IS-T-830

A SPECIAL PURPOSE COMPUTER TO COMPUTER INTERFACE

Benjamin Franklin Carter III

Based in a M.S. thesis submitted to Iowa State University

Ames Laboratory, USDOE
Iowa State University
Ames, Iowa 50011

Date Transmitted: January 1979

PREPARED FOR THE U. S. DEPARTMENT OF ENERGY
UNDER CONTRACT NO. W-7405-eng-82

Available from: National Technical Information Service
U. S. Department of Commerce
P.O. Box 1553
Springfield, VA 22161

Price: Microfiche $3.00

15-T-830

A special purpose computer to computer interface

by

Benjamin Franklin Carter III

A Thesis Submitted to the
Graduate Faculty in Partial Fulfillment of
The Requirements for the Degree of
MASTER OF SCIENCE

Major: Electrical Engineering

Approved:

[illegible]
____
In Charge of Major Work

____
For the Major Department

____
For the Graduate College

Iowa State University
Ames, Iowa

1975

TABLE OF CONTENTS

A special purpose computer to computer interface

Benjamin Franklin Carter III

The specific design criterion are presented for interfacing two dissimilar computers. The discussion also includes hardware and software protocol for communication between the systems.

INTRODUCTION

The work reported in this thesis was the result of upgrading the Ames Laboratory Research Reactor computer system. The original ALRR computer system (Figure 1) was designed and installed in the mid 1960's. Hardware included an SDS-910 computer interfaced to an IBM 1401 computer. The SDS-910 computer controlled various experiments while the IBM-1401 merely interfaced the SDS-910 to mass storage peripherals. The need to upgrade this system was due to the fact that the 910 computer was core-limited to the extent that scheduling of experiments was necessary to prevent system overloads. Disk space was entirely committed. Failures of the IBM peripherals were occurring more and more frequently and these failures were compounded by the fact that spare parts were becoming scarce.

The proposed system (Figure 2) was designed around a PDP-15 computer using an existing real time PL/1 software operating system. The SDS-910 computer would be maintained due to the remoteness of the experiments and the capital investments in existing experiment interface hardware. In the proposed system the SDS-910 computer takes on the role of an interrupt handler and front end processor. All users' programs are written in PL/1 and execute in the PDP-15. The subject of this thesis is the interface between the PDP-15 computer and the SDS-910 computer.

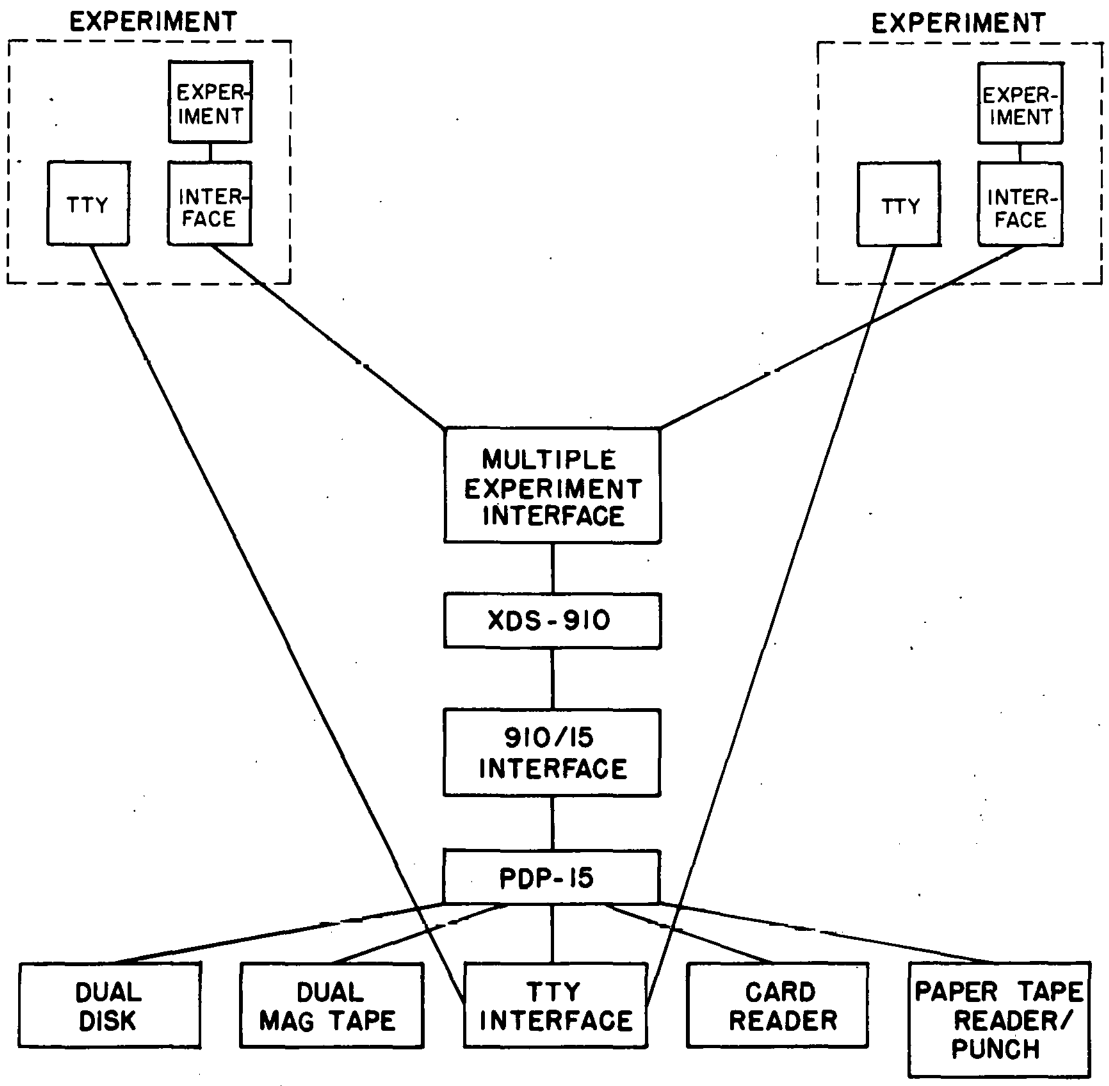


Figure 1. The original ALRR computer system

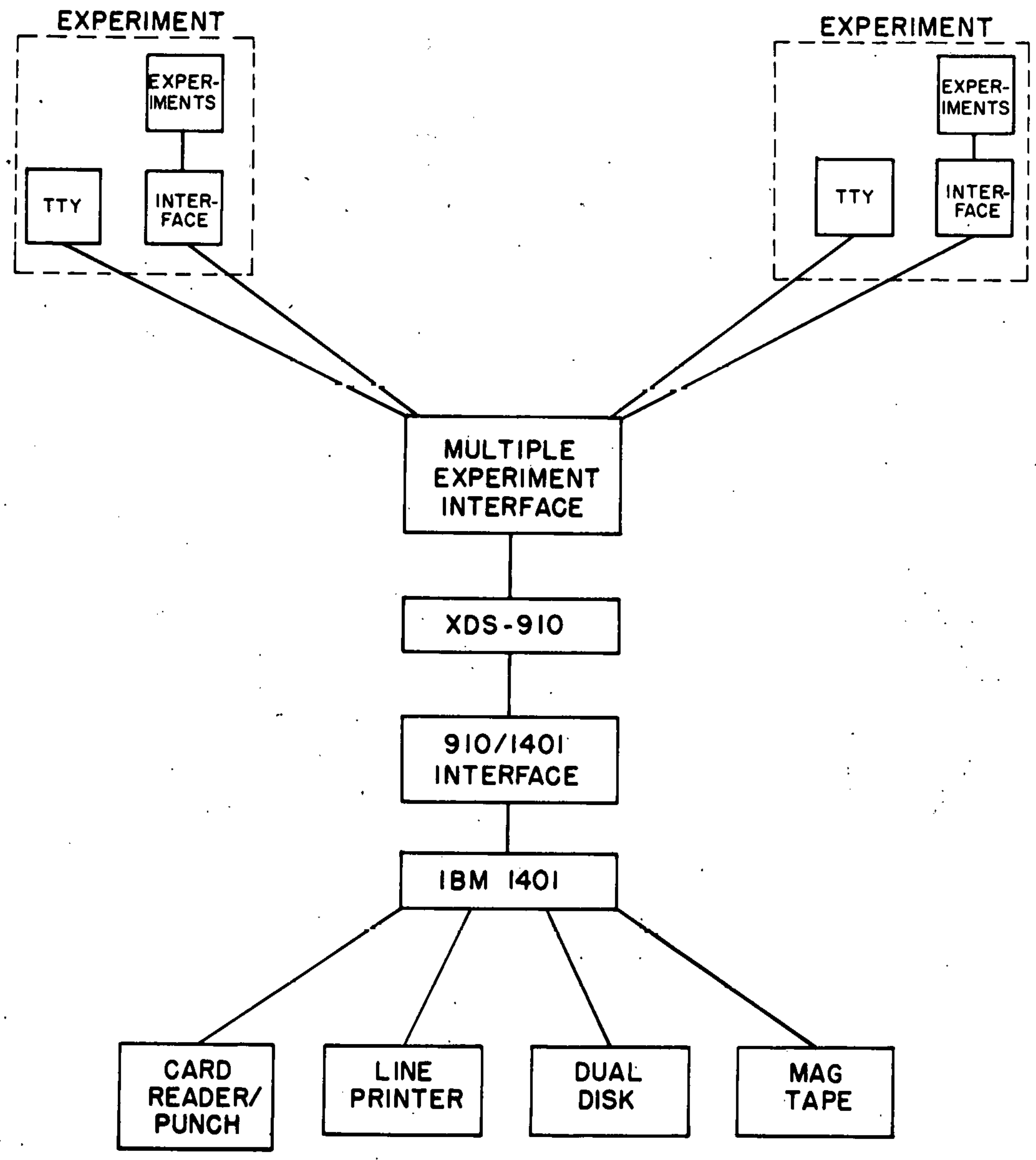


Figure 2. The proposed ALRR computer system

SDS-910 COMPUTER

The SDS-910 computer has a word length of 24 bits, a memory cycle time of 8 microseconds, and is constructed of discrete DTL circuit components. The 910 computer allows a maximum of 16K words of core memory, all directly addressable. It contains a priority interrupt system capable of handling 896 general purpose interrupts. The SDS-910 input/output system is capable of the following types of interaction:

1. Buffered input/output of data words, each under direct program control.
2. Input/output of blocks of characters (6 bits) or words time shared with memory and multiplexed with computation using interlaced buffers.
3. Direct parallel input/output of up to 24 bits of information to and from external registers under program control.
4. Single bit input/output.

PDP-15 COMPUTER

The PDP-15 computer uses an 18-bit word, has a memory cycle time of 800 nanoseconds, and contains TTL integrated-circuit construction. The computer contains three autonomous sub-systems:

1. Central processor
2. Memory
3. Input/output processor

All operate concurrently in overlapping cycles under console control.

The central processor is the main component of the computer, carrying on bidirectional communication with both memory and the I/O processor. Provided with the capability to perform arithmetic and logical operations, the central processor controls and executes stored programs.

Core memory is the primary storage area for the PDP-15 computer. It is organized into pages which are paired into memory banks. Each page contains 4,096 words of magnetic core storage, and each bank is physically an asynchronous unit of 8,192 words. The central processor can address up to 128K (131,072) words of core memory. Any word can be addressed by either the central processor or the I/O processor.

The I/O processor handles all peripheral communication. This processor contains three possible modes of input/output:

1. Program-controlled single word transfers through

the accumulator in the central processor.

2. Multicycle block data transfers at rates up to 250,000 words per second input and 188,000 words per second output using three memory cycles per word transferred.
3. Single-cycle block data transfers at rates of up to 1,000,000 words per second using one memory cycle per word transferred.

The I/O processor provides timing, control, and data lines for information transfers between the memory of the central processor and the peripheral device. It also includes an automatic priority interrupt system.

DESIGN CONSIDERATIONS

The foremost design consideration is to transfer the data between the two computers efficiently and conveniently. It appeared that most transfers would either be very small (two or three words) or extremely large (thousands of words). The operating system is to have experiment-controlling programs in the SDS-910 computer and "number crunching" programs in the PDP-15. As an example of this interaction, the experiments driven by the system contain detectors that record the number of neutrons present. The detectors are positioned to a given point in space by the SDS-910 computer. The PDP-15 computer calculates the proper position and passes it to the SDS-910. This small transfer between the two computers would contain a user identifier and one word of data. The 910 starts the detector moving and receives reports of its present position through interrupts, typically one every one hundredth of a degree. When the detector reaches the proper position the SDS-910 stops its motion.

Examples of a large transfer would be loading the 910 operating system or reading an entire spectrum of data. These involve thousands of words. The PDP-15 can accomplish the three types of transfer briefly described earlier (program-controlled transfers; multicycle block transfers; single-cycle block transfers). The primary distinction between the two modes of block transfer is that the multicycle mode has word count and current memory address pointers resident in the memory of the computer while the single cycle mode requires that the word count and current address pointer be maintained by the peripheral device. The word count contains the 2's complement of the number of words remaining to be transferred. The current memory address pointer records the memory address of the word currently being transferred. Block transfers are useful in transferring large amounts of information because they communicate directly between memory and the peripheral device. However, a penalty is paid in terms of overhead required to initialize the word count and current memory address pointer registers. In contrast, the program-controlled transfers require no overhead but the transfer uses the accumulator of the central processor. Thus there is the implication that a certain amount of care must be taken to preserve the contents of the accumulator either before or after the transfer has occurred. The amount of central processor time required for a given length transfer is plotted in Figure 3 for the three types of PDP-15 transfers. The amount of I/O Bus time required for a given length transfer is plotted in Figure 4. It can be seen from the plot of central processor time, that the program-controlled transfer is by far the best method when the total transfer is less than seven words. However, the single-cycle transfer is obviously preferred for large transfers where the overhead

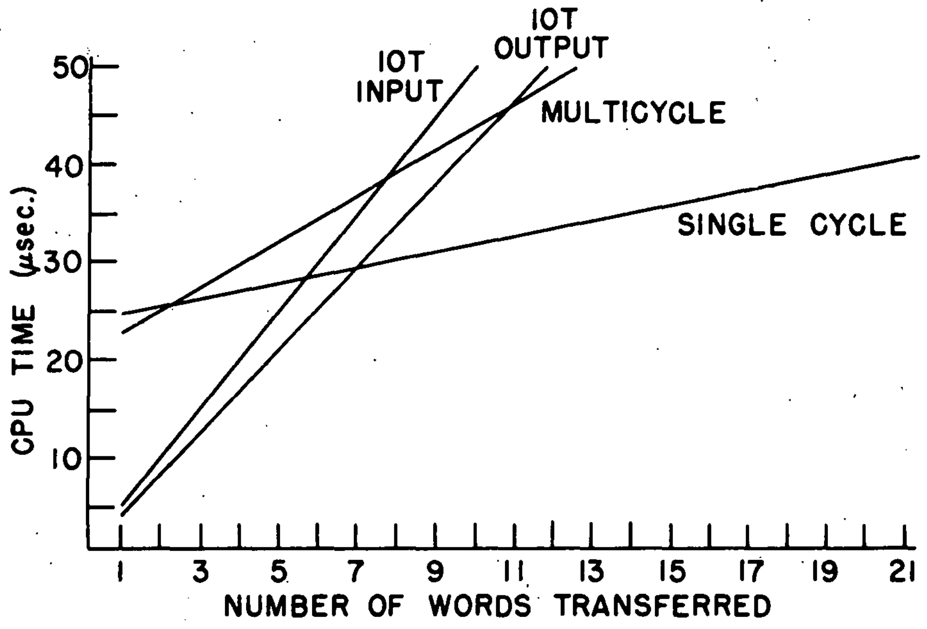


Figure 3. PDP-15 CPU time versus number of words transferred

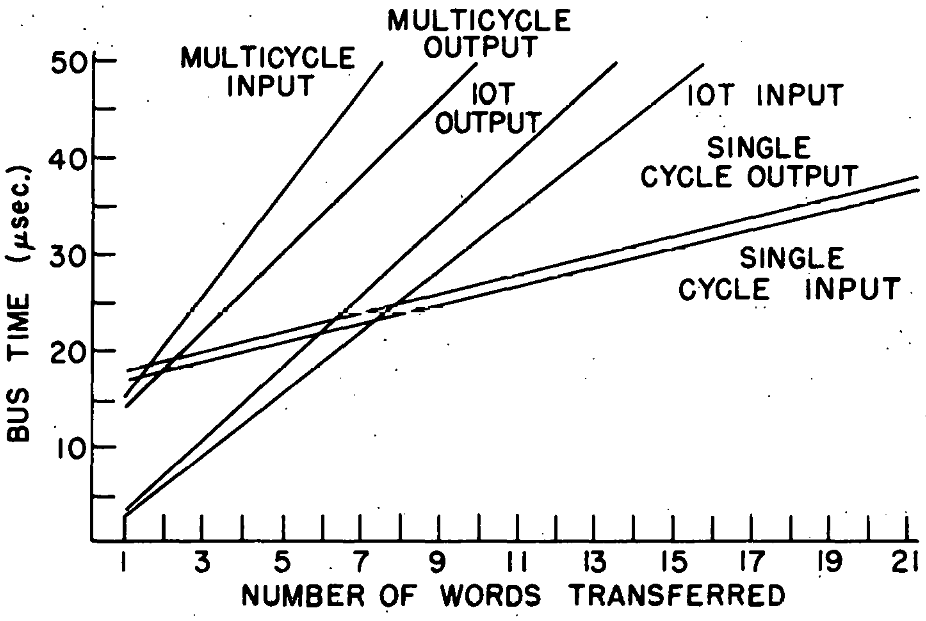


Figure 4. PDP-15 I/O bus time versus number of words transferred

involved in initiating the transfer is small compared to the number of words transferred. Similar observations apply to the plot of the time the bus is occupied (Figure 4). On this basis program-controlled transfers were used in passing small blocks, while the single-cycle block transfer method was chosen for transferring large blocks of data.

The SDS-910 computer offers several methods of input/output:

1. Buffer interrupt system

   A buffer assembles and disassembles data words as they are transmitted between core memory and the peripheral equipment. The buffer maintains control of operations such as the number of characters per word transmitted.

   The W buffer performs input/output of data words, each under program control. The buffer transmits and receives 6-bit characters, and packs them into 24-bit words. A second buffer, the Y buffer, is available and is essentially identical to the W buffer, but will allow the character length to be defined to any desired length between 6 and 24 bits. Transfer rates can reach 32 microseconds per word for input and 24 microseconds per word for output.

2. Buffer interlace system

Each buffer may have a hardware interlace associated with it. Interlace allows input/output of blocks of data words, word transmission being completely automatic and multiplexed with computation. The interlace supplies the memory address of data coming from or going to memory and maintains the word count determining the number of words transferred. The maximum number of words per transfer is 4,095. The character manipulation occurs in the buffer, consequently the memory sees only complete 24-bit words. Transfer rates will reach 16 microseconds per word plus 16 microseconds per character.

3. The word parallel system

The word parallel system allows for transfers of 24-bit words under program control. The parallel input requires 32 microseconds per word and the parallel output requires 24 microseconds per word.

4. The parallel interlace system

Here again the interlace provides a word count and current address pointer. Words are transferred 24-bits at a time, under interlace control, at a rate of 8 microseconds per word with an upper limit of 4,095 words per transfer. Unlike the buffered interlace the parallel interlace prevents the cen-

tral processor from operating concurrently.
Table 1 shows the relative speeds of each of these methods of transfers.

Table 1. SDS-910 I/O transfer times (microseconds per word)

	Without Interlace	Using Interlace
W buffer	32 input	80
	24 output	
Y buffer	32 input	32
	24 output	
Word Parallel	32 input	8
	24 output	

The main contender for rapid communication is the parallel interlace system but the parallel interlace system is unsuited for use in the proposed system primarily because it would lockout the central processor and all other input/output while a transfer is taking place. This is completely intolerable in a real time environment. Also the Ames Laboratory SDS-910 computer does not support this optional method of transfer, nor does it support a Y buffer.

This leaves the word parallel system as the fastest option available. The W buffer is eliminated because transfers occur there as 6-bit characters and not as entire 24-bit words. This would greatly increase the gating circuitry in the interface if it were used. All experiments now use the word parallel system of communication and considerable interfacing work has been done with it. The PDP-15 would look much like any other experiment to the SDS-910, thereby taking advantage of the interfacing already done to the 910 computer. With the word parallel system each device is assigned a unique address. Three types of instructions can be executed in connection with these device addresses:

1. PIN-parallel input of 24 bits of data.
2. POT-parallel output of 24 bits of data.
3. SKS-test and skip the next instruction if true.

Prior to executing any of these instructions an EOM (Energize Output M) must be issued. This instruction determines which instruction will follow it and also alerts the peripheral device that a PIN, POT, or SKS will follow.

The PDP-15 computer will communicate in two modes with the SDS-910 computer; program-controlled transfers and single-cycle block transfers. The PDP-15 will communicate efficiently with the SDS-910 thus freeing it for system operations and computation. The 910 on the other hand will operate more inefficiently but will be able to perform all of its external experiment-control functions and communicate with the PDP-15.

The vast differences in speed require the use of some type of data buffering in the interface. Along with this is the difference in word lengths with which one must also contend. The size of the buffer required is directly related to the size of the transfers to be made. The smallest transfers across the interface would be one data word long.

Table 2. Comparison of transfer rates

	Max Transfer Rate (words/sec)	Bits per Word
PDP-15 single cycle block transfer	1,000,000	18
PDP-15 program controlled transfer	234,192	18
SDS-910 word parallel system		
input	31,250	24
output	40,000	24

The actual length of a one data word transfer, including all of the software key words, is three 18 bit words for integer data and four 18-bit words for floating point data. The first word is a key word giving a user identifier and status concerning the transfer. The second word is a special address concerning that user and the PDP-15 operating system. The data comprises both the third and fourth words. This type of transfer is more common than block transfers with many data words.

The most logical way to specify the size of the buffer is to make it the same size as the number of words most frequently transferred. In this way, the sending computer could load the entire buffer in essentially one pass and at its own speed. The receiving computer would then be able to read the entire transfer at its own speed. Some overhead will be required in acquiring the interface to make a transfer but this would need to be done only once. By making the buffer large enough to contain four 18-bit words (72 bits of data), which is the equivalent of three 24 bit words, the interface maintains the integrity of the information, but the transmitting computer is forced to pack the words properly for the receiving computer.

Large block transfers will occur in bursts of four 18-bit words at a time. The size of the buffer need not be affected by these block transfers because they will occur infrequently while the system is operating, however, the interface can handle them. Secondly, most of the time spent during a block transfer will be taken waiting for the SDS-910. As the PDP-15 can transfer data more than thirty times faster than the 910, it would appear that the data will be transferred at the rate of the 910. By keeping the buffer small, the bursts of data on the PDP-15 I/O bus will be small minimizing the time the I/O bus is occupied.

The interface will be considered a peripheral device of both machines. Both the PDP-15 and the SDS-910 must be able to initiate transfers through it. Because of this, a scheme for requesting the use of the interface is needed. By using a simple flip-flop to indicate whether the interface is busy or not, and by providing the means for both computers to test, set, and clear this flip-flop, the state of the interface can be determined by both computers. Due to the differences in speed, however, the SDS-910 could test BUSY and receive a signal back that the interface was not being used. Before the SDS-910 could set the BUSY flip-flop, the PDP-15 could also test BUSY, find the interface is free, and set the BUSY flip-flop. Both machines would think they had acquired the interface. To alleviate this problem a second flip-flop called REQUEST was added. When the SDS-910 wants to make a transfer it first sets the REQUEST flip-flop, then tests BUSY. The PDP-15 is required to test REQUEST prior to testing BUSY. If it finds that the SDS-910 is waiting or in the process of acquiring the interface, it is forced to wait until REQUEST is cleared. The REQUEST flip-flop is cleared after the SDS-910 acquires the interface buy setting the BUSY flip-flop. This eliminates the conflict in acquiring the interface due to the difference in speeds.

The interface needs to know what type of transfer and which computer is sending the information in order to set up the proper gating within the interface. An interface control word was developed for this purpose and is shown in Figure 5. Prior to each transfer the initiating computer writes the control word into the interface. There are three bits in the control word that directly effect the transfers in the interface. The 910 INITIATE and 15 INITIATE bits indicate which computer initiated the transfer. The DMA bit indicates in which mode the PDP-15 will be transferring data. If the DMA (Direct Memory Access) bit is set the transfer is a single cycle block transfer, otherwise the transfer is under program control. The 15 LOCKUP does nothing in the interface. It was provided so that when set by the PDP-15 it would tell the 910 software that the 910 could not initiate any transfers. This allows the software to move priority data from the 15 to the 910. The two final bits in the control word are unrelated to the interface. Due to the number and varied qualifications of the experimenter operators using the computer system and due to the remoteness of the experiments

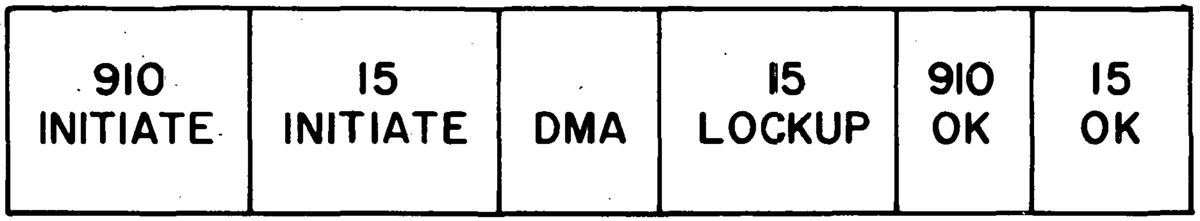


Figure 5. Interface control word

from the computer, an indication to the experimenter of the status of the computing system would be useful. Under the old reactor computer system when the computers failed, the experimenters had no indication of whether or not the system was in operation. If they could elicit no response from their instruments via computer control they would immediately start to indiscriminantly push buttons hoping to solve whatever problem existed. Whatever state the computer failed in was usually changed by this pushing of buttons, causing severe maintenance problems. The 15 OK and 910 OK bits are set only by their respective computers and are automatically cleared every 1.6 seconds. The computers set these bits every time they cycle through their respective dispatchers. If either computer stops or gets hung in a loop, that OK bit will not get set. It is obvious that some failure could leave the computer still setting the OK bit, but chances of this are remote. The two OK bits are logically anded together to form a SYSTEMS OK which is sent to every experiment and displayed with an indicator. Even though shown as part of the control word, the OK bits are set by their own instructions but may be read by either computer as part of the control word.

The various combinations of transfers across the interface are flowcharted in Appendix C and D. A brief explanation of these transfers follows.

A transfer from the SDS-910 to the PDP-15 is initiated by the SDS-910. The 910 sets the REQUEST flip-flop, and then tests 15 LOCKUP. If it is not set, the 910 proceeds to test BUSY. If the interface is free BUSY is set and REQUEST is cleared. The control word is loaded. Finally three 24-bit data words are written in the interface buffer. If the transfer is not a DMA transfer, the 910 is done. The interface causes an interrupt of the processing of the PDP-15 computer. Upon acknowledging the interrupt the PDP-15 reads the control word and the 72 bits of data, clears the BUSY flip-flop, and the transfer is complete.

Had the transfer been a DMA transfer, the 910 would have set up a software word count and a current address pointer. It then would have tested a flip-flop which indicates whether or not the 15 has read the data. The 15, meanwhile, would have received the interrupt and read the control word and the 72 bits of data. It would then load the outboard word count and current memory address registers and start a DMA transfer. The central processor is relieved of any duties relative to the transfer as the input/output processor takes control. After detecting that the 72 bits of data have been read, the 910 loads three more words into the interface, modifying its software word count and current memory address pointer correspondingly. The interface starts up the PDP-15's DMA transfer and the 72 bits are read. This continues until the last word is read by the PDP-15 at which time an interrupt of the central processor occurs (generated from the word count register in the interface) and the PDP-15 clears the BUSY flip-flop.

A transfer from the PDP-15 to the SDS-910 is initiated by the PDP-15. The PDP-15 must first disable its interrupts. This is required because the PDP-15 could test REQUEST and find it not set and proceed to test BUSY. An interrupt of the PDP-15 could occur at this time and this interrupt would be serviced immediately if the interrupts were enabled. During this time the SDS-910 could have set REQUEST and found that the interface was not BUSY. The 910 would set BUSY and at the same time it would be possible for the 15 to be released from its interrupt routine and it also could set BUSY. Both machines could think that they had control of the interface.

The same net result can occur when the 15 LOCKUP flip-flop is being used by the software. If the 15 LOCKUP is set and the 15 is ready to test BUSY but gets interrupted, it is entirely possible that the interrupt-handling routine will clear the 15 LOCKUP, allowing the 910 access to the interface. Here again both the 15 and the 910 could simultaneously test BUSY, find it not set, and set it. Disabling the PDP-15 interrupts for the testing of REQUEST and BUSY guarantees that the interface can be acquired by only one computer.

Once the PDP-15 has acquired the interface it loads the control word and 72-bits of data which automatically causes the interface to interrupt the SDS-910. If the transfer is not a DMA, the PDP-15 is done. In servicing the interface interrupt the 910 reads the control word and 72-bits of data. This generates another interrupt to the PDP-15 indicating that the transfer is complete and the 15 clears BUSY.

If the transfer was a DMA transfer, the PDP-15 continues to load the outboard word count and current memory address registers and initiates a DMA transfer. The 910 is interrupted as before and services this interrupt by reading the control word and the 72-bits of data. The 910 sets up its own software word count and current memory address pointer. Once the 910 has read the 72-bits of data the 15 immediately loads anothers 72-bits of data. This transfer and all ensuing transfers are under the control of the input/output processor until the word count is exhausted. A flip-flop, BUFFER FULL, is used by the 910 to test to see if the PDP-15 has loaded the next 72-bits of data. Once it has, the BUFFER FULL flip-flop is set and the 910 reads the next 72-bits of data. BUFFER FULL is used exclusively by the 910 to know when the data is ready during a DMA transfer. When the hardware word count of the 15 overflows and the 910 has read the last word of the transfer (its software word count also being zero) an interrupt of the 15 is generated by the interface. This alerts the 15's central processor that the transfer is complete, and the 15 clears the BUSY flip-flop. To generate this interrupt, the interface hardware requires that no matter when the 15's word count overflows, the 910 must read the full 72-bits of data stored in the buffer even if the 15 has loaded only one word (of the three). This frees the interface from having to know exactly when to generate the interrupt of the 15 (if the 15 loaded only 36 of the 72 bits and the word count overflowed, the 910 would read all 72 bits. The 910 would know which bits were valid because of its own software word count.).

The same requirement is made of the 910 when the transfer goes from the 910 to the 15. The 910 must always completely fill the buffer even though its word count goes to zero in the middle of the 72-bits. This allows the interface to start and stop the 15's input/output processor more easily. Also the interface has no way of knowing when the 910 has transferred its last word in a DMA because this is recorded only in the memory of the 910. The 15, in this case, will read only the required number of words (until its word count overflows). Since the 15's word count resides in the interface, proper termination of transfers can occur whenever the word count overflows because of the access to the signals. An alternative to this might have been an instruction that was issued by the 910 to indicate that its word count was zero. It is not possible to use the 15's word count because it is stored in terms of 18 bit words not 24 bit words and is not changed while the 910 is accessing the data buffer in the interface.

HARDWARE CONSTRUCTION

The interface is constructed, in part, with standard Digital Equipment Corporation (DEC) Logic Modules and in part with Scientific Data Systems (SDS) Logic Modules. The decision to use purchased modules was forced due to the lack of manpower to construct in-house modules, limited time available, and a limited budget. Ames Laboratory already owned a considerable stock of SDS Logic Modules and associated hardware and none needed to be purchased. Because of this the majority of the interface is constructed with SDS Logic Modules. The interface to the PDP-15's I/O bus was constructed with DEC Logic Modules. This part of the interface contains some rather critical timing and DEC provides modules that directly solve all problems. The DEC logic is built from TTL integrated circuits and uses a 3 volt logic true. On the other hand SDS logic is discrete component logic and uses 6 volt logic true. The conversion between the two types of logic is done with a standard SDS Logic Module.

In the PDP-15 program-controlled transfers occur as the result of the IOT (input/output transfer) instruction execution. These instructions are microcoded to effect a response of a specific device on the I/O bus. The microcoding includes unique device selection codes and appropriate pulses to initiate device operations such as transmitting data from the device to the central processor, or from the processor to the device. All program-controlled transfers are executed through the accumulator. This portion of the I/O processor also contains facilities for skipping on device flags and interrupts. Both of these features were used extensively in the PDP-15/SDS-910 interface.

The format of the IOT instruction shown in Figure 6 consists of a 6-bit device select code and a 2-bit subdevice select code. These codes are placed on the I/O bus to indicate with which device the processor wishes to communicate. It also includes a clear the accumulator option if bit 14 is set, and three I/O pulses, any one or all of which may be given in a single IOT instruction. The normal use of these pulses is as follows; IOP 1 transmits data to the device, tests the device flag, and causes the program to skip the next sequential instruction; IOP 2 transmits data to or from the device; IOP 4 transmits data to the device from the accumulator.

The PDP-15/SDS-910 interface uses three of the IOT device selection codes including: device select 14, device select 15, and device select 16. A complete listing of the IOT instructions can be found in Appendix B. In general, device select 14 was used to read and write each of the four data words in the interface; device select 15 was used in reading, writing, and testing the various controlling flip flops in the interface; device select 16 was used in control-

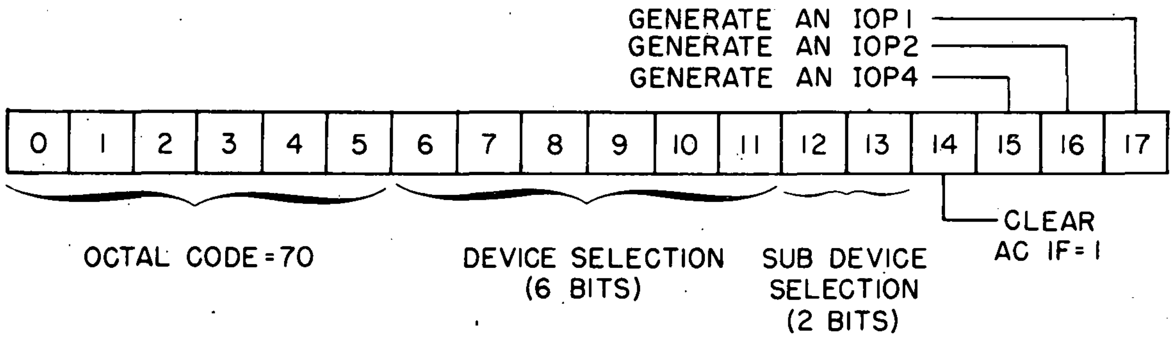


Figure 6. PDP-15 IOT instruction format

ling and initiating the single-cycle block transfers.

The relative timing of the IOT instruction signals on the I/O bus can be found in Figure 7. I/O SYNCH is a synchronizing signal present on the I/O bus and every transfer is carried out in synchronization with it. IOT REQUEST is an internal I/O processor signal generated when an IOT instruction is encountered by the central processor.

Communication between the SDS-910 and peripherals using the word parallel system is accomplished by the execution of a pair of instructions. The first is an EOM (Energize Output M) which sets up the experiment address and provides advance information as to what the next instruction will be.

The format of the EOM instruction is shown in Figure 8. The W field indicates what type of instruction will follow. Axy contains the device selection code and Bz contains the sub-device codes. The EOM instruction can be followed by a POT (Parallel Output), PIN (Parallel Input), or an SKS (Skip if true) instruction. The format for each of these instructions can be found in Figure 8. The X field in the PIN and POT instructions represents the address in memory where the data is to be placed or removed. In the SKS instruction the X field is a mask field where a bit or combination of bits can be logically anded with the testing signal. A single SKS instruction may be used to test many different flags just by using the mask field to determine which circuit is to be

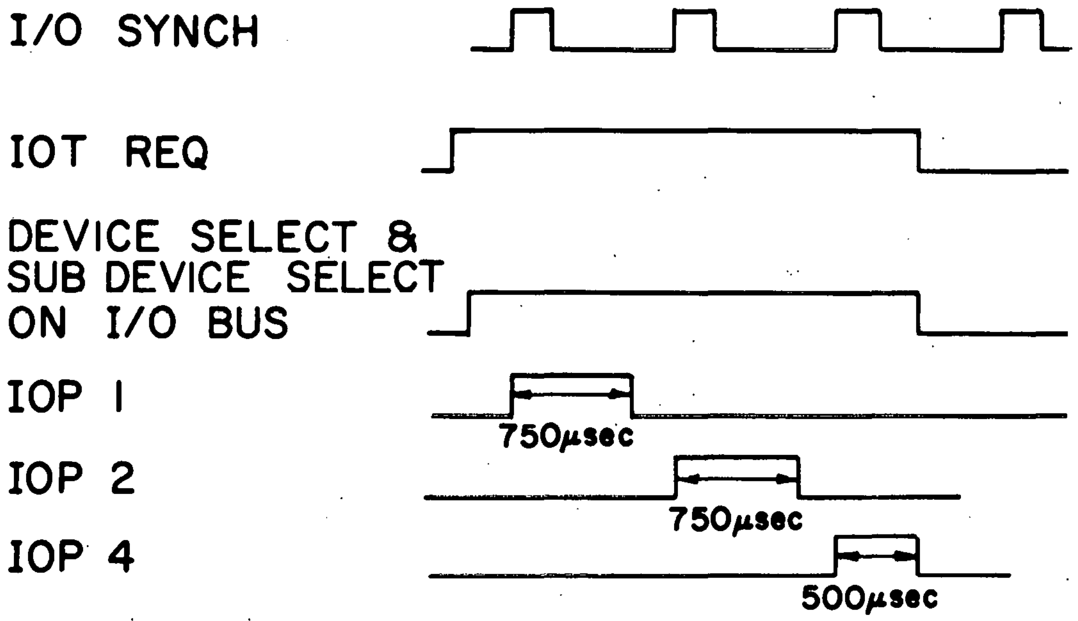


Figure 7. PDP-15 IOT instruction timing

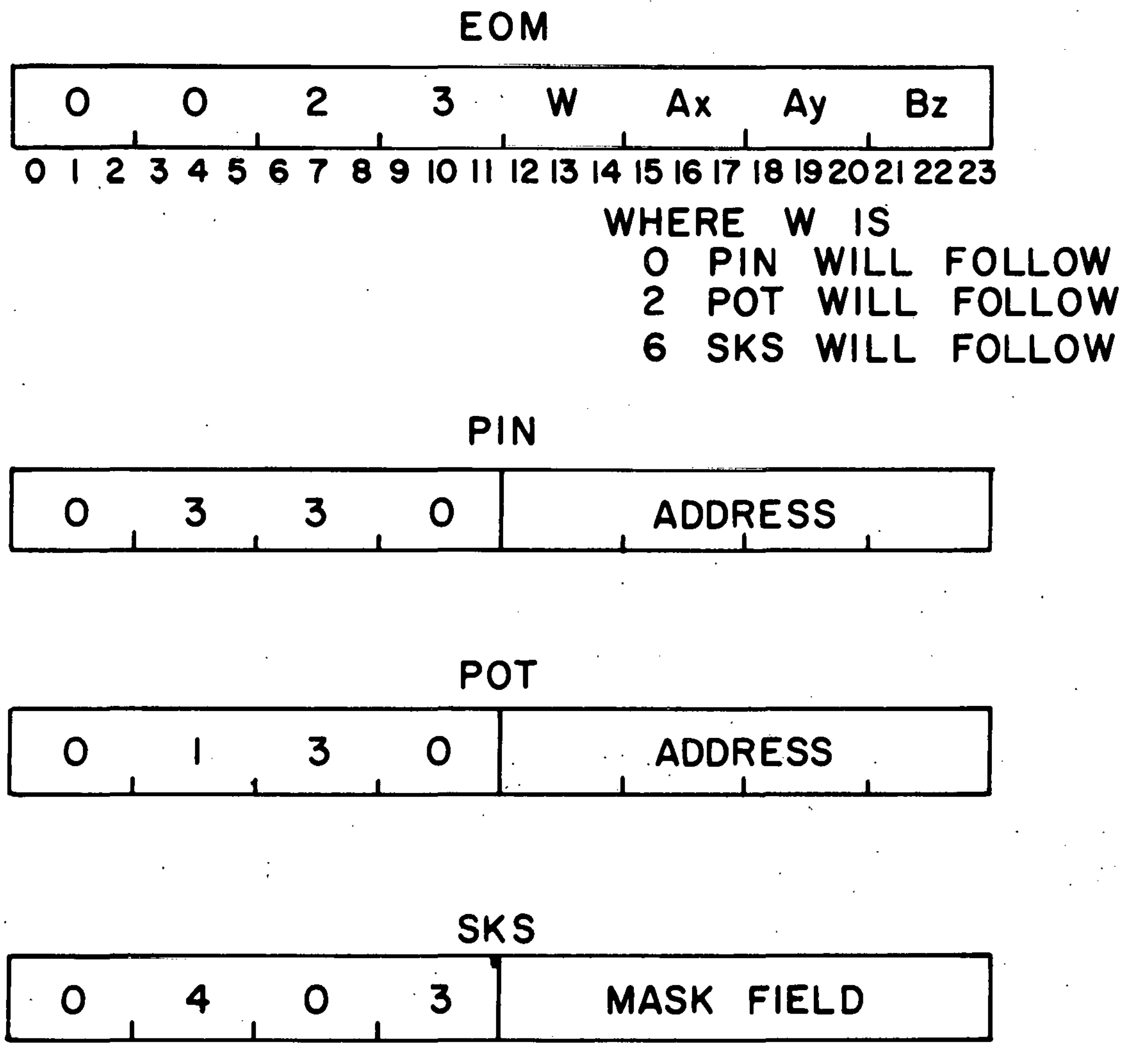


Figure 8. SDS-910 I/O instruction formats

tested. The PIN, POT, and SKS instructions automatically generate their respective gating pulses as can be seen in Figure 9.

The interface was assigned a device address (Axy) of 30. A complete listing of the instructions used by the SDS-910 to communicate with the interface can be found in Appendix A.

The instructions of both the PDP-15 and the SDS-910 as mentioned above and listed in the Appendix are relatively simple in implementaton. The PDP-15's block mode of transfer requires a more subtle hardware implementation, since it is automatic once initiated. Several IOT instructions are used in conjunction with the single cycle block transfer (DMA). These instructions are used to load the word count and current memory address pointers in the peripheral's hardware. Also an IOT is used to initiate the transfer.

Assuming that a program has written a word count, current memory address, and initiated a DMA, the interface posts a single-cycle request to the I/O processor. The I/O processor, as soon as it becomes available, acknowledges the request by returning a DATA CHANNEL GRANT. The interface then places the current memory address on the I/O bus. If data is being transferred to the processor from the interface it is also placed on the I/O bus lines at this time. The data channel controller requests a memory cycle and the current memory address sent by the interface is the address

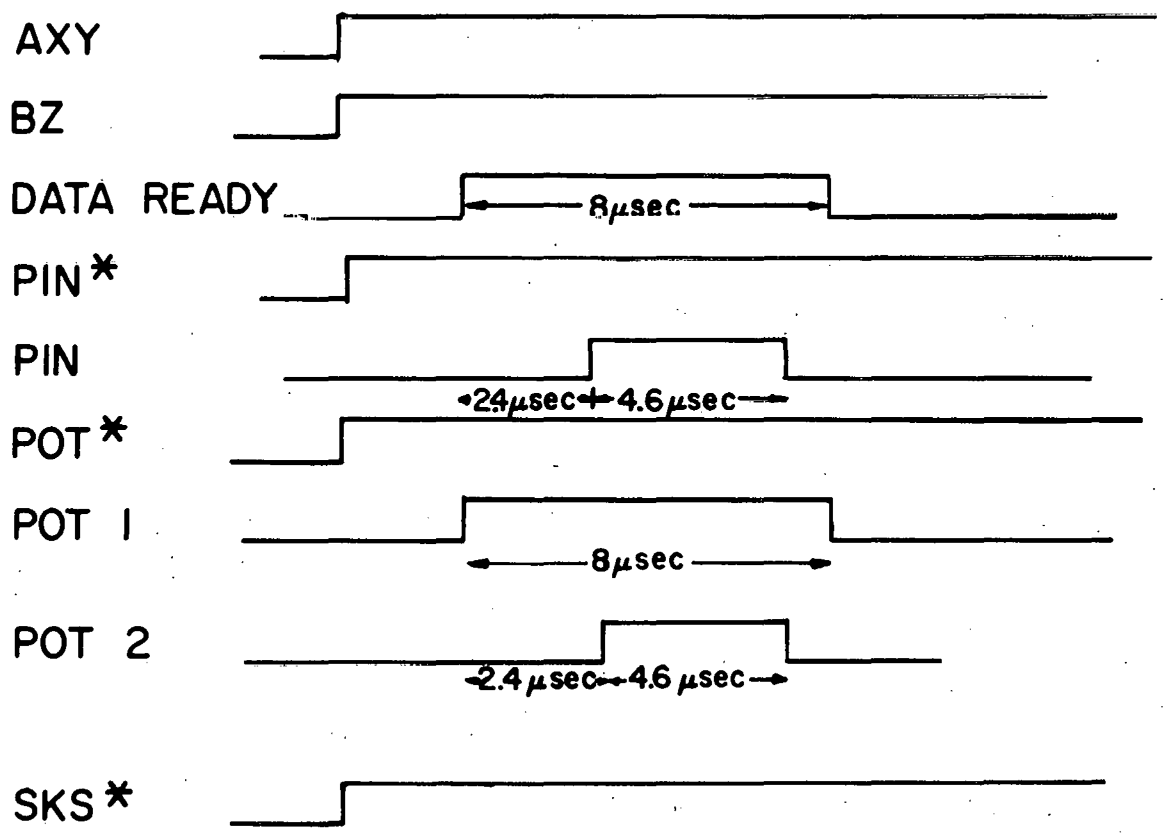


Figure 9. SDS-910 I/O instruction timing

where the data is stored. If the transfer is from the memory to the interface, data is read from the memory location specified by the current memory address from the interface and the data is placed on the I/O bus lines. During this time the interface is incrementing both the word count and current memory address in preparation for the next transfer. If the word count overflows, the interface disables itself and posts an interrupt to the 15. If the word count did not overflow the second of the four interface words is transferred and this continues until all four have been transferred. The data channel is freed until the 910 responds appropriately by either writing out or reading the 72-bits of data.

Due to the apparent automatic nature of the data channel, timing and control become critical. As the words are transferred to and from Because of this a four bit ring counter was used to gate each word to its proper destination. The ring counter is initialized when the initiate DMA IOT is issued by the 15 and the ring counter is cleared when overflow occurs.

The single-cycle transfers can operate in one of two modes: burst mode, and normal mode. In normal mode one word is transferred for each request of the data channel. In burst mode transfers, once started, continue until the data channel is returned to normal mode, usually just prior to the last word to be transferred. The burst mode transfers proceed much faster than those in normal mode due to the fact that there is only one request of the data channel for each burst of data to be transferred, whereas in normal mode the peripheral must synchronize with the data channel after every word that is transferred. The interface operates in burst mode for this reason.

DATA CHANNEL GRANT is an extremely important signal to the interface. One GRANT occurs with every word transferred. DATA CHANNEL GRANT indicates that the interface now has control of the data channel and that the transfer will occur during the time this signal is present. The interface uses the trailing edge of the DATA CHANNEL GRANT signal to generate a pulse called ADDRESS ACCEPTED. ADDRESS ACCEPTED clocks the ring counter during a DMA transfer.

For transfers from the PDP-15 computer to the interface the ring counter is clocked once prior to receiving the data due to the wait time for the current memory address pointer to get from the interface to the memory and also due to the time it takes to get the data out of memory and to the interface. The I/O processor generates a signal called IOP 4 which is used to used by the interface to gate the data into the interface buffer as shown in Figure 10. In Figure 10 the outputs of the ring counter are indicated by DMA1, DMA2, DMA3, AND DMA4. For transfers from the interface to the PDP-15 the data is placed on the I/O bus prior to clocking

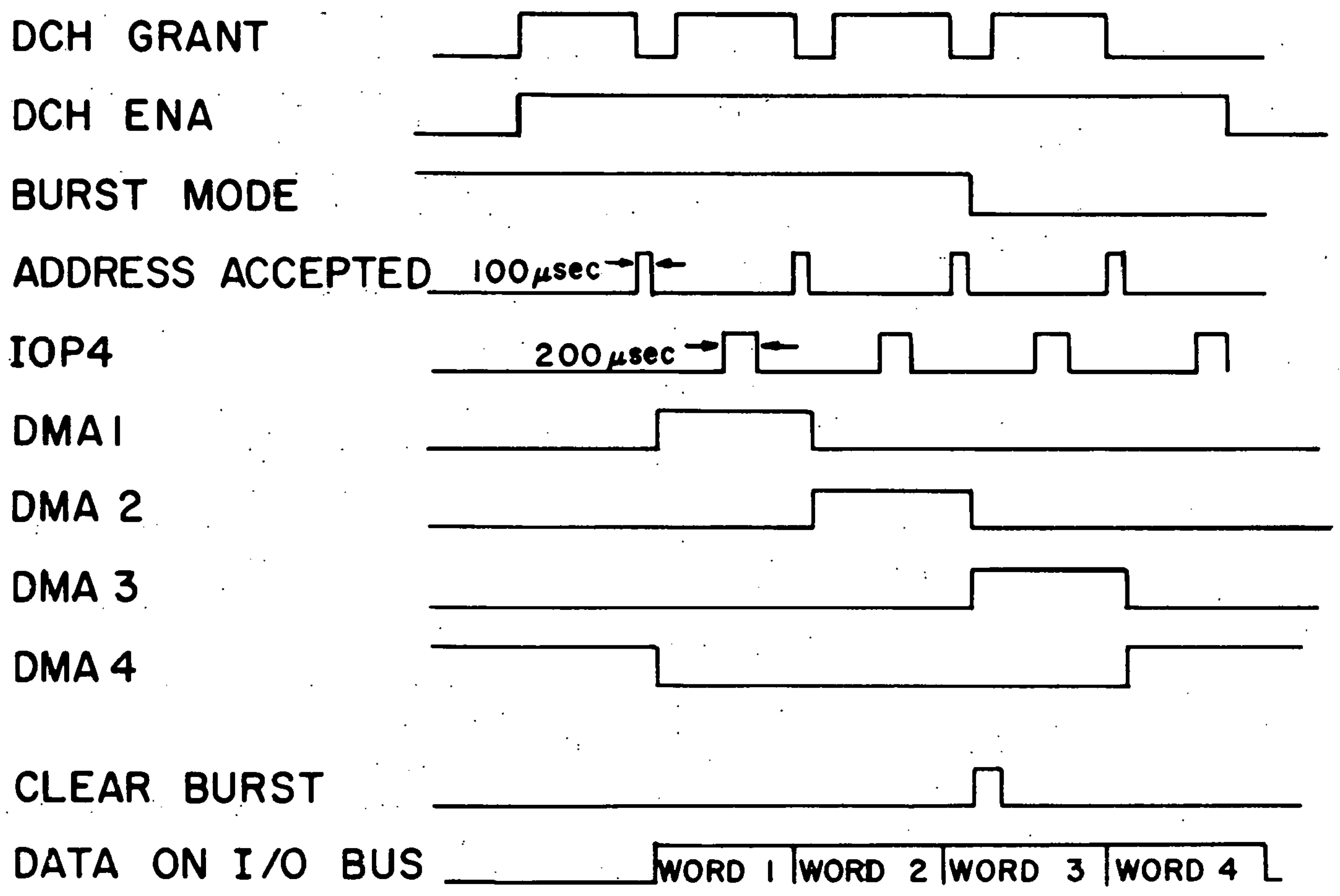


Figure 10. Data channel timing for transfers from the PDP-15

the ring counter. This can be seen in Figure 11.

The 910 is interrupted once at the start of the transfer, and from then on it must test BUFFER FULL to determine when the data is ready. The 910 must also keep track of its own word count and current memory address pointer in software. During transfers from the PDP-15 to the SDS-910 BUFFER FULL is set when the 15 has loaded four words and they are ready to be read by the 910. The 910 software would be waiting for this in a loop as follows:

```
          .

          .

          .

LOOP EOM                  /SKS FOLLOWS
     SKS  BUFFER FULL     /SKIP NEXT INSTRUCTION IF BUFFER FULL
     BRU  LOOP            /BRANCH TO LOOP
     XXX                  /READ BUFFER
          .

          .

          .
```

Transfers from the SDS-910 to the PDP-15 leave the BUFFER FULL flip flop cleared when the 15 has read the data. The 910 software wait loop would now be:

```
          .

          .

          .
```

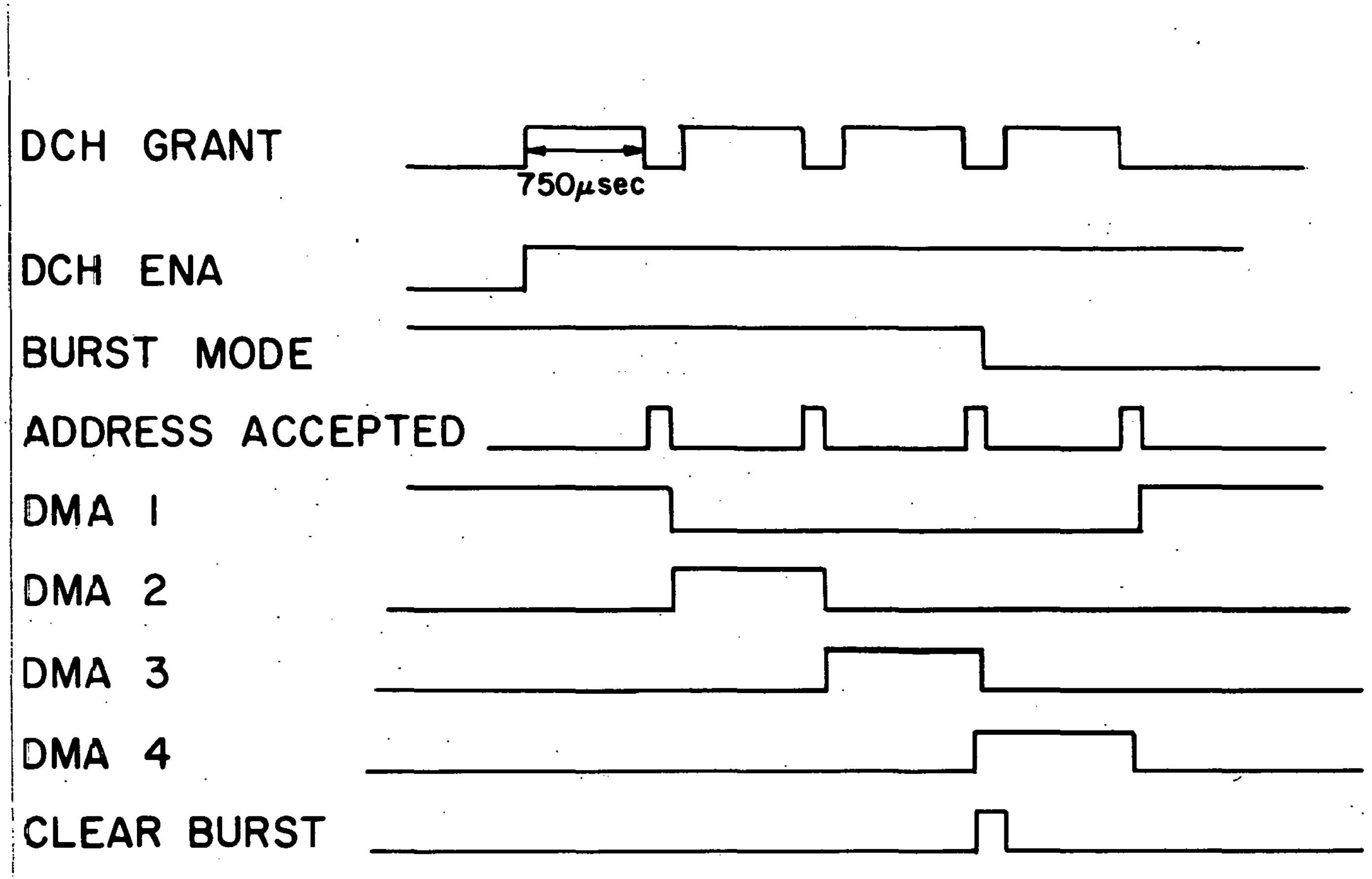


Figure 11. Data channel timing for transfers to the PDP-15

```
  LOOP EOM                  /SKS FOLLOWS
       SKS   BUFFER FULL    /SKIP NEXT INSTRUCTION IF BUFFER FULL
       BRU   EMPTY          /BUFFER IS EMPTY BRANCH TO EMPTY
       BRU   LOOP           /BUFFER IS FULL BRANCH TO LOOP
 EMPTY XXX                  /WRITE TO BUFFER
        .
        .
        .
```

The EOM SKS pair of instructions require a minimum of 16 microseconds to be executed. It will take the data channel only four microseconds to transfer the words to or from the interface once the interface has acquired the data channel. It is expected that the 910 will not have to wait in this loop for any extended length of time due to the speed of the 15.

The interface maintains a flip flop named DMA CONTINUE for the PDP-15. DMA CONTINUE acts much like BUFFER FULL in that it indicates to the 15 when the 15 should transfer data. DMA CONTINUE is used to request the data channel when the interface is ready to send or receive more data during a DMA transfer. This flip flop is inhibited by the overflow of the word count in the interface. The initial data channel request is triggered by the issuing of the IOT that initiates the DMA and this is logically anded with the DMA CONTINUE signal. If the transfer is from the 15 to the 910 the DMA

CONTINUE is set after the 910 has read the 72-bits of data. Conversely, if the transfer is from the 910 to the 15 DMA CONTINUE is set after the 910 has written the 72-bits of data.

CONCLUSIONS

The major stumbling block of this interface was centered around the use of SDS Logic Modules for much of the design. Not only is this discrete logic cumbersome to design with but it causes severe mechanical problems due to hand-soldered wiring. The manhours used in actually constructing the interface more than offset the cost of using all newly purchased TTL integrated circuit technology. As in the case of many government intallations there is a large supply of manpower but little money. Because of Ames Laboratory's large stock of SDS LOGIC MODULES they were used in the implementation of this design.

Two interfaces were constructed, and were in operation in September of 1974. The interface not previously mentioned was constructed as a prototype, to be installed on a PDP-15 experiment controlling system on the ISU campus. It was connected to an unused Ames Laboratory-owned SDS-910 computer. This protoype was to be used by the software group to test their operating system, thus allowing the reactor's experiments to be performed using the old computer until the operating system was almost completely debugged. It was intended that the prototype system would remain in operation after the initial checkout expanding the capabilities of the existing PDP-15 system.

BIBLIOGRAPHY

Anderson, R. D. "Acqusition Proposal for a PDP-15/77A Computer to Replace the IBM 1401." Personal communication. Ames, Iowa: U.S.A.E.C., 1973.

Anderson, R. D. "Interface." Personal communication. Ames, Iowa: U.S.A.E.C., 1968.

Anderson, R. D.; Campbell, Jerry H.; Carter, Benjamin F.; Conley, Marsha K.; Helland, Barbara J.; Thomas, William D. "PDP-15/SDS 910 Systems Configuration." Personal communication. Ames, Iowa: U.S.A.E.C., September 27, 1973.

Digital Equipment Corp. PDP-15 Maintenance Manual. Maynard, Massachusetts: Digital Equipment Corp., 1970.

Digital Equipment Corp. PDP-15 Systems Interface Manual. Maynard, Massachusetts: Digital Equipment Corp., 1971.

Holland, Ed. "Minicomputer I/O and Peripherals." IEEE Computer Group News, 3(July/August, 1970), 10-14.

Kintner, P. M. "Interfacing a Control Computer with Control Devices." Control Engineering, 16(November, 1969), 97-101.

Korn, G. A. "Digital-Computer Interface Systems." Simulation, 11(December, 1968), 285-298.

Rinder, R. "I/O Architecture of Minicomputers." Datamation, 16(May, 1970), 119-124.

Scientific Data Systems. SDS-910 Computer Reference Manual. Santa Monica, California: Scientific Data Systems, 1963.

SDS Standard Module Data Sheets. Santa Monica, California: Scientific Data Systems, 1965.

Soucer, B. Minicomputers in Data Processing and Simulation. New York: John Wiley and Sons, 1972.

ACKNOWLEDGMENTS

I would first like to express my appreciation to my wife Jan and my son Jason for their many sacrifices and to my parents for their constant support and help in editing this thesis. I also wish to extend my appreciation to Dr. Terry Smay, my major professor, for his guidance and to Bill Thomas, Dale Anderson, Barb Helland, and Jerry Campbell for making this project possible. I would also like to thank John Erickson for his patience, support, and never ending smile in helping debug the interface.

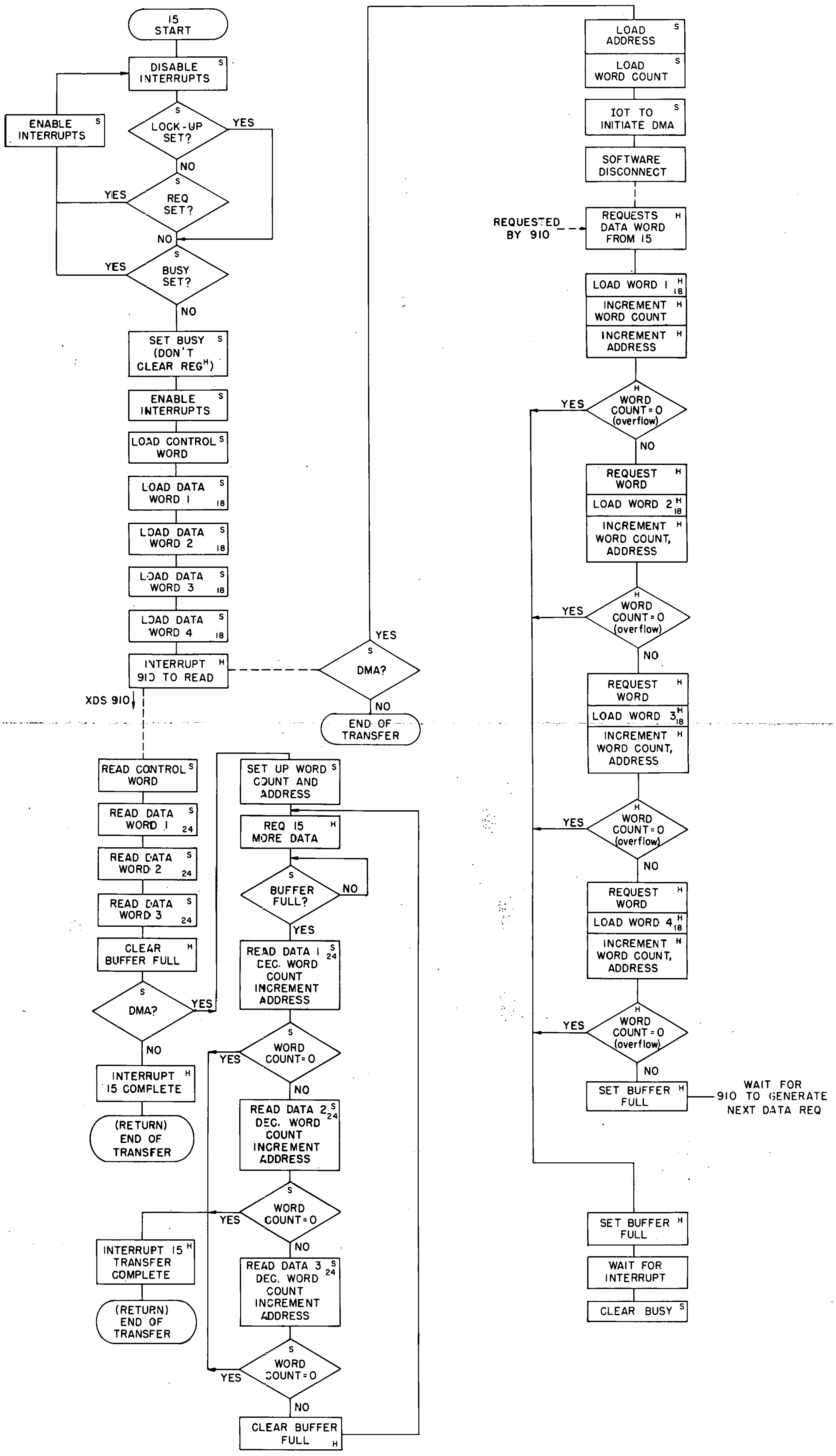
APPENDIX A: SDS-910 ADDRESS ASSIGNMENTS

FLIP FLOP(S)	Axy	Bz	INSTRUCTIONS	THAT APPLY	
CLEAR 910 INTERRUPT	30	0		POT	
BUFFER FULL	30	0			SKS
DATA WORD 1	30	1	PIN	POT	
DATA WORD 2	30	2	PIN	POT	
DATA WORD 3	30	3	PIN	POT	
REQUEST	30	4		POT	
BUSY	30	5		POT	SKS
15 INTERRUPT	30	5	PIN		
910 OK	30	6		POT	
CONTROL WORD	30	7	PIN	POT	

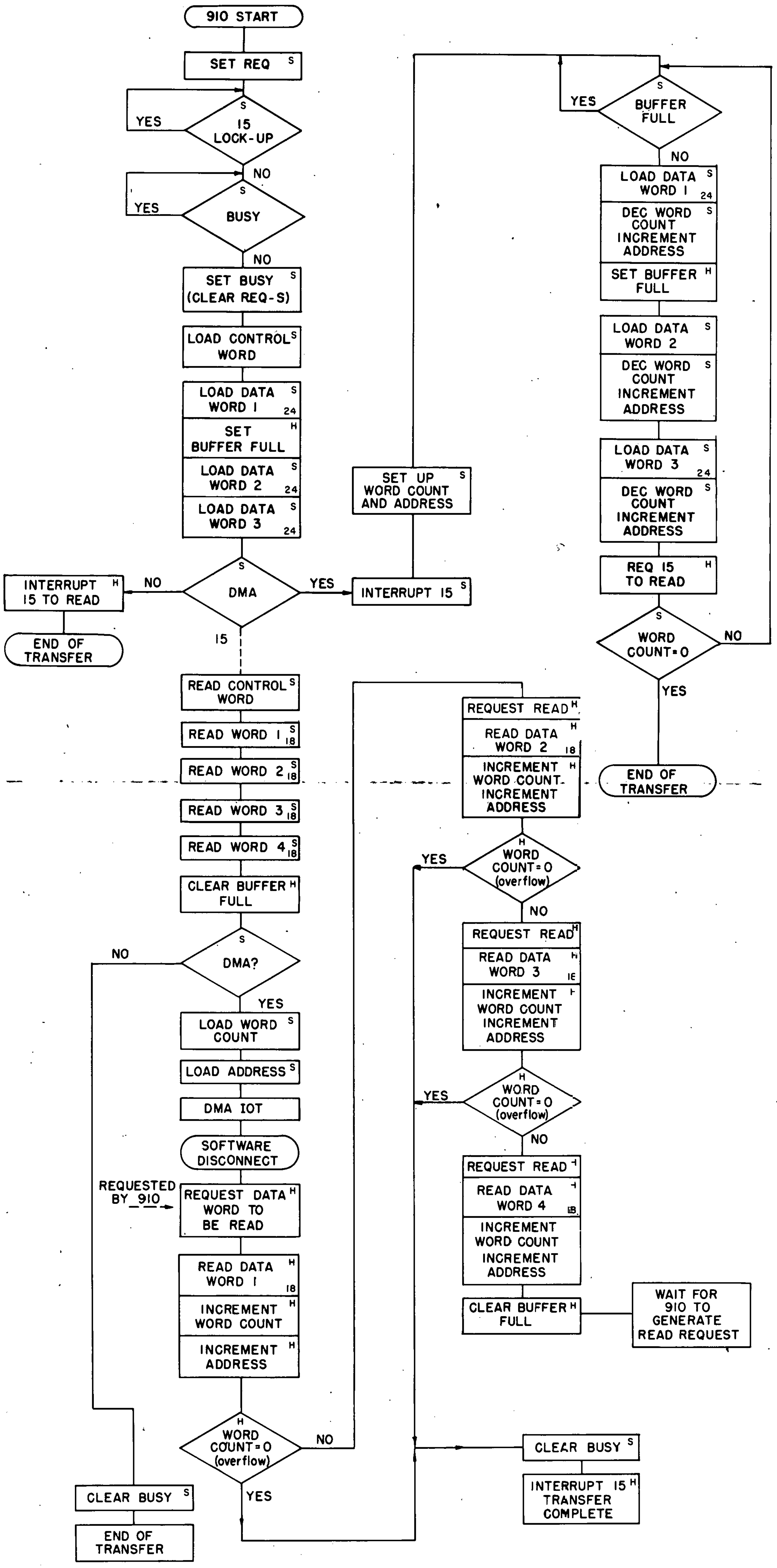
APPENDIX B: PDP-15 INTERFACE IOT INSTRUCTIONS

DATA WORD 1 READ	701412
DATA WORD 1 WRITE	701404
DATA WORD 2 READ	701432
DATA WORD 2 WRITE	701424
DATA WORD 3 READ	701452
DATA WORD 3 WRITE	701444
DATA WORD 4 READ	701472
DATA WORD 4 WRITE	701464
REQUEST TEST	701541
BUSY WRITE	701524
BUSY TEST AND WRITE	701525
15 OK WRITE	701544
CONTROL WORD READ	701572
CONTROL WORD WRITE	701564
CURRENT MEMORY ADDRESS POINTER WRITE	701604
WORD COUNT WRITE	701624
CLEAR INTERRUPT FLAG	701601
INITIATE DMA TRANSFER TO THE SDS-910	701664
INITIATE DMA TRANSFER TO THE PDP-15	701662

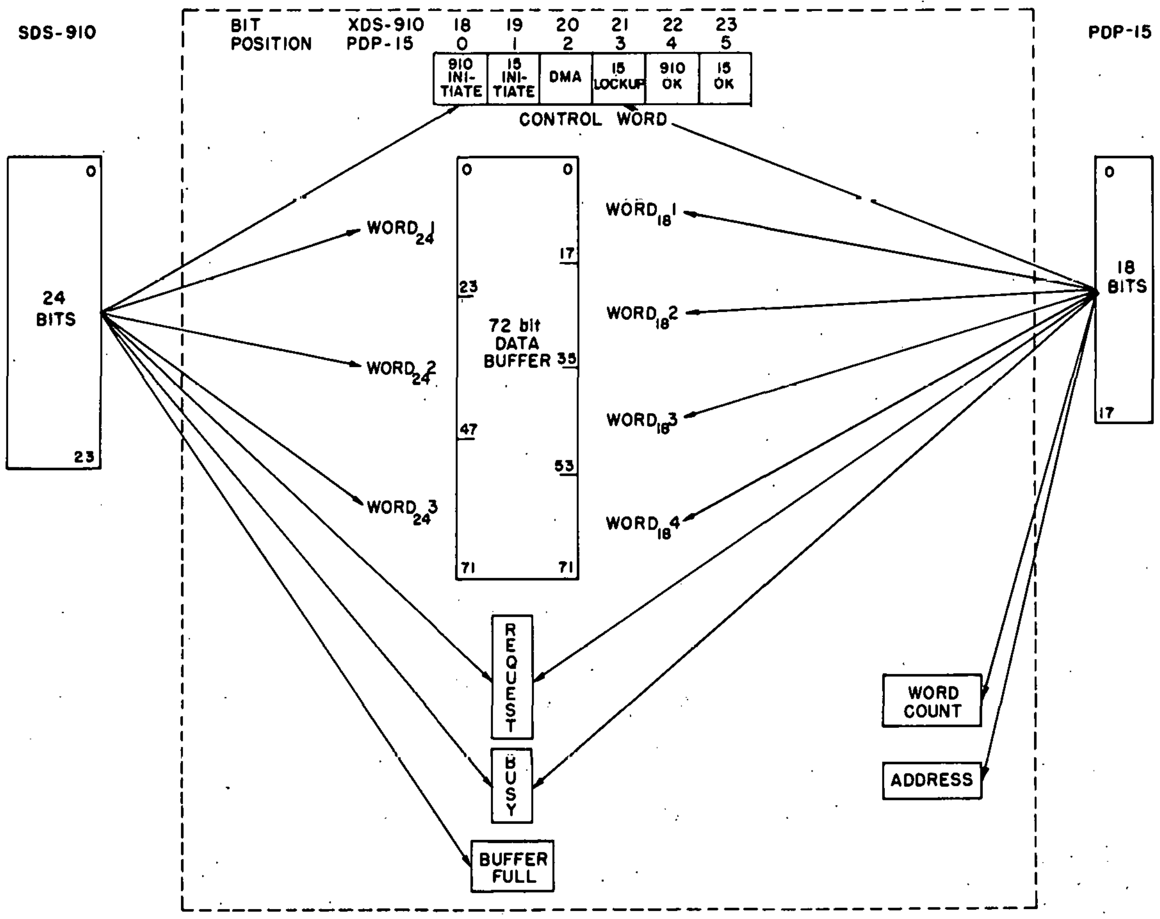
APPENDIX C: FLOWCHART OF TRANSFERS FROM THE PDP-15 TO THE SDS-910


15 START
DISABLE INTERRUPTS S
ENABLE INTERRUPTS S
LOCK-UP SET? S
YES
NO
REQ SET? S
YES
NO
BUSY SET? S
YES
NO
SET BUSY (DON'T CLEAR REG^H) S
ENABLE INTERRUPTS S
LOAD CONTROL WORD S
LOAD DATA WORD 1 S 18
LOAD DATA WORD 2 S 18
LOAD DATA WORD 3 S 18
LOAD DATA WORD 4 S 18
INTERRUPT 910 TO READ H
XDS 910
DMA? S
YES
NO
END OF TRANSFER
READ CONTROL WORD S
READ DATA WORD 1 S 24
READ DATA WORD 2 S 24
READ DATA WORD 3 S 24
CLEAR BUFFER FULL H
DMA? S
YES
NO
INTERRUPT 15 COMPLETE H
(RETURN) END OF TRANSFER
SET UP WORD COUNT AND ADDRESS S
REQ 15 MORE DATA H
BUFFER FULL? S
NO
YES
READ DATA 1 DEC. WORD COUNT INCREMENT ADDRESS S 24
WORD COUNT=0 S
YES
NO
READ DATA 2 DEC. WORD COUNT INCREMENT ADDRESS S 24
WORD COUNT=0 S
YES
NO
INTERRUPT 15 TRANSFER COMPLETE H
(RETURN) END OF TRANSFER
READ DATA 3 DEC. WORD COUNT INCREMENT ADDRESS S 24
WORD COUNT=0 S
YES
NO
CLEAR BUFFER FULL H
LOAD ADDRESS S
LOAD WORD COUNT S
IOT TO INITIATE DMA S
SOFTWARE DISCONNECT
REQUESTED BY 910
REQUESTS DATA WORD FROM 15 H
LOAD WORD 1 H 18
INCREMENT WORD COUNT H
INCREMENT ADDRESS H
WORD COUNT=0 (overflow) H
YES
NO
REQUEST WORD H
LOAD WORD 2 H 18
INCREMENT WORD COUNT, ADDRESS H
WORD COUNT=0 (overflow) H
YES
NO
REQUEST WORD H
LOAD WORD 3 H 18
INCREMENT WORD COUNT, ADDRESS H
WORD COUNT=0 (overflow) H
YES
NO
REQUEST WORD H
LOAD WORD 4 H 18
INCREMENT WORD COUNT, ADDRESS H
WORD COUNT=0 (overflow) H
YES
NO
SET BUFFER FULL H
WAIT FOR 910 TO GENERATE NEXT DATA REQ
SET BUFFER FULL H
WAIT FOR INTERRUPT
CLEAR BUSY S

APPENDIX D: FLOWCHART OF TRANSFERS FROM THE SDS-910 TO THE PDP-15


910 START
SET REQ S
15 LOCK-UP S
YES
NO
BUSY S
YES
NO
SET BUSY (CLEAR REQ-S) S
LOAD CONTROL WORD S
LOAD DATA WORD 1 S 24
SET BUFFER FULL H
LOAD DATA WORD 2 S 24
LOAD DATA WORD 3 S 24
DMA S
NO
INTERRUPT 15 TO READ H
END OF TRANSFER
YES
INTERRUPT 15 S
SET UP WORD COUNT AND ADDRESS S
BUFFER FULL S
YES
NO
LOAD DATA WORD 1 S 24
DEC WORD COUNT INCREMENT ADDRESS S
SET BUFFER FULL H
LOAD DATA WORD 2 S
DEC WORD COUNT INCREMENT ADDRESS S
LOAD DATA WORD 3 S 24
DEC WORD COUNT INCREMENT ADDRESS S
REQ 15 TO READ H
WORD COUNT=0 S
NO
YES
END OF TRANSFER
15
READ CONTROL WORD S
READ WORD 1 S 18
READ WORD 2 S 18
READ WORD 3 S 18
READ WORD 4 S 18
CLEAR BUFFER FULL H
DMA? S
NO
YES
LOAD WORD COUNT S
LOAD ADDRESS S
DMA IOT
SOFTWARE DISCONNECT
REQUESTED BY 910
REQUEST DATA WORD TO BE READ H
READ DATA WORD 1 H 18
INCREMENT WORD COUNT H
INCREMENT ADDRESS H
WORD COUNT=0 (overflow) H
NO
YES
CLEAR BUSY S
END OF TRANSFER
REQUEST READ H
READ DATA WORD 2 H 18
INCREMENT WORD COUNT INCREMENT ADDRESS H
WORD COUNT=0 (overflow) H
YES
NO
REQUEST READ H
READ DATA WORD 3 H 18
INCREMENT WORD COUNT INCREMENT ADDRESS H
WORD COUNT=0 (overflow) H
YES
NO
REQUEST READ H
READ DATA WORD 4 H 18
INCREMENT WORD COUNT INCREMENT ADDRESS
CLEAR BUFFER FULL H
WAIT FOR 910 TO GENERATE READ REQUEST
CLEAR BUSY S
INTERRUPT 15 TRANSFER COMPLETE H

APPENDIX E: INTERFACE BLOCK DIAGRAM


SDS-910
PDP-15
BIT POSITION
XDS-910 18 19 20 21 22 23
PDP-15 0 1 2 3 4 5
910 INI-TIATE
15 INI-TIATE
DMA
15 LOCKUP
910 OK
15 OK
CONTROL WORD
24 BITS
0
23
18 BITS
0
17
WORD$_{24}$1
WORD$_{24}$2
WORD$_{24}$3
72 bit DATA BUFFER
0 23 47 71
0 17 35 53 71
WORD$_{18}$1
WORD$_{18}$2
WORD$_{18}$3
WORD$_{18}$4
REQUEST
BUSY
BUFFER FULL
WORD COUNT
ADDRESS