OS910

THE SDS-910 OPERATING SYSTEM

MASTER

Anthony P. Lucido and Leonard C. Moon

AMES LABORATORY, USERDA
IOWA STATE UNIVERSITY
AMES, IOWA

Date Transmitted: January 1975

PREPARED FOR THE U. S. ENERGY RESEARCH AND DEVELOPMENT ADMINISTRATION
UNDER CONTRACT W-7405-eng-82

RESTRICTION OF THIS DOCUMENT UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

OS910

THE SDS-910 OPERATING SYSTEM

Anthony P. Lucido and Leonard C. Moon

Ames Laboratory, ERDA
Iowa State University
Ames, Iowa 50010

Date Transmitted: January 1975

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

MASTER

PREPARED FOR THE U. S. ENERGY RESEARCH AND
DEVELOPMENT ADMINISTRATION UNDER CONTRACT W-7405-eng-82

—NOTICE—

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

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

Price: Microfiche \$2.25
 \$5.45

TABLE OF CONTENTS

	PAGE
ABSTRACT	v
I. INTRODUCTION	1
II. COMMUNICATIONS VECTOR TABLE	5
III. TRANSIENT STORAGE AREAS	8
IV. COLD START	9
V. DISPATCHER	10
VI. CLOCKS	16
VII. TASK SCHEDULING	18
VIII. I/O SCHEDULING	19
IX. REAL-TIME ERROR MANAGEMENT	23
X. TRANSIENT STORAGE AREA MANAGEMENT	25
XI. INTERFACE COMMUNICATION PROTOCOL	27
XII. SDS 910/PDP-15 INTERFACE PROTOCOL	32
XIII. SDS 910 TO PDP-15 TRANSMISSION ROUTINES	35
XIV. PDP-15 TO SDS 910 READ ROUTINE	36
XV. FLOATING POINT POPS	38
TABLE 1. POPS AND THEIR LOCATIONS	39
APPENDIX A. SYSTEM FLOWCHARTS	40
APPENDIX B. PRIORITY SCHEDULING	61
APPENDIX C. ROUND-ROBIN SCHEDULING	65

TABLE OF CONTENTS (CONTINUED)

	PAGE
APPENDIX D. USER SECTION	67
DEC PDP-15 TO SDS 910 COMMUNICATION	74
APPENDIX E. USE OF THE INITIAL LOADER	78
APPENDIX F. HOW TO SET-UP A DATA SET ON THE PDP-15 DISK TO BE LOADED ACROSS THE INTERFACE AND INTO THE SDS 910	80

ABSTRACT

The OS910 operating system for the SDS-910 computer is discussed in detail. The operating systems communications with the PDP-15 computer are enumerated. Examples are given on how user programs are to communicate with the operating system and their experiment.

Distribution List

Chicago Patent Office	1
TIC	27
Ames Laboratory Library	9
A. P. Lucido	12
L. C. Moon	12
	<hr/>
Total	61

I. Introduction:

OS910, the operating system for experiment control using the SDS 910 computer, has been designed to do task management and experiment control in an interrupt-driven environment. The Ames Laboratory Research Reactor has a multitude of experiments on it that need real-time data collection and control facilities.

The computer system that is designed to perform the requisite task management and experiment control consists of an SDS 910, used as an "intelligent interrupt handler", and DEC PDP-15 used as the computational element, with the two computers linked by an interface designed at the Ames Laboratory. Both computers have Ames Laboratory designed operating systems.

This report will detail the operating system for the SDS 910 computer, OS910. Facilities provided by OS910 include:

- a) SDS 910/PDP-15 communications,
- b) task scheduling,
- c) scheduling of tasks on clock queues,
- d) fielding of real-time error interrupts.

Also, great flexibility is allowed the user in managing experiment communication and experiment derived interrupts.

OS910 is based on a round-robin task scheduling scheme. Because of the need for flexibility, and the desire for minimum operating system intervention with a task, experiment interrupts are managed directly by the user. All SDS 910 - PDP 15 communication is handled by OS910 routines,

as well as the four clocks (12.8 msec. fast clock, 102.4 msec. medium clock, 1.6384 sec. slow clock, and 1.6384 sec. real-time clock), and real-time error interrupts. The latter may be generated by either the user or the operating system, and are, for example, symptomatic of an invalid experiment address. Treatment of real-time (RT) errors depends on whether the operating system caused the error, resulting in a halt with the interrupt system disabled, or a user caused the error, resulting in termination of that task.

Under OS910, tasks may not be transient during execution. The code for a task must reside completely in storage, and may not be moved around. However, for control purposes, a set task/program format is imposed. A users program has the form of:

- a) ID/KEY table entry for the user (to be explained later),
- b) the sequence of tasks for the user.

Both to maintain system integrity, and allow RT error management, the user must, upon entering an interrupt handling routine,

- a) save the A,B, and X registers,
- b) save locations zero and one of the machine (control areas),
- c) save the current CCBP (Current Control Block Pointer) value,
- d) set CCBP to the users ID (as assigned for use on the PDP-15).

Upon relinquishing control, the user must restore the above registers and storage areas. These actions may be accomplished by using two POP's in the system, PSAV and PRST. The POPs are used as follows:

```

a) STA      savearea
   LDA      0
   PSAV     savearea
   LDA      userid
   STA      CCBP

b) PRST     savearea

```

(Both POPs will be explained more fully, later.) Thus, a linked environment situation is maintained, and the operating system can always determine the current system user.

As a key is used with READs/WRITEs in the PDP-15, and (it is assumed that) the user program is set up to accept one data length for a specific key, a table of the keys and their related buffer areas may be maintained. Because of this, when the PDP-15 transmits data to the SDS 910, a predefined buffer area already exists in the SDS 910 to accept it. Dynamic buffering must exist for SDS 910 to PDP-15 transmissions, as the user will not know when the data are finally sent, nor, when the buffer is empty. OS910 supplies the user with POP's that take care of the buffering problem.

To provide this dynamic buffering, a transient buffer storage area is system maintained. The transient buffer storage area contains nodes of seven words in length. The format of a buffer will be shown later.

Also included are POPs to perform queuing of tasks. A task may be placed on the task queue, or one of the three clocks (fast, medium, or

slow). Also, a parameter list may be referenced in this call. The format of a node on one of these queues is:

Here, the next node pointer is a simple forward reference, to create a linked list of nodes. The ID value (in bits 0-8 of the second word) is the user ID of the task to be invoked. Bit 9 of word two is set to zero, with the entry point address in bits 10-23. If there aren't any parameters, the third word is set to minus one, else it contains the address of the parameter list.

A transient storage area is maintained for queue nodes. Currently, there are 128 nodes of three (3) words each for queue nodes. If necessary, this area can easily be expanded or contracted to accommodate any run-time needs.

II. Communications Vector Table (CVT):

The CVT is really the heart of OS910. Locations 2 through 330 (octal) are encompassed. Included are constants, an interrupt idler routine, POPs, interrupt vectors, and the table of entry points. We will now explain the different CVT entries.

The interrupt idler routine entry point, DUMMY, is at location 2, octal. When a regular interrupt entry uses a BRM DUMMY, then the interrupt level is cleared, and a return to the interrupted program results.

The real-time clock counter, TIME, is at location 4, octal. This counter may be directly referenced by a user. The value at TIME will not be allowed to exceed 400000, octal. Thus, an 18 bit word will always be able to hold the counter value, in its entirety.

The next four locations contain entry point address to the allocation/deallocation routines. Each routine is invoked via an indirect branch and mark (BRM*). To allocate a queue node, GETNDE is used. To get a buffer node, GETBUF is used. To free up either a queue node or buffer node, use FRENDE or FREBUF, as required. The servicing routines, themselves, will be explained later.

At locations 11 and 12, octal, are the free storage list addresses. QNDEFQ references the free storage list for queue nodes, and BNDEFQ references the free storage list for buffer nodes.

Next come six numeric constants that may be used by the programmer, and are heavily relied upon by the system. These constants are in

locations 13 to 20, octal. In order, the constants are -1, 0, 1, 2, 3, and 4. These may be symbolically referenced by MINUS1, ZERO, ONE, TWO, THREE, and FOUR, respectively. Also, at octal location 21, is ADRMSK, which is an address mask. When used as the operand of an extract (ETR), ADRMSK will set all but bits 10-23 to zero.

Locations 22 through 23, octal, contain pointers to the dispatcher (TASKER), and RAID (RAID), respectively. The general user will use these pointers.

Octal locations 24 through 26 contain C510, C210, and CCBP. C510 is the system ID (identifier). Each task in the PDP-15 is assigned a unique user ID. The system assumes control of ID 510. To signify which user currently has control, the variable CCBP is kept updated. To reference the main entry point of a task, there is the Entry Point table, starting at location 300 octal. Mapping the user/system ID's into a unique location between 300 and 330 octal, becomes easy; just subtract C210 (octal value 210) from the current ID value. CCBP contains the ID of the task that is currently in control of the computer.

BIT9, at octal location 27, and IDMSK1, at octal location 34, are masks used by the system. BIT9 is used to test/set bit location 9 of a SDS 910 word. IDMSK1 is used to extract bits 0-8 of a SDS 910 word.

Location 35, octal, starts the save POP (PSAV). This POP, which runs in a non-interruptable state (priority mode), will save the contents of the A, B and X registers, locations 0 and 1, and CCBP. As location zero is destroyed by the invocation of a POP, the contents of the A

register are first saved in the six word save area, and then the contents of location zero are loaded into the A register.

Octal location 53 is the entry point of the restore POP (PRST). This POP will reload the contents of locations 0 and 1, CCBP, and the A, B, and X registers from the referenced save area. As for PSAV, PRST operates in priority mode.

Location 73, octal, is the address of the SETOVF POP, which sets the overflow bit. RSTOVF at octal location 74 will reset the overflow bit. Note that a dummy operand must be used with these POPs, even though none is used in the POP, itself.

Location 100 octal begins the POP invocation area. Included are all system and user oriented POPs. This table will change as user/system requirements change.

Location 200, octal, begins the interrupt vector area. The interrupt locations should be obtained from the listing of the operating system code. However, location 200, octal, is the real-time clock special interrupt, and octal location 201 is the RT error interrupt location.

Locations 300 to 330 octal comprise the Entry Point table, which contains the entry point address value for each user/system task. Initially set to -1 (77777777, octal), each location is changed when the user/system code is brought into storage. Unless C210 is suitably modified, moving this table will provide catastrophic results.

III. Transient Storage Areas:

Octal location 342 begins the transient storage area for the system at the reactor. Each area currently contains 128 nodes. The free queue node uses three words per entry, and extends from locations 342 through 1140. The buffer node free storage area extends from location 1141 through 2740, and give 128 seven word nodes. Each list is forward linked, with the last nodes' link field set to minus one.

It should be noted that changing the amount of free storage area is very easy, but an appropriate change to the cold start routine (COLD) must be made to allow for the change in number of entries.

IV. Cold Start:

The Cold Start routine will initialize the free storage areas, set the user interrupts to the idler state (MIN DATA, MDE DATA for special interrupt pairs, and BRM DUMMY, for regular interrupts), set system-oriented interrupts to the required values, set locations 300 and 330 to the desired values (point to the system), then enable the interrupts and branch to the dispatcher (TASKER).

If the transient storage area locations or sizes are modified, the appropriate cold start routine locations must also be modified. Also, if the Entry Point table locations are changed, the cold start routine must again be changed.

V. Dispatcher:

We now come to the heart of OS910, the dispatcher (TASKER). Three queues are managed by the dispatcher, the interrupt and regular write queues, and the task queue. The interrupt write queue has the highest priority, with the regular send (write) queue next in priority, leaving the task queue lowest in priority. Queuing of tasks will be covered later.

As noted earlier, the format of a queue node is:

address of next node		
ID	0	entry point loc.
parameter list loc.		

(if this is the last node on the queue, the "address of next node" field is set to -1)

For the write (send) queues, a buffer node or buffer node list is expected to be referenced by the parameter list field. The entry point field is assumed to be the routine to perform the I/O, either SEND72 (for a 72 bit send), or DMASND (for a DMA transfer). The ID field is expected to be that of the user initialing the transfer.

With all tasks, if the user has been set inactive, then the task is not dispatched. This goes for any of the I/O tasks, and task queue entries. To determine activity, a bit is kept in a preamble to each users' code. The format of a users task is:

ID/Key table section
users programs

The ID/Key table is of the form:

	0	9	10	23
word: 1	ID	a/i	# key entries	
2	entry point of the shutdown routine for this user			
3	Key 0 Buffer loc.			
4	Key 0 Entry point loc.			
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
(2n+2)	Key n Buffer loc.			
(2n+3)	Key n Entry Point loc.			

where:

ID - the users ID.

a/i - the active/inactive bit (a/i=1: inactive state indication).

key entries - the number of key entries for this user. The highest key to be used will then be (# key entries) -1.

Entry point of the shutdown routine for this user - each user has a routine to effect the deactivation of interrupts and other housekeeping facilities. This routine, which must be supplied by the user, is activated by either receiving a shutdown code from the PDP-15 (a CLOSE generates this), or encountering a RT error for the user.

KEY i Buffer loc. - the location of a buffer area to hold data sent from the PDP-15 by a write using Key i.

KEY i Entry point loc. - as a user will do processing on data sent from the PDP-15, the ability to schedule a task associated with each buffer area is provided. The referenced entry point may be either a (user supplied) dummy routine, or some piece of useful code. In any case, the field must contain a legitimate entry point address, as the system will create a task queue entry using the entry point loc. value, and the buffer loc. value.

A buffer area will have to be large enough to contain the maximum amount of data sent, plus whatever control information kept (see the I/O routines).

When information is sent from the SDS 910 to the PDP-15, it is placed in a dynamically acquired buffer area, and a queue entry is made. The queue entry is of the form:

link field		
ID	0	service routine address
address of buffer		

If this is a 72 bit transmission, then the service routine address is to the SEND72 routine. Otherwise, it is to the DMASND routine. The format of a dynamic buffer is:

a) for a 72 bit transfer:

word 1	-1 (grounded)
2	72 bits of data
3	
4	
5	72 bits of unspecified data
6	
7	

b) for a DMA transfer:

1. First buffer node;

	0	1	8	9	10	23
word 1	e	# transfers	0	Link		
2	72 bits of data to set up DMA transfer					
3						
4						
5	First 72 bits of DMA data					
6						
7						

where:

e - extent bit. If this is the last node on the list, then e=1, else e=0.

transfers - this is the number of 72 bit transfers to be done, excluding the 72 bits of set-up data.

Link - this is the address of the next buffer node, or undefined if this is the last node on the list.

2. Intermediate nodes (full):

		0	10	23
word	1	0	0	Link
	2	72 bits of data		
	3			
	4			
	5	72 bits of data		
	6			
	7			

where: link is the address of the next buffer node.

3. Final node:

	0	10	23
Word 1	1	0	undefined
2	72 bits of data		
3			
4			
5	either 72 bits of data, or undefined		
6			
7			

Note that bit 0 (the extent bit) is set to one (1).

Each queue has the same form, and each is treated the same. A two word queue pointer set exists for each queue, consisting of a head and tail pointer. If the queue is not empty (head, tail $\neq$ -1), then the first node on the queue is tested for being from an active task. If the task is active, then the task is dispatched. Otherwise, the task is cleared off. This latter operation amounts to releasing all buffer space, for an I/O queue entry, and freeing the queue node.

One curious point needs to be made. When a task returns, and the node space has been released, the interrupts, having been disabled before this operation, are again enabled, and given time to hit. To allow the interrupts to hit, it is not sufficient to do NOP's because a NOP is an uninterruptable instruction. In the system we perform a

STA AVAIL

which will not cause the overflow indicator to be affected, but will allow interrupts to hit.

To activate a task, the following steps are taken:

- a) the entry point address is extracted from the second word of the queue node;
- b) the A register is loaded with the parameter list address field contents;
- c) the interrupt system is enabled (all previous work was done with the interrupts disabled);
- d) a BRM is done relative to the entry point pointer.

Task deactivation is very simple. For an I/O queue node, the buffer list is assumed to be already freed. Otherwise, the A register is set to the address of the node. Then, the instruction:

BRM* FRENDE

is executed. This routine will chain the queue node back onto the queue node free storage list, and return.

The routine ACTIVE tests the a/i bit of the indicated task. On entrance, the X register must point to the node to be scheduled. The ID value is then extracted. If the ID value is 510_8 then the task is forced to be "active". Otherwise, the associated ID/Key tables a/i bit is checked. If set to 1, then the return address is incremented before returning. If $a/i = 0$, then the normal return is performed.

VI. Clocks:

There are four clock signals which are supplied by the interface. First there is the real-time clock. This clock, with a 1.6384 second period, may be used to determine how long a program has been running.

The three other clocks are used for time-dependent task scheduling. Tasks may request scheduling on the queue for one (or more) of the clocks. The task chain looks and is used in the same way as for the dispatcher. Also, tasks are subject to the same tests and conditions.

The three clocks have periods of 12.8 ms (fast clock), 102.4 ms (medium clock), and 1.6384 second (slow clock), and are normally employed to provide a set time delay between operations in the SDS 910. Shaft movement, character transmission, etc., are some areas where one might employ the clocks.

A user may schedule a task onto one of the three clock queues by using one of three POPs. When a task is on a clock queue, it is executed the next time that clock "ticks". The action taken when a clock "ticks" is:

1. If the queue for that clock is empty, then the interrupt level is cleared and execution resumes where it was interrupted.
2. If there are nodes present, then the contents of the head pointer for the respective clock queue are copied to the area referenced by CLKxQ, for x=A,B, or C (fast, medium, or slow clocks), and the queues' head/tail pointer are both set to -1. Then,
3. Each task on the queue now referenced by CLKxQ is, if active, invoked in turn, then deleted from the queue. Meanwhile, new tasks may be scheduled on this or any other queue.

This task dispatching operates with the interrupt system enabled, thus allowing higher priority interrupts to hit at any time, but stopping the

same level interrupt, or any lower one. When all the nodes on the queue (referenced by CLKxQ) have been treated, the interrupt level is cleared and execution resumes where the clock "tick" interrupted.

The one exception to the normal clock servicing routine is for the medium clock manager (clock B). This routine is used to check the real-time clock counter (TIME). If the current value of the clock is greater than octal 377777, the value is reset to zero.

VII. Task Scheduling:

Using OS910, the facility to cause the scheduling of tasks exists. A user may schedule a normal task (managed by the dispatcher), or a clock task (managed by the respective clock queue manager).

There are four POP's used for scheduling, SKDFST, SKDMED, SKDSLO, and SKDTSK. The parameters to each of these POPs is the same:

1. The A register contains the entry point argument,
2. The B register contains the parameter list pointer,
3. CCBP (at octal location 27) supplies the users ID.

Using these parameters, the routine MAKNDE constructs a queue node and sets the X register to the address of the newly constructed node.

The MAKNDE routine employs the A and B register, and CCBP, to fill in the node fields. An attempt is first made to allocate a queue node from the free storage list. If the list was empty at the time of the request, the X register will have been set to -1 (77777777, octal), and the MAKNDE routine will halt. Otherwise, the link field (first word) of the newly allocated node is set to -1 (to "ground" it); word two is set to the pair (user id, entry point address) arranged as:

ID - bits 0-8
Entry Point address - 10-23
Bit 9 is set to zero

The third word is set to the contents of the B register. When MAKNDE returns to the calling routine, the node may be added to the required queue list. Normally, the addition will be to the tail of the queue.

VIII. I/O Scheduling:

There are a set of four POPs to queue up data transmission to the PDP-15. Three POPs, SNED1, SEND2, and SENDMA, are used to send data back on a read return (see communication modes Section XI). The fourth, SNDINT, effects the transmission of an ONCODE resulting from an experiment interrupt (see the ALECS manual).

Each POP will construct a queue node, and as many buffer nodes as required for the amount of data. For all but SENDMA, this will be one buffer node. In any case, if a node (buffer on queue) is required, and none are available, the POP will execute a halt instruction. This condition, similarly raised in the task scheduling POPs, is indicative of a condition of too few nodes on that free storage list. Thus, the system programmer response must be to increase the transient storage area for that type node.

The SEND1 POP requires an operand to be present. The data referenced by the operand must be the three words:

Word 1: Length of transfer
 Word 2: "KEY Loc. in CAL" value
 Word 3: Data value

The "Key Loc. in CAL" value is passed by the PDP-15 on the initial set-up. This value will be put into your desired location by the system. The above data, along with the current CCBP value will be used to construct a buffer node. Then, a queue node will be constructed using the address of the buffer node (whose first word has been set to -1) as the parameter list field value, and the entry point to the I/O routine SEND72 and the CCBP value as the entry point/ID field values. The link word of the node will be set to -1, and the node chained onto the tail of the send queue.

SEND2 is similar to SEND1 except that the area referenced by the POPs' operand is the four words of:

Word 1: Length of transfer
 Word 2: "Key loc. in CAL" value
 Word 3: Data value
 Word 4: Data value

The same send queue node format is used, and the buffer is constructed in the same fashion.

SENDMA will construct a (possible) list of buffer nodes. The length of a buffer node is set at seven (7) 24 bit words. This allows for a link word (word 1), and two 72 bit data sections (words 2-4, and 5-7). This not only makes filling the buffer area easier, but also makes the transmission of the data easier, as boundary problems won't exist (the SDS 910 must, because of the interface structure, transmit data in groups of 72 bits).

The operand of the SENDMA POP references an area arranged as:

data entries
 "Key loc. in CAL" value
 data value
 .
 .
 .
 data value

The # data entries field gives the number of data entries to be transmitted back. The user has the responsibility of correctly setting this area.

The buffer nodes are set up as:

1. First buffer node:

	0	1	9	10	23
Word 1	e	# words	0	Link	
2	72 bits of initial data to start the DMA				
3					
4					
5	first 72 bits of DATA				
6					
7					

Note: a) if this is the only buffer node on the list, e=1; otherwise, e=0.

b) the # words field gives the number of 24 bit words to be transmitted in the DMA transfer, not counting the first 72 bit setup transmission that prepares the PDP-15 for the transfer. Zeroes are added onto the right of the data as required, to pad out to 72 bits in a transmission. Thus, if a DMA transmission were to be for only 54 bits of data, 18 bits, all zero, would be appended onto the right of the actual data to get to a 72 bit transmission.

Other than the possibility of having a list of buffers, and that the service routine will be DMASND rather than SEND72, the DMA node creation is the same as the SEND1 and SEND2 POPs.

In all three cases, SEND1, SEND2, and SENDMA, the data words are compressed, using a subroutine, from 24 bit format into the 18 bit format required by the PDP-15. Thus, it takes four (4) words from the SDS 910 (24 bits each), to compress down to the 72 bits of data.

The fourth POP in this series will construct an interrupt send queue (INTQUE) entry, and an associated buffer entry. This POP, SNDINT, assumes that the A register contains the ONCODE to be returned to the PDP-15. A dummy operand value must be present on the POP, even though it isn't used. The queue nodes entry point field is set to SEND72, the ID field is set to the current CCBP value, and the parameter list field points to a buffer node. The buffer node contains -1 as the first word entry. Then, the 72 bit field includes the ONCODE value.

IX. Real-Time Error Management:

A real-time (RT) error is caused by an I/O operation, using the interface to either the PDP-15 or an experiment, that incurs an addressing error.

There are two possible results from the RT error routine (RTERR).

If the RT error was incurred by a system routine, then a halt, with the interrupt system disabled, is performed. If a user task caused the error, then:

1. Create a task queue entry, placed at the head of the queue, that has the ID field being set to 510 (system ID), the entry point field being that of the shutdown execution routine (SHTDWN), and the parameter field being the entry point to the users task (the value at (CCBP-C210));
2. Create an interrupt queue entry with the ONCODE set to zero. This node also goes at the head of the queue.

The above two actions will schedule the shutdown of a task.

It should be noted that:

1. The queuing priority is (from highest to lowest): interrupt queue, send queue, and task queue;
2. To guard against scheduling problems, it is better to queue up the shutdown action than to do the work directly;
3. We must send the RT error generated ONCODE before the routine is shutdown; otherwise, the a/i bit = 1, and the send will be ignored.

Because of hardware infirmities, the above actions should be adhered to, as a hardware/software resource deadlock condition is otherwise likely.

The above constraints appear to guard against this eventuality.

Along with the above queuing of information, the RT error routine clears the interrupt level and returns to the interrupted program. As the interrupted program, which incurred the RT error, will have no global side effects, it appears to be safe to let it complete on its own, rather than trying to shut the user down immediately and return to the task previous to the user causing the RT error. This return is effected by executing the instruction: BRU* RTERR , where RTERR is the entry point name of the RT error interrupt servicing routine.

If a user generates a second RT error the system will again go to the RT error handling routine and proceed through the above procedure. Eventually, the PDP-15 will send a shutdown for the user which will cause the SDS-910 system to shutdown this user.

X. Transient Storage Area Management:

There are two transient storage areas in OS910. These provide nodes for queuing and buffering. Queue nodes are three words long, buffer nodes are seven. Each transient storage area is arranged as a simple front linked list, as:

Note that the value -1 (octal 77777777) is used to denote a "grounded" node; i.e., the last node on a list.

There are two routines that directly affect the free storage list (transient storage area). The routine ALLOC will allocate a node from the free storage list referenced by the X register. On return, the X register will either be set to -1 (if the list was empty) or reference a node from the free storage list.

The FREE routine will return the node referenced by the A register to the free storage list referenced by the X register. No modifications are made to the registers.

Neither ALLOC nor FREE is used directly. To allocate a queue or

buffer node, the user employs either GETNDE or GETBUF, respectively.

On returning, these routines have the X register set to -1, if the required free storage list is empty, or to the address of a node, if there is at least one node left.

To free a node (return it to the required free storage list), the user sets the A register to the address of the surplus node, and executes either DALNDE or DALBUF to deallocate a queue or buffer node, respectively.

XI. Interface Communication Protocol:

The SDS 910/PDP-15 interface protocol developed by D. Anderson, J. Campbell, B. Carter, W. Thomas, B. Helland, and M. Conley was to establish the most feasible communications mode possible between the two computers. The SDS 910 uses a 24 bit word and the DEC PDP-15 uses an 18 bit word. Thus, to minimize waste and for other hardware consideration, a 72 bit transfer was decided upon. The following description of the types of transfers were detailed by B. Helland. They are in terms of the 18 bit word format, i.e., using four words to create a block of data.

Each transfer, whether it is a DMA transfer or not, starts with some status and identifying information. Bits 0-8 of the first word constitute the status portion. The explanation of this part is:

1. On a DEC PDP-15 to SDS 910 transfer:

bit: 0 = 1 : shutdown indication
 1 = 1 : read operation
 2 = 1 : write operation
 3 = 1 : DMA transfer-only in case of write
 4 = 1 : 2 word write (bit 2 = 1)
 5 = 1 : 1 word write (bit 2 = 1)
 6 = 1 : system task send

2. On a SDS 910 to DEC PDP-15 transfer:

bit: 0 = 1 : interrupt
 1 = 1 : four word read from SDS 910
 2 = 1 : DMA read from SDS 910
 3 = 1 : RT error, only if bit 0 = 1
 4 = 1 : if bit 3 = 1, then this indicates software problem
 5 = 1 : system task response
 6 : unused

Also, bits 7,8 of the first word are currently unused.

Bits 9-17 give the ID of the user/system doing the I/O operation. The ID value, in the range of 510-540 (octal) is a unique code assigned to a user when he first comes on line.

The buffer forms for the different I/O operations are:

1. On a transmission from the PDP-15 to the SDS 910:

a) Write

i. One word

word 1: bits: 0-5: 001001
6-8: unused
9-17: ID value

word 2: Key value

word 3: Data value

word 4: Unspecified

ii. Two word:

word 1: bits: 0-5: 001010
6-8: unused
9-17: ID value

word 2: Key value

word 3: Data value

word 4: Data value

iii. DMA transfer:

word 1: bits: 0-5: 001100
6-8: unused
9-17: ID value

word 2: Key value

word 3: Length of the DMA transfer

word 4: Unused

Then, the data will follow.

iv. SHUTDOWN:

word 1: bits: 0-5: 100000
 6-8: unused
 9-17: ID value

words 2-4: Unspecified

b) Read:

A read is generated by a read of an experiment file from the PDP-15. It generates a read return transmission that asks the SDS 910 to transmit data back to the PDP-15. The transfer form is:

word 1: bits: 0-5: 010000
 6-8: unused
 9-17: ID value

word 2: Key value

word 3: Length value (in terms of 18 bit words).

word 4: "key location in CAL" value. This field is used for a check on returning to the PDP-15.

This read request is for a one or two word transfer, or a DMA transfer.

2. On a transmission from the SDS 910 to the PDP-15:

a) Read Return:

i. One word of data:

word 1: bits: 0-5: 010000
 6-8: unused
 9-17: ID value

word 2: Address of key location in CAL

word 3: Data value

word 4: Unspecified

ii. Two words of data:

word 1: bits: 0-5: 010000
 6-8: unused
 9-17: ID value

word 2: Address of key location in CAL

word 3: Data value

word 4: Data value

iii. DMA transfer:

word 1: bits: 0-5: 001000
 6-8: unused
 9-17: ID value

word 2: Address of key location in CAL

words 3, 4: Unspecified

b) Interrupt transmission:

This results from an interrupt generating an interrupt signal containing data required by the PDP-15, or from incurring a real-time error (in which case the ONCODE is forced to be zero).

word 1: bits: 0-5: 100000
 6-8: unused
 9-17: ID value

words 2, 3: Unspecified

word 4: ONCODE value

The layout of this first transfer is, in detail, and in terms of the SDS 910 word format:

0	8 9	17 18	23
STATUS	ID value	first part of key	

0	11 12	23
rest of key	upper part of value in a 1 or 2 word write, or length in a read or DMA	

0	5 6	23
rest of value or length	addr. at key loc. in CAL, second data value, or ONCODE	

XII. SDS 910/PDP-15 Interface Protocol:

This section will discuss the machine interface from the point of view of the SDS-910. The PDP-15 side is discussed in another manual. The task of the interface is to allow the exchange of data between the two computers. It does so by allowing 72 bit at a time transfers. This was determined to be the most efficient method of transfer for many reasons. For a more complete discussion see B. Carter's masters thesis.

Both computers have access to certain registers and indicators of the interface. To communicate with the interface, and allow data transmission, a SDS-910 user employs the three instruction set combinations of:

1. To Read from a register, buffer, etc.:

EOM	30(A _{XY} B _Z)
PIN	loc.

2. To Write to a register, buffer, etc.:

EOM	32(A _{XY} B _Z)
POT	loc.

3. To test an indicator:

EOM	36(A _{XY} B _Z)
SKS	300(mask)

The A_{XY}B_Z value is an experiment/machine interface address. The mask is a six bit value. It is set to test the control register, buffer full, etc. The following table gives the indicator addresses, functions, and what operations are allowed.

indicator	A _{XY}	B _Z	C* bit	Operations allowed
Buffer full	30	0		SKS
15 interrupt	30	0		PIN
Data word 1	30	1		PIN,POT
Data word 2	30	2		PIN,POT
Data word 3	30	3		PIN,POT
Request	30	4	23	POT
Busy	30	5	23	POT,SKS
910 OK	30	6		POT
15 lock up	30	7	21	PIN,SKS
DMA	30	7	20	PIN,POT
15 initiate	30	7	19	PIN
910 initiate	30	7	18	PIN,POT
15 OK	30	7	23	PIN
Clear outboard				
910 interrupt	30	0		POT
910 OK	30	7	22	PIN

*This is the control register bit. Also, the three status indicators of BUFFER FULL, BUSY and REQUEST, are considered to be bit 23 of a register. Thus, C bit is set to 23 for these indicators.

The following notes are also important:

1. The SDS-910 must fill the data buffer with 72 bits of data;
2. Even though a transfer consisting of fewer than 72 data bits is being done, the full 72 bits must be read;
3. The SKS's on the SDS 910 side behave somewhat differently than one might at first expect. Note that to test buffer full, and branch out when it is not full, the code will be:

EOM	35300	SET UP THE TEST
SKS	30001	TEST BIT 23
BRU	*+2	EXIT THE LOOP
BRU	*-3	TRY AGAIN

Thus, we skip when the buffer full indicator is set.

Similarly, the "15 lock-up" test skips if not set, and the busy test skips if not busy.

At present, the only times the computer to computer interface will cause an interrupt of the SDS 910 is when an RT error is incurred, and when the PDP-15 wants the SDS 910 to read some data. At all other times, interrupts are not generated for the SDS 910, by said interface.

XIII. SDS 910 To PDP-15 Transmission Routines:

There are two programs in OS910 to handle data transmission from the SDS 910 to the PDP-15. They are SEND72 and DMASND. Both are used as tasks in the system, and are invoked because of having been placed onto either the interrupt or regular send queues (INTQUE or SNDQUE).

SEND72 will only transmit 72 bits of information across the interface. It is employed by all interrupt transmissions, and by one and two word read returns.

DMASND will be executed only when a regular send queue entry must perform a DMA-type read return. Any amount of data desired may be sent. The system also may employ DMASND for other purposes, for example, as a debugging tool.

Both SEND72 and DMASND will have their data presented to them as a (possible) list of buffer nodes. After all the information from a buffer node has been sent, the node must be freed up. Each I/O routine encountering buffer nodes does this management.

XIV. READ Routine:

This routine handles all transfer from the PDP-15 to the SDS-910.

These transfers are one of the following:

1. System send.
2. Read request.
 - a) one word.
 - b) two words.
 - c) DMA.
3. Write of information to the SDS-910.
 - a) one word.
 - b) two words.
 - c) DMA.

Note: A one word transfer (either read request or write) is one FIXED or BIT data. A two word transfer is one FLOAT data. A DMA transfer is an array of FIXED, BIT, FLOAT, or CHARACTER data or a character string or a structure of data items.

4. OPEN of experiment file
(i.e., awaken user's programs in the SDS-910).
5. CLOSE
(i.e., shutdown all user's programs in the SDS-910).

On a system send, the status word is changed to show it is a SDS-910 to PDP-15 transfer and the key and "Key in Cal Location" are placed in the appropriate positions for the transfer. The information is sent back immediately so the PDP-15 can see if the interface is still awake and reliable.

With a read request the "Key in Cal Location", which the PDP-15 system needs back so it can match up the transfer with the right user, and the length of transfer desired are put into the user's data area as specified in the ID/Key Tables. The first word put into the user data area is the length of transfer desired. The second is the "Key in Cal Location". The read routine will then schedule the user's task for execution.

A write of information moves the data from the PDP-15 into the user specified data area and then schedules that user. The three SDS-910 interface data words (4 PDP-15 words) are translated into four SDS-910 words. If the fifth bit of the new SDS-910 word (the sign bit in the PDP-15 representation) is set, the system will propagate this to the left so the number will be the same in the SDS-910. If the user is not transferring fixed numbers he must make the necessary adjustment to his data for the above modification.

On an OPEN the user is scheduled so he can enable his interrupt vectors and do whatever initialization is necessary.

On a CLOSE the system gives control to the user's shutdown routine. The user can then reset all of his interrupt vectors to the idler state and return to the system. The system will then clear off all queues for this particular user.

XV. Floating Point POPS:

All of the floating point routines, except the conversion routines, have been taken directly from the SDS-910 POPS Manual. The ones which are available are addition (FSA), subtraction (FSS), multiplication (FSM), division (FSD), and negation (FSN). Two other POPS fixed to float conversion (FFL) and float to fixed conversion (FFI) are also available to the user. All of the above POPS are for single precision floating point numbers.

The conversion routines assume the value to be converted is in the A register (for fixed values) or the A and B registers (for float values) and return with the value converted in the A or A and B registers.

TABLE 1. POPS AND THEIR LOCATIONS

<u>Location</u>	<u>POP</u>	<u>Purpose</u>
100	SKDFST	Schedule a TASK on the fast clock queue.
101	SKDMED	Schedule a TASK on the medium clock queue.
102	SKDSLO	Schedule a TASK on the slow clock queue.
103	SKDTSK	Schedule a TASK in the system.
104	PSAV	Save locations 0 and 1 and A, B and X registers.
105	PRST	Restore locations 0 and 1 and A, B and X registers.
106	SKE	Skip if equal.
107	SETOVF	Set the overflow indicator.
110	RSTOVF	Reset the overflow indicator.
111	SEND1	Send one word (FIXED) to the PDP-15.
112	SEND2	Send two words (FLOAT) to the PDP-15.
113	SENDMA	Send a DMA type transfer to the PDP-15.
114	SNDINT	Send an interrupt to the PDP-15.
115	FFL	FIXED to FLOAT conversion.
116	DIV	FIXED divide.
117	MUL	FIXED multiply.
120	FSN	FLOAT negate.
121	FSD	FLOAT divide.
122	FSM	FLOAT multiply.
123	FSS	FLOAT subtract.
124	FSA	FLOAT add.
125	FFI	FLOAT to FIXED conversion.

APPENDIX A

<u>System Flowcharts</u>	<u>Page</u>
PSAV POP	41
PRST POP	41
COLD ROUTINE	42
TASKER ROUTINE	43
MAKNDE ROUTINE	44
CLOCKA and CLOCKC ROUTINES	45
CLOCKB ROUTINE	46
SEND1 POP	47
SEND2 POP	48
SENDMA POP	49
DMASND ROUTINE	51
SEND72 ROUTINE	52
GETINT ROUTINE	52
SNDINT POP	53
COMPRES ROUTINE	54
ACTIVE ROUTINE	55
SHTDWN ROUTINE	56
RTERR ROUTINE	57
READ ROUTINE	58

APPENDIX B

Priority Scheduling*:

OS910 employs a priority scheduling and queuing technique that, although simple in nature, insures a useful priority ranking. Two places exist where task dispatching may take place. First is in the dispatcher routine, TASKER, where transmissions (SENDS) from the SDS 910 to the DEC PDP-15 are arranged and regular user task scheduling is handled. Secondly, the clock may have tasks scheduled on their queues for execution at the time of a clock "tick".

In light of their volatile nature, the transmission of interrupt derived information must be handled in a high priority fashion. Thus, interrupt queue (INTQUE) transmissions are managed first. Regular data transmissions are next in importance for the dispatcher, and are handled after all queued interrupt transmissions have taken place. Last in line for dispatcher handling are the user tasks that have been scheduled for execution. The clock queue management will be covered later.

The above method ensures that transmissions are handled with the greatest speed. Although using a scheme where there must be a specified (threshold) number of nodes present in a set of queues before any node scheduling is done seems desirable, the pitfalls presented by this method are many. Included in the pitfalls are the following:

1. Interrupt transmissions will be unnecessarily held up;
2. Transmissions may never be scheduled if the threshold

*The reader should consult the appropriate flowcharts in Appendix A.

value is too high;

3. No discernable difference between having the threshold and not having it can occur if the threshold value is too low;
4. Since the DEC PDP-15 will normally be waiting for data from the SDS 910, it is necessary that the data are sent without interruption.

Thus, the queues available to the dispatcher are handled sequentially from the interrupt send queue (INTQUE), to the regular send queue (SNDQUE), to the task queue (TSKQUE). The first non-empty queue, in order, is handled in the following fashion (the interrupts are first disabled):

1. The front node on the queue (queues are arranged as a front linked list, and with a front/back pointer pair available that references the list) is unchained from the list, and the list is rechained to exclude this node;
2. If the ID present in the node is for a currently inactive task, then:
 - a) if this is not a TSKQUE entry, all the acquired buffer storage is freed,
 - b) the queue node free storage is released,
 - c) start over on checking the queues;
3. If the task is active:
 - a) save the contents of the A,B and X registers, locations 0 and 1, and CCBP;
 - b) set the A register contents to the value of the parameters pointer field, and
 - c) execute the task, with the interrupts enabled;
4. When the task returns:
 - a) disable the interrupts,

- b) free the queue node entry,
- c) enable the interrupts,
- d) and start over again on scanning the queues;

Thus, a rudimentary priority ranking is maintained.

The clock queues are managed somewhat differently, as each clock has only a single queue associated with it. A clock queue's entries are collected until that clock "ticks".

The form of a clock queue is:

When the clock "ticks":

1. the queue pointer contents are copied to a working area, allowing later reference to the queue;
2. the queue pointer value is set to "empty" (-1), as indicated by λ ;
3. the tasks on the queue are sequentially dispatched (if active);
4. when the queue is empty, the clock manager returns to the interrupted environment by executing an indirect branch, thus clearing the interrupt level.

Obviously, the algorithm used prevents a clock level from "hitting" during execution of the tasks on that clock's queue. Also, all lower priority interrupts are excluded from being recognized until the clock

queue is emptied, and all higher interrupt levels are allowed to hit, as the interrupt system is enabled.

It is anticipated that all tasks queued onto a clock queue are kept small (no more than 50-100 instructions, each), and perform only absolutely essential time-dependent processing. Any operation that usurps a great amount of time will seriously degrade the performance of the complete system.

APPENDIX C

Round-Robin Scheduling:

The task scheduling technique employed in OS910 is the most rudimentary of all (multiprogramming) methods available. Called the round-robin approach it is a scheduling technique based on the "trust thy neighbor" concept. This section will demonstrate the basic concept.

First, we will consider all tasks to be on a simple (forward linked) list, as:

Beginning with the first task on the list, we:

1. chain around the first node on the list,
2. execute the code for the task that was just unchained from the list,
3. upon completion of execution of that code (indicated by a return from the code), free up the list node storage, and
4. start over on the above process.

As can be seen, the amount of time allocated to each task is that amount of time the task needs to completely execute. If a task were to get into an "infinite loop", then no subsequent tasks would be scheduled for execution.

At this point, the reader may wonder why anyone would even employ such a naive scheduling technique as the round-robin. As OS910 assumes relatively short tasks will be scheduled (300-500 instructions, taking, at most, about 5 msec.), the use of even the fast clock (12.8 ms.) is too slow to cause a resolution.

Another problem avoided by using a round-robin technique is that of rather extensive queuing and data structuring problems. Interrupts would be more difficult for users to directly service under a time-sliced method (using a clock to interrupt the execution of one task and go on to the next), and would force the operating system to do more of the processing than is really required by the problem. Because of the queuing difficulties, a more extensive set of data structures would be needed.

APPENDIX D

USER SECTION:

This section will detail the parameters imposed upon users of OS910, including user task structure, register management, and available system functions. The section will be kept as self-contained as possible, and thus will take the role of a users manual.

User Program Structure

The structure of a users program is the following:

ID/Key table
task ₁
.
.
.
task _n

The ID/Key table, which is supplied by the user, has the form:

		0	8	9	10	23
word:	1	ID	a/i		# key entries	
	2	addr. of users shutdown routine				
	3	Key 0 buffer location				
	4	Key 0 service routine entry point addr.				
	.	.				
	.	.				
	.	.				
		Key n buffer location				
		Key n service routine entry point addr.				

Taking each section in order, we have:

1. word 1:
 - bits 0-8: the ID value (in the range 511-537) assigned by the system staff.
 - bit 9: the a/i (active/inactive) flag for the user. If this bit is set to zero (0), then the user is assumed to be in the active (running) state.
 - bits 10-23: a count of the number of key entries (one greater than the maximum key to be used) to be expected.
2. word 2: this is the entry point address of a user supplied task that is to be executed when:
 - a) a CLOSE is performed in the DEC PDP-15 for the users experiment file, or
 - b) a real-time (RT) error is incurred by the user (discussed later).
3. The entries: key; buffer location: these fields give the location of a buffer to hold any data to be transmitted from the DEC PDP-15, using the associated key value.
4. The entries: key; service routine entry point address: These fields contain the entry point address of the task to be used to treat the data associated with the ith key.

This table is used in the following manner:

1. When a task is encountered that needs dispatching, the a/i bit for the user is interrogated. If the bit is set to one (1), then the task is not dispatched; rather, it is flushed from the system;
2. When the task that resides in the DEC PDP-15 performs an OPEN for the EXPERIMENT file, then the a/i bit of the associated user in the SDS 910 is set to zero (0);

3. When information is sent to the SDS 910 for the user, the key, say 1, associated with the data, is used to find the address of that area of the users storage to be used to contain the data (the storage area must be large enough to contain both control information and the data). Associated with the buffer address is a service routines entry point address. Using the address of the (user supplied) buffer, the entry point address for the service routine, and the users ID value, a task queue node is created, thus scheduling the execution of the data's service routine;
4. When the user incurs a real-time (RT) error, the RT error servicing routine (a part of OS910) uses the ID/Key table entry at word 2, to instigate a shutdown of the users activity;
5. When the users code in the DEC PDP-15 performs a CLOSE of the EXPERIMENT file, a record is transmitted to the SDS 910 to indicate this. In this case, the basic action in (4), above, is used to deactivate the users code.

The ID/Key table is user generated. The table is very easy to create. For an ID value of 512, and a total of three (3) keys, the table can be constructed in the following manner:

OCT	51240003	(ID, A/I, # KEY ENTRIES) VALUE
HLT	SHUTDN	ADDRESS OF THE SHUTDOWN ROUTINE
HLT	KEYOBF	KEY ZERO BUFFER ADDRESS
HLT	KEYOEP	KEY ZERO SERVICE ROUTINE E.P.
HLT	KEY1BF	KEY 1 BUFFER ADDRESS
HLT	KEY1EP	KEY 1 SERVICE ROUTINE E.P.
HLT	KEY2BF	KEY 2 BUFFER ADDRESS
HLT	KEY2EP	KEY 2 SERVICE ROUTINE E.P.

The first word, declared OCT 51240003, has the field values: bits 0-8 set to 512₈, bit 9 set to 1 and bits 10-23 set to 00003. The routine names have been arbitrarily chosen. The declarations done using

HLT name

create a word containing the storage address for name.

If a user does not desire to have a task automatically scheduled for a specific key, a simple artifice must be used. The service routine entry point field must reference a (user supplied) routine consisting of nothing more than:

```
RTNE      PZE
          BRR      *-1
```

Thus, no work will be done by the "servicing routine", but rather will just return.

User Interrupt Processing

Let us now turn to the matter of interrupt processing. The user has the responsibility of managing the interrupts associated with his experiment. It is required that one routine be accessible by OS910, the shutdown routine. The action of this program is very specific. It must, for regular interrupts, replace the contents of the interrupt vector location with

```
BRM      2
```

A recommended code sequence would be similar to the following:

SHUTDN	PZE		
	STA	SHUTTI	
	LDA	BRM	
	STA	iv ₁	
	.		
	.		
	.		
	STA	iv _n	
	LDA	SHUTTI	
	BRR	SHUTDN	
SHUTTI	HLT	0	WORK AREA
BRM	BRM	2	REFERENCES "DUMMY"

The instructions "STA iv₁" through "STA iv_n" are used to store the code "BRM 2" in each regular interrupt vector location for the user. To safeguard the system from having the overflow indicator set at random times, it is recommended that the user also set his special interrupts to the instruction "NOP 0". This will ensure that no side effects will take place, and that the interrupt is efficiently and effectively idled.

The user also has the responsibility of "activating" his own set of interrupts. Interrupt activation depends on what kind of interrupt is being set up. For special interrupts a MIN, MDE, etc., for the desired counter area must be stored in the users special interrupt locations. For regular interrupts, the user needs to store a BRM to the entry point of the respective servicing routine.

There is one error that affects the user, but is not within his sphere of control. This is the real-time (RT) error interrupt. Among other factors, an RT error may be incurred from the action of incorrectly addressing a users experiment. In all cases, the user is scheduled for shutdown; i.e., an interrupt code, with the ONCODE set to zero, is scheduled for transmission to the DEC PDP-15, and a shutdown routine is scheduled for task execution that references the users ID/Key table, and will deactivate his task, then invoke the (user supplied) shutdown routine.

System Conformability

There are two actions a user must have in his code to maintain system integrity. When the system causes the invocation of a users task, the task that was entered should:

1. Save the contents of the A,B, and X register, locations

zero and one, and the CCBP value (see the CVT description). This is done very simply by employing code similar to the following (assuming SAVE is a six word area that is free for use):

STA	SAVE	save the A register contents
LDA	0	retain location 0's and contents
PSAV	SAVE	execute the save POP
LDA	user id	A gets the users ID
STA	CCBP	store it in CCBP

The above code will also replace the contents of CCBP (a system variable) with the users ID value. This allows the operating system to always know the current users identity.

The second action to perform is required of the users task that does the eventual return to the operating system. The action is that of restoring the above saved areas. This is performed by executing:

PRST SAVE

This will restore the old contents of the A,B and X registers, locations 0 and 1, and CCBP, which are kept in SAVE in that order.

The above save/restore actions must also be employed by all (user furnished) interrupt handling routines. This will help to ensure that the program that was interrupted will not be adversely affected.

User Task Scheduling

Users have the ability to effect the scheduling of tasks. There are two classes of user scheduled tasks: normal, and clock dependent.

A normal task is one that will do, say, angle derivation for one of the experiments, or a task that OS910 schedules for the user, to treat data sent to this user from the DEC PDP-15. To effect the scheduling of such a task, the user executes the POP:

SKDTSK 0

Here, the operand field is not used, but cannot be eliminated due to assembly language restrictions. The contents of the A and B registers supply the required parameters. The A register must contain the entry point address of the task being scheduled, and the B register must contain either the address of the (possible) set of parameters, or be set to -1 (indicating no parameters). Also used is the contents of CCBP.

There exist three clock queues that the user may have tasks scheduled on. The associated clocks have periods of 12.8 msec. (fast clock), 102.4 msec. (medium clock), and 1.6384 sec. (slow clock). The clocks may be profitably employed when tasks must be executed after performing a delay operation. This could happen when a shaft on a test bed is to be swung thru an arc, and the amount of time required for the operation is both known and at least as much as the fast clock. The user could then schedule a task on the appropriate clock, and "delay" processing.

To perform the scheduling, the user:

1. sets the contents of the A register to the entry point address of the task to be scheduled;
2. sets the B register contents to reference any parameters;
3. executes one of:

SKDFST	0	Fast Clock
SKDMED	0	Medium Clock
SKDSLO	0	Slow Clock

where the operand, although required by the assembler, is not used, and is arbitrary.

DEC PDP-15 to SDS 910 Communication:

It should be kept in mind that programs residing in the SDS 910 have an associated major program residing in the DEC PDP-15. The DEC PDP-15 program treats the SDS 910 as nothing more than a special purpose file structure, and interrupt generator. Thus, it will perform OPEN's, CLOSE's, READ's, and WRITE's, to the EXPERIMENT file (which is the SDS 910, and the users servicing program residing in the SDS 910).

An OPEN of the EXPERIMENT file will cause a message to be transmitted to the SDS 910 with a key field value of zero. Upon receipt of a key zero record, and subsequent notification of the SDS 910's user program, the users interrupts may be "activated", along with any other desired action.

A CLOSE of the EXPERIMENT file causes a "shutdown" message to be transmitted to the SDS 910. Upon receipt of this message, OS910 will schedule the shutting down of the users SDS 910 resident code.

A WRITE to the EXPERIMENT file will transmit data to the buffer area in the SDS 910 users code. Because of this action, a user supplied task will also be scheduled to treat the data. Along with the data sent via a WRITE, the user also receives the length, if this was a DMA transfer.

A READ of the EXPERIMENT file does not transmit any data. Rather, it requests the SDS 910 resident users program to return some information (thus, the term "read request"). Upon receipt of this message, OS910 will store, in the user supplied buffer area, the length parameters, and the field termed "key location in CAL".

Please note that, when numeric values are transmitted to a users program in the SDS 910, the 18 bit data values are expanded to 24 bit form. The 18 bits of data are placed in bits 6-23 of the word, and bit 6 is expanded, to the left, throughout bits 0-5 of the SDS 910 word.

SDS 910 to DEC PDP-15 Communication

There are two methods of communication that an OS910 user has available. First, one is able to initiate a read return (a reply to a DEC PDP-15 initiated READ). Secondly, the user may transmit an ONCODE corresponding to an interrupt signal. Indirectly, the user may cause an RT error interrupt signal (ONCODE = 0) to be transmitted.

There are three (3) read returns; one word, two words, and DMA returns. Each of these requires a specific data structure. These are:

1. One word read return:

word 1:	Length of transfer
word 2:	"Key location in CAL" value
word 3:	Data value

where the first word of this area is the same as that sent with the read request, initially.

2. Two word read return:

word 1:	Length of transfer
word 2:	"Key location in CAL" value
word 3:	Data value one
word 4:	Data value two

3. DMA read return:

word 1:	number of data values to send
word 2:	"Key location in CAL" value
word 3:	data value one
.	.
.	.
.	.
word (n+2)	data value n

where the "number of data values to be sent" does not include the "key location in CAL" field.

These data areas are referenced by the operand field values of the POPs:

1. SEND1 ref:

The reference is to a one word read return area;

2. SEND2 ref:

The reference is to a two word read return area;

3. SENDMA ref:

The reference is to a DMA read return area.

The invoked POP will:

1. create a regular send queue (SNDQUE) entry that references the appropriate transmission routine, and a buffer (or list, thereof) containing the data;
2. compress the data from 24 bit format to 18 bit format, as required by the DEC PDP-15;
3. insert the data into the buffer node(s), as required.

This action, which amounts to dynamic buffering, is done to ensure the

integrity of the users data. If the data were to be left in the users area while awaiting transmission, there would need to be some notification technique developed to ensure that the user did not write over his own data. This approach, although a viable one, seemed too complex for the operating systems protocol and the desired operating environment.

APPENDIX E

Use of the Initial Loader:

1. Turn power on to both machines.
2. Zero PDP-15 core. See systems operation book.
3. Read in PDP-15 system using the system loader tape.
4. Read in the SDS-910 loader tape on the SDS-910.
5. BRU to 6300 in the SDS-910. This will zero core, set-up the interrupt vectors, and cause the loader to wait for the transfer from the PDP-15.
6. On the PDP-15 side do:
 - a) J10OCR press continue on the console.
 - b) CNTL A, the system will respond with
REQ, INQ or ZAP?
 - c) Reply REQ CNTL D, the system will then respond with
TYPE IN USER LIBRARY NAME?
Reply with UNIVERSAL CNTL D.
 - d) The system will respond with
TYPE IN TASK NAME?
Reply with LOADER CNTL D.

The system will respond with
 USER ID IS ##### CORE ADDRESS IS #####
 TASK DISPATCHABLE.

7. The loader routine on the PDP-15 side will now ask:
 DATA SET NAME?
 Reply with name of data set which contains the users program
 and the SDS-910 system to be loaded.

The loader will load the above data set across the interface into
 the SDS-910. It will then reply on the PDP-15 teletype

NORMAL END NEXT DATA SET NAME?

If there are more data sets to be loaded across the interface
 specify them at this time. Otherwise, reply with a CNTL D. This
 will cause the loader to terminate in both machines.

Now you can start any program desired by entering CNTL A.

The system will reply again

REQ, INQ or ZAP?

Reply REQ CNTL D and reply to the rest of the questions with the library and data set that you want to execute.

APPENDIX F

How to Set-up a Data Set on the PDP-15 Disk to be Loaded Across the Interface and into the SDS-910:

1. Use DISKORG to create a norm data set with record length = 44.
To figure the number of sectors needed count the number of cards and divide by 5 and add 1 if there is any fraction left over.
2. Use card read routine to read in program cards through PDP-15 card reader. The program will ask
DATA SET NAME?
Reply with above created data set name.

The program will then read in the cards. If any errors are encountered while reading the cards the program will respond with the appropriate error message.