) is 0011.)

W - function sub-selection for Write

1 = clear attention code to 1800

0 = do not clear attention code

XYZ - function sub-selection

a) Write (QR = 01)

100 = Bit Reset (Status Register only)

010 = Bit set (selected register(s))

001 - Clear (selected register (s))

(Any of WXYZ may be on at once.)

b) Control (QR = 10)

100 = Start block-transfer

010 = Blast-Interface reset

001 = Halt block-transfer

1800 Command List (right-hand words of IOCC's)

For brevity, the following abbreviations will be used:

A - 1800 Address Register

B - PDP-7 Address Register

C - Unit Count Register

D - Status Register

Command	IOCC (hex)	
Read A	A280	
Read B	A240	
Read C	A220	
Read D	A210	
Sense A	A780	
Sense B	A740	
Sense C	A720	
Sense D	A710	
Write A	A183	(= Clear & Set)
Write B	A143	" "
Write C	A123	" "
Write D	A113	" "
Clear A	A181	
Clear B	A141	
Clear C	A121	
Clear D	A111	
Set D	A112	
Reset D	A114	

Write Attention Code	A11A	(= Clear Attention Code + Set D)
Clear Attention Code	A118	
Test & Set	A714	
Test & Set (Pending)	A716	
Blast	A401	
Start	A404	
Halt	A402	
Sense Interrupt Level	0300	

PDP-7 Command List (IOT's)

For brevity, the following abbreviations will be used:

A - 1800 Address Register

B - PDP-7 Address Register

C - Unit Count Register

D - Status Register

Command	IOT (octal)	
Read A	705210	
Read B	705110	
Read C	705050	
Read D	705030	
Write A	705603	(= Clear & Set)
Write B	705503	" "
Write C	705443	" "
Write D	705423	" "
Clear A	705601	
Clear B	705501	
Clear C	705441	
Clear D	705421	
Set D	705422	
Reset D	705424	
Write Attention Code	725422	(= Clear Attention Code + Set D)
Clear Attention Code	725420	
Test & Set	715020	
Test & Set (Pending)	715060	
Blast	715002	
Start	715004	
Halt	715001	

APPENDIX C: DIAGNOSTIC TEST PROGRAMS

C.1 1800 Test Programs

C.1.1 Introduction

The 1800 Interface Test Program consists of three major sections in addition to initialization and error-reporting. These sections correspond to their counterparts on the PDP-7:

A. Control functions tests

B. Cooperative tests

C. Block-transfer tests

The program is named ITEST, and normally resides in the relocatable program area of the 1800 disk. The usual TSX control cards required for execution of the program are as follows:

```
// JOB
// XEQ ITEST
*CCEND
```

The desired tests are selected by means of the SENSE/PROGRAM and DATA ENTRY switches on the 1800 console. When the program is loaded it types "READY", then pauses to allow the operator time to set the switches. Execution resumes when the operator presses the console START button.

It is assumed that the operator has an assembly listing of ITEST at hand while running these tests. Switch options are specified by comments on the first pages of the listing, and interpretation of error messages

requires reference to the program text in the body of the listing.

C.1.2 Error Reporting

Error messages in sections A and B have the format

XXXX YYYY ZZZZ

where XXXX is the address from which the error subroutine was called (relative to the program origin).

YYYY is the contents of the accumulator when the error subroutine was called; this is usually the value in error.

ZZZZ is the value of the "current comparand"; this is usually the value that should have been in the accumulator.

Since the error routine returns normally to the point from which it was called after printing the error message (and possibly pausing, depending on switch settings), the test that was in progress will be continued. In certain cases this will lead to further error messages which are actually spurious, resulting from the initial error. Standard procedure when this happens is to set the switches to suppress error-printing until things re-synchronize, possibly stopping and re-selecting the test.

The section C tests are all of one type, in which a data table is generated, written into the PDP-7, read back into another table, and compared with the original

table. Therefore this section has really only one error message, and its format is

AAAA BBBB CCCC DDDD EEEE FF

where

AAAA is what should have been received;

BBBB is what was received;

CCCC is the offset, from the beginning of the data table, of the erroneous word;

DDDD is the exclusive-or of AAAA and BBBB (to aid in recognition of failing bits);

EEEE is the total number of erroneous words in the table;

FF is "16" or "18", depending on whether the mode of transmission was 16/16 or 18/16, respectively.

Note that if more than a single error occurs, only the first is logged out in detail; the others are simply counted.

C.1.3 Section A Tests - Registers and Control Functions (Sense switch 0)

1. Blast Test (Data Switch 4)

An all-one's constant ($FFFF_{16}$) is written into all four registers. The Status Register is then read and compared with a mask to see that only those bits which are supposed to be program-settable were in fact set.

A Blast command is issued, and all four registers read to be sure they were set to zero by the Blast.

2. Attention-Interrupt Test (Data Switch 5)

This test sets the Attention 1800 bit (bit 1 of the Status Register), then verifies that an interrupt is received on level one (an interrupt on any other level will be trapped; see Section C.1.6). A Sense Interrupt Level command verifies that the proper bit (4000_{16}) of the Interrupt Level Status Word is generated. The Status Register is tested to be sure the proper bit is still on; then it is reset, and the reset verified. Then the 1800's interrupt priority circuits are reset; a spurious interrupt from the Interface would cause a trap at this point.

3. "Test & Set" Test (Data Switch 6)

This test checks the immediate form of the Test & Set command. (Note: a complete test of this command requires cooperation from the PDP-7; see Section C.3.) Test & Set command is issued and verified, then is issued redundantly and verified again. The 1800 resolution bit is reset via the Reset Status command, and verified.

4. "Test & Set (Pending)" Test (Data Switch 7)

This test checks the pending form of the Test & Set command. (As above, a complete test requires cooperation from the PDP-7; see Section C.3). A Test & Set (Pending) command is issued and reception of the resulting

interrupt is verified. A Sense Interrupt Level command verifies that the proper bit (2000_{16}) of the Interrupt Level Status Word is generated. The Status Register is tested to be sure the Op-Complete (bit 0), Enable 1800 (bit 3), and 1800 Resolution (bit 5) bits are all on. A redundant Test & Set (Pending) is issued, and a check made to see that the above three bits are still on. Next the test resets the Enable 1800 bit, verifies this, and resets the 1800's interrupt priority circuits; absence of a spurious-interrupt trap (see Section C.1.6) at this point verifies the Enable function. Finally the Op-Complete and 1800 Resolution bits are reset and verified.

5. Registers Tests (Data Switch 11)

Each register is written with a test bit pattern, then read back and verified. The test patterns are different for each register, and are kept in variables named CPU@A, CPU@B, CPU@C, and CPU@D. On each cycle the register is read to be sure the previous pattern is still present; then the pattern is incremented by a quantity constant but different for each register. The new pattern is written into the register, then read back using both Read and Sense commands. Any discrepancies are reported. The above procedure is carried out on each of the other registers in turn, then the whole process is repeated.

The Status-Register test masks the test pattern to include only the controllable bits of that register; additionally, it tests the bit-set and bit-reset functions.

Any register(s) may be omitted from the test loop by switch selection.

C.1.4 Section B - Cooperative Tests (Sense switch 1)

This section consists of three tests which check the Attention Code and Resolution circuits more thoroughly than is possible from either machine separately. The tests may be run one at a time under manual selection (Data switches 4, 5, and 6 for the Attention Code, Test & Set, and Test & Set (Pending) tests, respectively), or they may be run under direction of the PDP-7 test program in auto-sequence mode (Data switch 15). See Section C.3 for further details.

C.1.5 Section C - Block Transfer Tests (Sense switch 2)

Tests in this settion are all of the same basic type: a table of data is generated, written into PDP-7 memory, read back into a different region of 1800 memory, and compared with the original table. The write-read-compare cycle is performed a fixed number of times, then the table is regenerated and the whole process repeated. The operator, by switch setting, selects the type of data table to be generated, and also certain options.

The table types and options are as follows.

1. Fixed tables

The table size and starting addresses in each machine are fixed (assembly parameters of ITEST), and the table is filled with the constant $AAAA_{16}$ (Data switches 5 and 6 off), $FFFF_{16}$ (Data switch 5 on, 6 off), or 0 (Data switch 5 off, 6 on). The alternate ones and zeros ($AAAA_{16}$) are considered to to be a worst-case test of the Interface Data Register's serial reliability, while the all-ones ($FFFF_{16}$) and all-zeros (0) are a test for bits picked up or dropped during parallel tranfers. Each table is 2048_{10} words in length.

2. Random Tables

Within constraints established by assembly parameters, the table sizes and starting addresses in both machines are varied randomly. The initial data table is also generated randomly (the distribution is uniform over 0 to 2^{16}-1), and is regenerated from a new random sequence after every 16th write-read-compare cycle. The table size and starting addresses are varied every time the table is regenerated. This random test (Data switch 7 on) is considered the most stringent overall test of the Interface block-transfer functions.

3. Options

Data switch 4 controls the mode: if off, the above

tests are performed in the 16/16 mode; if on, in the 18/16 mode.

Data switch 8 selects testing of the Halt command; if on, each block transfer is interrupted by a Halt issued at a randomly-chosen point in the transfer. The Interface registers are logged out, a Blast is issued, the registers are reloaded, and a Start is given. If everything works, this interruption is transparent to the "higher level" portions of the test program; if not, block transfer "data errors" will appear.

The "hardware fault" circuits test is selected by Data switch 9. Hardware faults are parity and storage-protect errors; attempted modification of the Address or Count Registers during a block transfer is also included in this category. This test attempts to alter illegally the Address and Count Registers, verifies that the "hardware fault" attentions are generated, then clears them. The data transfer should remain unaffected; if not, errors will appear during the data table comparisons. In addition, a location in the receiving data table is store-protected; detection of this condition by the Interface is verified, after which the store-protection is removed and the aborted transfer restarted from the beginning. Parity errors cannot be forced by program control, and hence cannot be checked by ITEST.

N.B.: This option should not be selected jointly with the random test, since it makes assumptions about the length of the data tables which are valid only during the fixed-tables tests. This test should also not be jointly selected with the Halt test.

In selecting options for the block transfer tests, the operator should be aware that the test program examines the switches only at table-regeneration time--once every 16 write-read-compare cycles. Thus the effect of a new switch setting may be delayed nearly two seconds.

C.1.6 Initialization - All Sections

The system mask register is set to $FFFF_{16}$ to force the TSX I/O subroutines to run on indicators instead of interrupts. The timer(C) is stopped to prevent timer interrupts from occurring. The interrupt vector in low core (locations $000A_{16}$ through 0015_{16}) is saved and replaced with pointers to a set of spurious-interrupt handlers; thus any unexpected interrupts will cause error message(s). After typing "READY", the program waits for the last typewriter interrupt to occur and clears it; if this were not done, a "spurious interrupt" would appear on level zero.

Upon exit (Data switch 0 on), the program restores

the system mask register and interrupt vector. The clock, although probably in error, is restarted. Return to TSX is via CALL EXIT.

Summary of Switch Settings for 1800 Interface Test Program

A.	Control Functions Tests	Sense switch	0
	Blast test	Data switch	4
	Attention-interrupt test	" "	5
	Test & Set test	" "	6
	Test & Set (Pending) test	" "	7
	Registers test	" "	11
	Omit 1800 Address Register	" "	12
	Omit PDP7 Address Register	" "	13
	Omit Unit Count "	" "	14
	Omit Status "	" "	15
B.	Cooperative Tests	Sense switch	1
	Attention-code test	Data switch	4
	Test & Set test	" "	5
	Test & Set (Pending) test	" "	6
	Auto-sequence mode	" "	15
C.	Block-Transfer tests	Sense switch	2
	18/16 mode	Data switch	4
	All 1's data table	" "	5
	All 0's data table	" "	6
	Random table	" "	7

Include "Halt" test	Data switch	8
Include "hardware-fault" test	" "	9
Stop after "Halt"	" "	10
For all tests:		
Exit test program	Data switch	0
Stop on error		1
Suppress error messages		2
Type "X" (off), nothing (on)		3
(Switch 3 is effective only if 2 is on)		
Force end of current timeout	Program switch	4

C.2 PDP-7 Test Program

The PDP-7/Interface Diagnostic program is divided into five sections:

1) Symbol definitions, service routines, and initial command sequences,
2) General registers tests (Section A),
3) Cooperative (Section B) tests,
4) Data-Break (Section C) tests,
5) Error reporting and miscellaneous routines.

In general, the register tests involve those functions that can be tested from the PDP-7 independently of the 1800. The cooperative tests involve those functions which require the active cooperation of both 1800 and PDP-7 to test. The block data tests involve the cycle-steal and halt tests.

Sections A, B, and C are completely independent of each other so that a special purpose test may be made be assembling the first and last parts together with the desired test section.

This diagnostic uses the routines PROCT, PRDEC, and ASCOUT of LOCOSS for I/O conversion. Special device routines have been substituted for KBDA and TPRA which run with interrupts off and use only the device flags. A completely self-contained diagnostic could be provided by including copies of the PRDEC, PROCT, and ASCOUT routines with the service routines of the first section in a manner similar to that used for KBDA and TPRA.

Errors are reported as three octal numbers, as follows: the first is the address of the point where the error was detected. This address

may be used to look into the program listing to identify the particular error and the context in which it occurred. The second number is the contents of the accumulator at the time of the error call. This is almost always the value that is in error. The third number (usually) is the expected value that should have been in the accumulator. From examination of the program listing the actual error will hopefully be determined.

After the test program is loaded and started, it asks the operator which of the three groups, called A, B, and C respectively, is to be used. Then the operator may obtain a summary of the switch assignments for selecting the individual tests to be used.

To facilitate locating the appropriate section of program from the address of an error report, each call to the error-reporting routine has been given a program label. The first two letters of the label are "ER". The next letter is "A", "B", or "C" to identify the group of tests. The next letter is a numeral giving the number of the test within each group. Next follows a period ".". The sixth letter is a numeral giving the position of the error call within the test. For example, if the typed-out location of an error corresponded to label "ERA1.1", then one would know that the error was the first error check in the first test in the register group of tests. This label may quickly be found in the listings to find the probable cause of the error.

The following subsections will discuss the various diagnostics tests in detail. This summary of intended function may be useful in understanding the program listings and interpreting the significance

of error type-outs. Where program labels are used in the text they are enclosed in parentheses, e.g., (ERA1.1).

C.2.1 Section A - General Register Exercises

Section A consists of five tests. These are selected by switches 6 through 9 and 13.

Switch 6 - Blast Test - (A1). All ones are written into all four registers of the Interface. Not all bits of the STATUS register are writable. The first test reads the STATUS and checks that all and only settable bits are ones. Next a BLAST command is given to clear all bits of the interface. Each register is read and compared in turn with zero to verify.

Switch 13 - Registers Test - (A2). All four registers are written with a sequence of bit patterns to test the ability to write and re-read the contents of any register independent of any other register. The four memory locations (OLDXA), (OLDXB), (OLDXC), and (OLDXD) contain the expected contents of the respective registers. Working with each register in turn, the register is read and compared to verify that the old value is still present. Next a new value is written and immediately read back to verify that it was properly written.

The bit pattern to be used is systematically varied across the four registers. The base pattern is used with register A. This is then complemented and used with B. Adding one gives the twos complement to be used with C. This is further complemented, and masked to settable bits only, for use with D. Next the base pattern is incremented and the cycle repeated.

Any or all of the registers may be excluded from this test using switches 14 through 17 (corresponding to A through D). Switch 14 up, for example, leaves the A register untouched by this test. Using a similar selection capability on the 1800, it is possible for one machine to exercise some of the registers and the other machine to exercise the remaining ones. As long as the same register is not simultaneously selected by both machines, the tests should proceed without interfering with each other.

Note: Testing register D with this test will cause interrupts at the 1800.

Switch 7 - Attention Flag Tests - (A3). The PDP-7 sets its own attention flag in the status register and skips on it. A check is also made that the "skip on operation complete" does not also occur. Next the flag is cleared and failure to skip with it cleared is verified.

Interrupts are set up and the attention flag turned on again. This should cause a trap through location one, after which the skip is checked once more. After clearing the flag, interrupts are turned on yet once more to verify that no further traps occur.

Switch 8 - Test & Set Function - (A4). The Test & Set function is executed. Since the 1800 has not seized the interface, the instruction should skip. Status is read to verify the appropriate seize bit is set. A second seize instruction while the bit is set should not skip. The bit is then cleared and this verified.

Switch 9 - Test & Set Pending Form with Interrupt - (A5). Interrupt handling is established, then the pending Test & Set issued. This should cause an immediate interrupt. This form also sets the operation-complete

and enable interrupt to PDP-7 bits in the Status register. This is checked. The operation complete skip is verified. Then the interrupt enable is cleared and the skip and lack of interrupt verified. The rest of the status is cleared one bit at a time and the expected states verified at each step.

C.2.2 Section C - Block Transfer Tests

Section C consists of tests of the data-break or block transfer capabilities of the Interface. These require only that the 1800 be cycling (or in WAIT state). Test C1 may be run simultaneously and independently with data-break tests running on the 1800 side. This provides a thorough exercise of nearly all interface functions under heavy load conditions.

Switch 6 - Basic Block Test - (C1). This test uses only the 16/16 mode transfer of a small block of core. It is also called as a subroutine by other tests. For reasons too muddy to explain, it is entered at label TSTC.9 when called directly and at label TSTC.1 when called as a subroutine.

A block A7 of SIZE words is written into 1800 core at location A18A. This same block is then read back into a second PDP-7 area (A7A) and compared against the original data for errors. Next a known pattern of words with exactly one one in each word is written into the same 1800 area as before. This alternation of data patterns insures that a failure to transfer will not appear to succeed because of data left from the previous try.

Each of the parameters SIZE (initially 20_8) and A18A (initially

37760_8) may be modified. All uses of these parameters use dynamically computed values so that they may even be changed at run time. The restrictions are these: SIZE may not exceed 20_8 and must be at least 2. Test C2 will require that SIZE be at least 7. A18A must not be such that the block written into 1800 core will store into the area where the 1800 is executing. In addition, contents of the test pattern in A7 may be changed. The initial values alternate between mostly ones and mostly zeros in successive words. Other sequences may be desirable. Changes to SIZE or A7A are conveniently performed by DDT or LOCOSS. Locations must be determined from the symbol table.

Switch 7 - Edge Tests - (C2). This verifies that the block transfer is starting and stopping at the desired locations.

First the word after the end of the second PDP-7 data area A7A is marked with a known value. Test C1 is called, and on conclusion the stored value checked to verify that no more than the intended number of words were read back into A7A.

Next a two-word block is written into the midst of the data left in the 1800 by the call on test C1. This same block is read back and the words immediately before and after the return block are checked for no change. Then four words are read back from the 1800 including the original two in the middle. This permits the PDP-7 to verify that the original transfer involved only the desired number of words.

The call on test C1 uses the alternate entry TSTC.1 which results in no error type-outs from test C1 itself. Only errors detected in test C2 will be typed by this test.

Note that the requirement that SIZE be at least 7 is checked by the test on entry.

Switch 8 - General Data-Break Tests - (C3). This test is intended to exercise the Interface under conditions somewhat like those that might be found in actual applications. The test consists of two sections: a calling program that sets up the parameters of a given test, and a general subroutine for carrying out a test.

A test is specified by giving four parameters: the frame size conditions and number of blocks for the transfer over, and those for the transfer back. The calling program accesses a series of four-word sets which specify these parameters and uses them to call the general tester. The only constraint on these control words is that the number of words written into the 1800 must not be less than the number read for correct results. This is not checked by the called routine.

The general subroutine (TSTGEN) uses the four parameters to calculate the remaining parameters of the test. Then a block of randomly generated numbers of the desired length is computed. This block is transmitted to the 1800 using the given frame conditions. Next the PDP-7 computes an image of the core of the 1800 as it should look after this transfer, by doing a full simulation of the interface behavior. The data in the 1800 is then read back in 16/16 frame mode and compared to the anticipated values. Next the data is read back in the original frame conditions which should return it to its original word and bit pattern. This is also compared with the original pattern to verify correct operation.

Switch 9 - Permute 1800 Base Address - (C4). This is not really a test but rather a routine to vary the 1800 address parameter A18A used to cycle through the 1800 address space. The assumption is made that locations 0 and 1 of 1800 memory are to be avoided. These locations would normally contain a BSC L /0000 or possibly simply a WAIT in location 0.

Switch 10 - Data Table Shift Test - (C5). This curious diagnostic grew out of a real hardware debugging problem. On certain occasions it appeared that a block of data would be transferred correctly in all respects except that the entire block would be moved up or down a small number of words in memory. This test seeks to identify such a fault and print out the number of words by which the block shifted (rather than give an entire table of error messages such as test C1 would give.)

Test C1 is called at its alternate entry TSTC.1. Then the middle word of the transferred block (computed dynamically from SIZE) is sought in the returned block A7A. If found in the correct location the test is over. If found elsewhere, a comment is given telling the offset relative to where it should have been. The message "SHIFT BY 2" might mean, for example, the data was read into the 1800 two locations too high or back into the PDP-7 two locations too high. (Or even one too high in each direction.) The cases are best distinguished by halting after this type-out and examining 1800 core. If the desired data item cannot even be found in the expected area the message "SHIFT OUT OF BOUNDS" is given.

Switch 11 - Simple Formatting. Errors in Section C will often result in more than one line of type-out. When errors from successive cycles through the selection loop are typed one after the other, it may be difficult to separate whether a given error type-out occurred on the same pass or on a later pass of the selection loop. To resolve this problem, switch 11 may be selected. This causes an extra carriage return to be typed after each time through the selection loop which resulted in an error. Thus errors on the same pass are grouped together and errors on successive passes are separated. This option is recommended as the normal case with Section C tests.

Block Transfer Conventions - (CX). A single block transfer subroutine is used to initiate and complete all data break transfers. The Interface is seized using Test & Set (Pending). Then the appropriate initial register values are obtained from the calling sequence and loaded into the Interface registers. Next there are two possible courses of action: normal case and Halt variation. In the normal case, the Interface is given a Start command to initiate the data transfer. The transfer is considered done when the Count Register goes to zero, after which the SKPOPC (skip on operation complete) is tested. Next the final state of the registers is saved for possible use by the calling routines and the Status Register is cleared (in particular, bit 6 is cleared.)

If the Halt test is specified (indirectly by switch 12), a random delay proportional to the length of the data transfer is set up to begin with the issuance of the Start command. At the end of the delay, an Interface Halt is given. After waiting for the physical halt, the registers are saved. If switch 13 is up the program halts to permit

visual examination of the Interface registers. Next a Blast completely resets the Interface, the four registers are reloaded and the Interface restarted. Control passes to the normal termination procedure.

Because a Blast is given and the Interface is not re-seized when testing the Halt, this variation cannot be used with the 1800 simultaneously running data break tests.

Note that the Halt test is selected independently of the other tests and "underneath" them, as it were. Thus any errors in the Halt and restart procedure are detected as data errors by the higher-level tests. (Recall that such a Halt and restart sequence is supposed to be "transparent" to any on-going data transfers.)

C.3 Section B - Cooperative Tests

The most complex group of tests is the cooperative. To carry out these tests requires the active participation of both machines.

These tests may be used in either of two modes: auto-sequencing and nonauto-sequencing. In the nonauto-sequencing mode, the test to be performed is selected by switches on each computer. Then the two tests are started simultaneously (actually 1800 first). Only one test may be selected and it will continue indefinitely. If any error is detected by either machine it will probably be necessary to restart the test manually. After describing the individual tests, the auto-sequence mode will be described. (Note that certain register tests and block-transfer tests can also be performed simultaneously by both computers; see C.2.1 and C.2.2.)

There are five tests in this group selected by switches 6 through 10. Switch 17 down indicates nonauto-sequence mode.

To describe these tests, the behavior of each machine will be alternately described. These tests are constructed so that each action by one machine is used as the cue for the other machine to proceed. Thus, failure of any response will eventually be caught by a time-out at each step in the opposite machine. Further, these tests are symmetric with respect to the two CPU's.

Switch 6 - Cycle All Attention Codes - (B1). This tests the ability to write and read all 32 values of the attention code portion of Status.

PDP-7: Zero control variable.

(T1L) PDP-7: Write into attention register and set attention to 1800.

1800: Read attention code and compare with expected value. Echo complement of expected value and clear own attention bit. Set interrupt to PDP-7.

PDP-7: Read code and compare with expected value. Increment control variable and set into own attention code. Clear own attention bit. Go to (T1L).

Switch 8 - Test & Set Immediate - (B3). This sequence tests the ability of the Test & Set command to resolve seize attempts with and without the Interface previously seized.

(T3L) PDP-7: Attempt Test & Set with the Interface clear. Read Status to verify that it worked.

1800: 1800 sees TSET7 bit and attempts its own Test & Set. With PDP-7 in control, this should not work. Read Status to verify. Then clear TSET7 and immediately do Test & Set. This time it should work. Verify.

PDP-7: Sees TSET18 and attempts to do Test & Set. Should not work. Verify. Then clear TSET18 and go to (T3L).

Switch 9 - Test & Set Pending - (B4). This exercise tests the ability of the pending seize to wait if the Interface is already seized and to generate an interrupt when Interface becomes available. Only Status is checked; actual interrupt is assumed from test A5.

(T4L) PDP-7: Do Test & Set (Pending) command which works immediately setting TSET7+ENBL7+OPCMPL in Status. Verify. Reset ENBL7+OPCMPL leaving TSET7. Set ATN18 for 1800 cue to proceed.

1800: Clear ATN18. Attempt Test & Set (Pending) which should have no effect immediately. Verify only TSET7 in Status. Next clear TSET7 which allows the pending seize to occur setting TSET18+ENBL18+OPCMPL in STATUS. Verify. Reset all 3 bits. Try Test & Set (Pending) again, and verify that it works.

Clear ENBL18+OPCMPL. Set ATN7 for PDP-7 cue.

PDP-7: Clear ATN7. Verify only TSET18 in STATUS. Attempt Test & Set. No effect immediately. Then clear TSET18 which permits pending seize to occur. TSET7+ENBL7+ OPCMPL should be set in Status. Clear all three bits. Go to (T4L).

Auto-Sequence Mode: In the auto-sequence mode, the PDP-7 acts as a simple controller to tell the 1800 which test is to be performed and for how long. Each test in group B whose switch is up is performed a

fixed number of times, then a new test begun. If an error is detected by either machine, a type-out is given and then it waits for a flag from the other machine saying that it too has detected an error. Then the PDP-7 starts a new test from the beginning. The error action of one machine will prevent if from continuing, thereby guaranteeing an error in the other machine.

The start-up sequence is as follows: After the PDP-7 message at the beginning of part B, select the switches for the desired tests. Then start the PDP-7 just before the 1800. After clearing the Interface (BLAST), the PDP-7 writes into the AREG the number of the test to be performed. When the 1800 sees this number it clears the AREG and proceeds to the given test. The test is normally terminated by the PDP-7 setting the mode bit in the Status Register; when the 1800 senses this bit, it clears it and waits to begin the next test. Then the PDP-7 gives a new test number in the AREG and so the world goes 'round.

If an error is detected, no attempt is made to continue from the point of the error in the current test. Rather an error message is given, then PDP-7 waits for the free bit to be set by the 1800, and the 1800 waits for the mode bit to be set by the PDP-7. When both have occurred, a new test is begun from the top. If a time-out occurs during this wait, it is considered a fatal error and the PDP-7 halts. A manual restart is necessary.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

Concomp Project
University of Michigan

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE

DEC PDP-7/IBM 1800 High Speed Interface

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Technical Report

5. AUTHOR(S) *(First name, middle initial, last name)*

J. L. Foy, Jr.
R. F. Brender
D. R. Frantz
J. A. Miller

6. REPORT DATE
August 1970

7a. TOTAL NO. OF PAGES

7b. NO. OF REFS

8a. CONTRACT OR GRANT NO.
DA-49-083 OSA 3050

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
Technical Report 31

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

Qualified requesters may obtain copies of this report from DDC.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Advanced Research Projects Agency

13. ABSTRACT

ABSTRACT

This report describes an interface between an IBM 1800 computer and a DEC PDP-7 computer. It has the following features: 1) it allows the transfer of blocks of data directly from the memory of one computer to the memory of the other, at up to 125,000 words per second, in parallel with program execution; 2) it allows a program running on one machine to interact asynchronously with one running on the other through a system of "attention" interrupts; 3) it is relatively simple to control, being symmetric to both computers; 4) it compensates automatically, in either of two modes, for the difference in word length between the PDP-7 and the 1800.

DD FORM 1 NOV 65 1473

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT