516 - TSS.-SYSTEM - BOLTED IN-CORE SUBROUTINES

TVDF, A01, 330

Real Time Clock.
Link To Interrupt Handlers.
Link To Dummy Octal Package.
Sector ZERO Image Of THREAD SAVE BLOCK.
Thread Table Pointers.
DISK-DMA QUEUE Pointers.
Segment Table Pointers.
I/O Table Pointers.
Interrupt Save Table Pointers.
Teletype Characters.
System Temporary Storage (Non Reentrant).
Disk Interrupt Handler Temporary Storage.
Core Management Pointers.
Disk Management Pointers.
Interrupt Handler Pointers.
Clock Time Table Pointers.
The constants which start with .Z are associated with the THREAD SAVE BLOCK format, and all subroutines which deal with the THREAD SAVE BLOCKS use them making it easy to add, substract or change W/O major reprogramming.

TVDF, A02, 220

TRANSFER VECTOR to programs which are inside the system but are called by external segments. Also the intergers start with .P for + and .M for -

TVDF, A03, 20

SENSITIVE SYSTEM DATA (Save on disk).

TVDF, A03, 20

Disk table and manipulation routines GETID, PUTID, IDTOSZ. The table has 1 bit per block of 64 words (0-Busy: 1-Free). With 5880 words per track, there are 91 usable blocks per track. Hence, the table has 6 words (96 bits) per track. The last 5 bits of a track = 0. Also the first 4 tracks are reserved for system and not included in the table.

GETID allocates a 64-word block on disk by finding and clearing 1 bit in the disk allocation table. The ID (13-bit disk address plus 3-bit size) is returned in the A-register. The X and B registers are not saved.

PUTID frees a 64-word block on disk by suiting the corresponding bit. The ID is supplied in the A-register. It must be possible and the bit must be busy. The X, A, and B registers are not saved.

IDOTOSZ converts an ID to segment size (in words). The ID is supplied in A-register, where the segment's size is returned. The ID is checked for validity. The first block of 64 words must show busy. The X-register is saved.

VATOAA, C01, 242

VIRTUAL ADDRESS TO ABSOLUTE ADDRESS.
On entry, X points to VIRTUAL ADDRESS.
On exit, A has a segment starting address; ADDRA, has the relative address (RA); X & C have been lost.

ADTOID, C0 2, 50

ADDRESS TO ID - Convert a segment absolute address stored in the A-register to its equivalent relative address (RA) inside the segment. The relative address is returned in the B-register and the ID-number of the segment is returned in the A-register.

CALL, C03, 34

The first pair of the argument words (entry 0 in the argument list) is the virtual address (ID,RA) of the called subroutine. The absolute address of this VA pair and the BASE ADDRESS of the called segment are appended to the CALL PUSH DOWN LIST. The C-register is lost. Call executes in 42 cycles excluding ENTRY, JST, and VATOAA execution.

GETSEG, CP4, 157

Fetches the requested segment from disk and logs it into core storage.

GOTO, C05, 40

Acts similar to machine JMP across segment bound areas. The GOTO subroutine requires one 2-word argument, the virtual address of the location to which control is transferred. Thus GOTO is similar to call except that it has no argument list and does not manipulate the CALL PUSHDOWN LIST.

IDTOAD, C06, 14

IDTOAD (ID TO ADDRESS) converts an ID in the A register to the starting address of the segment associated with that ID.

IDTOCP, C07, 20

ID TO CHARACTER POINTER. Expects the address of a character pointer in the first argument and an ID in the A register. Upon exit the relocatable pointer part of the character pointer is pointing to the start of the ID's segment and the character pointer is initilized (ready to fetch the first character of the segment).

LOGIN, C08, 31

Finds room for segment in the SEGMENT TABLE and inserts the segment ID and BASE ADDRESS in the in-core ID TABLE. The routine is entered with the ID in the A-register.

SEGCKS (SEGMENT CHECK SUM), C09, 55

Expects a pointer in the X-register to DISK-DMA QUEUE ENTRY to be set up. Upon return the check sum of the segment has been computed and inserted in the last word of the segment. If it checks with the old check sum (last word), then the second exit is taken (JST+2). If it does not check, then first exit is taken (JST+1). Enter the routine with the ID in B-register and segment header in the A-register.

STSCAN (SEG. TABLE SCAN), C10, 20

Expects an ID in the A-register. The segment table is scanned for the ID and if found the first exit is taken with the X-register pointing to the segment table entry (ID word). If the ID is not found, the second is taken.

CHKCOR STEPHD, D01, 171

CHUNK CORE shifts the segments in core so all the holes are at the top of core. The SEGMENT TABLE base addresses, I/O pointers, DISK-DMA QUEUE core addresses, THREAD SAVE block addresses and THREAD TABLE addresses are relocated to reflect the core shift.

- .ST1 - Points to the present starting core location of the next segment to shift.
- .ST2 - Points to the future starting core location of the next segment to shift.
- .ST3 - Contains the size of the next segment to shift!
- .ST4 - Points to last hole entry in hole table.

Core is moved at about 8 cycles/word. Average CHKCOR time = 30 milliseconds

STEP TO NEXT HEADER. Steps to the next segment header word in core. One of 3 exits is taken depending upon the header and I/O pointer count of the next segment. If the segment is not a hole (segment type not 0) and the I/O pointer count is ZERO, the first exit is taken. If the I/O pointer is NONZERO, the second exit is taken. If the segment is a hole or there is not next segment, then the third exit is taken.

- .ST1 - Points to the present starting core location of next segment to shift.
- .ST2 - Points to the future starting core location of next segment to shift.
- .ST3 - Contains the size of the next segment to shift.

MOVCOR (MOVE CORE), D02, 17

Moves a block of core down over a deleted segment or segments (hole). Words are moved at a rate of 7.2 cycles per word.

- .ST1 - Points to the present starting core location of the next segment to shift.
- .ST2 - Points to the future starting core location of the next segment to shift.

RELBLK (RELOCATE BLOCK), D03, 15

ON ENTRY, X-register points to first word of block to be relocated. The A-register contains the distance between words in the block to be relocated.

.ST7 - Address +1 of the last word in the block to be relocated.

ON EXIT, all words in the block have been relocated.

RELCOR (RELOCATE CORE PTRS), D04, 74

Relocates all the SENSITIVE (ABSOLUTE) CORE pointers according to the core shift specified by the HOLE TABLE. The SEGMENT TABLE addresses, the DISK-DMA QUEUE core pointers, the THREAD TABLE THREAD SAVE BLOCK addresses and all the absolute core pointers inside the THREAD SAVE BLOCKS are changed to reflect the core shift.

Approximate time = 2250 + 300N cycles where N = number of threads.

RELITV (RELOCATE I/O TV), D05, 37

When a segment with I/O pointers to it is moved, the interrupt is turned off. The segment is moved, the I/O TV is relocated, and the interrupt is turned back on.

RELOC (RELOCATE), D06, 30

Expects a core address (14 bit) in the A-register and assumes it is greater than the starting address of the lowest HOLE in core (it needs relocation). This address is then relocated to reflect the core shift defined by the hole table.

Time (including JST RELOC)

HOLE 1-2 = cycles
2-3 = 14
3-4 = 29
4-5 = 34
5-6 = 47

RELPTR, SYSRP, D07, 121

RELOCATABLE POINTER updates a RELOCATABLE pointer (first argument) with the contents of the A-register. The pointer counts of the segment associated with the old and new contents (RELOCATABLE POINTER) are adjusted.

SYSTEM RELOCATABLE POINTER updates a RELOCATABLE POINTER (first argument) with A-register but ignores old contents of RELOCATALBE POINTER.

SPACE D08, 176
Expects the amount of core space required to be in .TT2. Upon exit, the space has been allocated and the starting address of the space is in the A-register.

CSPACE (CORE SPACE), E01, 14
Allows outside calls. Expects the total size in the A-register. Upon exit, a data type block has been generated and made permanent (POINTER COUNT = 1). The starting address is returned in the A-register.

FRESEG (FREE SEGMENT), E02, 101
Frees a segment. Expects SESSA to contain the segment starting core address and STPTR to be pointing to the segment entry in the SEGMENT TABLE. Upon exit, the segment has been logged out of the SEGMENT TABLE and marked for removal or placed on the DISK-DMA QUEUE if it must be restored to disk.

HOLSEG, E03, 11
Mark segment as removable (a hole). Expects the starting core address of the segment in the A-register.

NEWSTR, NTSEG, E04, 44
NEW STRING expects a character pointer in the A-register. Upon exit, a new segment (first of a string) has been generated and the character initialized to its start. (RELOCATABLE POINTER points to the start of the segment).
NEW TEXT SEGMENT generates a new text segment. Upon exit, the segment ID is in the A-register and its starting address is in the B-register.

POSEG (PUSH ONE SEGMENT), E05, 54
Causes one segment to be pushed out of core. The choice of segment is semirandom over the set of possible segments. If there is no possible segments to push out, the first exit is taken. If a segment is pushed, then the second exit is taken and the future top used core address (.FTUSE) is decreased to reflect the space obtained. The A-register is left with the negative of the pushed segment size.

TISEG (TAKE IN SEG), E06, 25
Expects the total size of the segment in the A-register. A new DATA SEGMENT is generated and its starting address and ID are returned in the A&B-register respectively.

TOSEG (THROW OUT SEGMENT), E07, 17
Marks a segment as a HOLE and takes care of SYSTEM POINTER including logging out the segment. Expects the ID of the segment in the A-register.

DISKIO (DISK I/O), F01, 251
Handles the interrupts for the DISK-DMA. When a disk-core transfer is finished, the 516 is interrupted and DISKIO scans the DISK-DMA QUEUE for best transfer to make next. It gives control to that entry types handler.

DISKRW (DISK READ AND WRITE), F02, 33
Called from outside using the following calling sequence:

```
JST DISKRW
DISK ADDR
WORD COUNT (-READ, + SIGN FOR WRITE)
```

DQUECL (DISK-DMA QUE ENTRY CLEAR), F03, 13
Marks the DISK-DMA QUEUE entry pointed to by the X-register as being unused.

DQUEX (DISK QUEUE X-REG), F04, 26
Finds an empty slot in the DISK-DMA list and leaves the X-register pointing to the first word. The transfer address for the new queue entry is given as the first argument. A pointer to the present THREAD TABLE ENTRY is also inserted in the queue entry.

RWDISK (READ WRITE DISK), F05, 53
Expects a pointer to a 3-word block in the X-register. The first word of the block is the starting disk address. The second is the starting core address. The third is the word count. Upon return the disk is read into or written from the section of core specified by the three word block. The read-write bit is passed in the C-register. (1 = read disk, 0 = write disk).

GATE, G01, 35
Called from outside the system. Allows only one thread to pass through. All others are roadblocked until the GATE is opened by the first thread. Another thread is then allowed through.

GETA, G02, 16

Causes: 1) The absolute address of the appropriate argument list entry to be stored in ADPTR 2. The conversion of this list address to an absolute address to be stored in ADARG 3. The content of the address stored in ADARG to be loaded in the A-register.

JRET, JSUBR, JSUBRA, G03, 73

JST RETURN is a partner to JSUBR. It returns to the caller through .JSTAD after popping an entry off the JST PUSH DOWN LIST and updating .JSTAD. The A, B and C registers are untouched or restored. (Time = 22 cycles including JMP JRET).

JST SUBROUTINE is called to protect a JST address from core shifts or reentry by another thread. A JST to JSUBR must be the first instruction after the JST address to be protected. The A, B and C registers are untouched or restored. Upon exit, the location JSTAD is set up and can be stepped with IRS. Also it can be used indirectly with other instructions (e.g., LDA, etc.) to fetch arguments. It should be left pointing to the desired return location when JRET is called. (Time = 39 cycles including JST JSUBR).

RCALL, RRET, RRET2, RRET3, G04, 34

RCALL AND RRET, call and return mechanism for external segments which does not hold the calling segment in core.

NEWTHD (NEW THREAD), G05, 102

Looks to see if the new THREAD SAVE BLOCK is in sector 0. If it is not, then the old THREAD SAVE BLOCK is moved out and the new THREAD SAVE BLOCK moved into sector 0. (Thread change takes approx. 650 cycles).

RET1, RET 2, RET 3, G0 6, 27

Return has a single, one-word argument, which is the argument number for the return. Thus, a subroutine with N arguments would ordinarily return to N+1, the first location after the argument list. The CALL PUSHDOWN LIST is popped one level. The C-register is lost. (Execution time is 36 cycles).

ROPAK (REENTRANT OCTAL PACKAGE), G07, 22

Incore part of REENTRANT OCTAL PACKAGE. It allows a one word link (JST .ROPK..*) in the calling segment. The machine registers and the calling virtual address are passed to segment part of ROPAK.

SYSERR (SYS. ERROR), SYSER2, G08, 30
Recoverable SYSTEM ERROR. Call ROPAK.

TTHYZ, TTXXYZ, G09, 50
THREAD TABLE ENTRY HEADER changes the third table entry header word pointed by .TIPTR as follows:
X indicates where to get transfer address for headers
I - Get address (indirectly) through JST ADDR.
D - Use the JST ADDR. (direct).
S - Use the (same) address as in the header.
Y - Indicates how to mark the header (high order bit).
P - (Plus) not roadblocked ready to process.
N - (Minus) roadblocked.
Z - Indicates what to do next.
S - (Step) to next thread.
- Do not step continue processing this thread.
TTXXYZ is the same as TTHXYZ except that the header pointed to by the X-register is affected.

RDBLK (ROADBLOCK), G10, 4
Called from outside the system. It allows all the other threads to get a turn at the computer and then returns to the caller.

IDLER H01, 221
Is the processing allocator of the multiprogramming system. It steps from thread to thread processing when it can, skipping over threads which are roadblocked. (14 cycles to skip over a roadblocked or unused thread.)

INTRPT, INTRET, H02, 337
Service interrupts. They save and store the machine states for multiple level interrupts.
INTRPT fields by the interrupts through 63_8.
INTRET used by special interrupt handlers to return control to interrupted programs.

GETCR, PUTCR, INCRCP, I01, 250
GETCR - Get character using relocatable character pointer.
GETCHR .ICOP
ERROR
PUTCR - Put character using relocatable character pointer.
PUTCHR . LOCP.

INCRCP - Increment character pointer.
INCRCP .IOCP

ESCAPE, J01, 24
It allows the user to escape to system level.

IODONE, IOSTEP, J04, 14
IODONE is called by I/O INTERRUPT HANDLERS when an I/O transaction is complete and the requesting thread should be unroadblocked. The I/O TABLE pointers are also reset. Expects the I/O table entry pointer in the X-register.
IOSTEP is called by I/O INTERRUPT HANDLERS when an I/O buffer is exhausted without finding a terminating character and a new I/O buffer is requested.

IØNO, J05, 30 - Disable an I/O TABLE ENTRY from interrupts.

IORBLK (I/O ROADBLOCK), J06, 16
Roadblocks the present thread and steps to the next. When the I/O is complete, the I/O HANDLER should unroadblock the thread and when its turn comes up, control will return to the calling segment.

IOTSRT, IOTEND, J07, 120
I/O TABLE - 5 words per entry contains the I/O associated with each node device on the I/O ring (node number ZERO is the control teletype on the DDP-516).
1st word - Absolute address of interrupt handler for node.
2nd word - Absolute address of I/O buffer for node.
3rd work - Buffer pointers.
4th word - End of message character and internal buffer pointer.
5th word - THREAD TABLE entry address.

RINGI, WCRING, WDRING, RSRING, RDRING, INRING, J08, 207
RINGI finds out which node interrupted. Reads node status into .RSTAT. Sets up .ITERM, .ITPTH, and branches to the I/O handler.
WCRING writes C(A) command to node (.ITERM).
WDRING writes C(A) data to node (.ITERM).
RSRING returns status of node (.ITERM) in A-register.
RDRING returns data from node (.ITERM) in A-register.
INRING writes command to node (.ITERM) for initialization.

SRTTHD, (START THREAD), J09, 106
Is called by the interrupt handler to initiate a thread in the multiprogramming system. It does part of the job under the interrupt and part under the multiprogramming time slot it is given. The rest of the bootstrapping procedure is completed by an outside segment.

TPCRLF J10, 11
Type a carriage return and a line feed.

TPTEXT, J11, 12
Expects the RA of a text string in the A-register. Upon exit, the string has been typed out. The RA must be in the segment calling TPTEXT, and the text string must end with a null (00).

TTIO, (TELETYPE I/O HANDLER), J12, 276
Interrupt handler for ring and control teletypes.

TYPEIN, TYPOUT, TYOXID, HANGUP, ENDID, HUTST, J13, 34
They provide the multiprogrammed buffered teletype I/O. They set up the appropriate I/O TABLE ENTRY to handle the I/O under interrupt control. The THREAD ENTRY is roadblocked during the I/O and other threads are processed. When the I/O is completed, the thread is unroadblocked. Upon exit, the A-register and .IOCP contain relocatable character pointers to the last character taken in or put out unless the I/O could not be setup. -1 is returned in the A-register with TYPOUT.

WAIT, J14, 12
Roadblock the thread for N-seconds where N is the first argument.

ZZ, J15, 3
Is a dummy segment used to keep the relocatable pointer count for the system and to define the top of the system for the pointer .TSYST.

SEGMENT BUILDER (BOOTSTRAP) - TEMPORARY SUBROUTINES

BOOTST, 001, 1433 - Fetches an out of core segment from the card reader. This segment builder is part of the hard-core system. It reads segmented programs from the card reader and puts them in core. There is no check on whether a segment with the same name already belongs to the system.

START, 002, 153

Initialize the ID-TABLE on disk.

OCTPKG, 003, 136

Saves core on disk and brings in the real OCTAL PACKAGE.