CHAPTER 4

PDOS MONITOR COMMANDS

The PDOS monitor is a set of resident routines for handling the most common commands to PDOS such as defining and deleting files or listing file directories. Commands are passed to the monitor from the Command Line Interpreter (CLI). A list of memory resident commands is searched followed by the disk directory using the command as the file name.

4.1 APPEND FILE

Format: AF <file1>,<file2>

The APPEND FILE command concatenates two PDOS files together. File <file1> is appended onto the end of file <file2>. The file type attribute of <file1> is transferred to <file2>. <file1> is not affected by the operation.

A control C (^C) interrupts this function on a sector boundary, closes both files, and returns to the monitor. This action is reported by the message '^C'.

```
.AF PART2/1,PART1
.AF PART3/1,PART1
.AF PART4/1,PART1
.
```

4.2 BAUD PORT

Format: BP <port #>{,<baud rate>{,<CRU base>}}

The BAUD PORT command initializes any one of the eight PDOS I/O ports and binds a physical TMS9902 UART to a character buffer. The command sets the 9902 character format, receiver and transmitter baud rates, and enables receiver interrupts.

The first parameter <port #> selects the console port and ranges from 1 to 8. The system variable ITBCRU, located at address >0096 (>00B6 for 102), points to the input CRU base table. This table binds a physical 9902 UART to a port character buffer and is generated during PDOS initialization. Entries in this table are changed by the BFIX utility or by the third parameter of the 'BP' command.

The TMS9902 UART's control register is initialized to 1 start bit, 7 bit character, even parity, and 2 stop bits (11 bits). The receiver and transmitter baud rates are initialized to the same value according to the <baud rate> parameter. The <baud rate> parameter ranges from 0 to 7 or the corresponding baud rates of 19200, 9600, 4800, 2400, 1200, 600, 300, or 110. Either parameter type is acceptable.

If a minus (-) precedes the port number, then the associated CRU base address is stored in the UNIT 2 (U2C(9)) variable. The <CRU base> is optional and is included when binding a logical port to a different 9902 UART.

```
.BP 2,1200       Set port #2 to 1200 baud
.BP 3,1,>A00     Set port 3 to 9600 baud
                   and CRU base address >A00

 Port #1 = >0080       TM9900/101MA main port
        2 = >0180       TM9900/101MA auxiliary port
        3 = >0E00       ER3232 sel #1 page #0
        4 = >0A00       ER3232 sel #3 page #0
        5 = >0A40       ER3232 sel #3 page #1
        6 = >0A80       ER3232 sel #3 page #2
        7 = >0AC0       ER3232 sel #3 page #3
        8 = >0B00       ER3232 sel #3 page #4

9902 initialized for 11 bits:
        1 start bit
        7 bit character
        1 even parity
        2 stop bits

<baud rate>  0 = 19200 baud
             1 = 9600 baud
             2 = 4800 baud
             3 = 2400 baud
             4 = 1200 baud
             5 = 600 baud
             6 = 300 baud
             7 = 110 baud

.BP -3,9600      Set port 3 for UNIT 2 at 9600 baud.
```

4.3 COPY FILE

Format: CF <file1>,<file2>

```
.CF PROGM:SR,PROGM:SR/1
.CF FILE1,FILE1:BK
```

The COPY FILE command copies <file1> into <file2>. The original <file2> is destroyed and replaced by <file1>. The file type attribute of <file1> is transferred to <file2>. <file1> is not affected by the operation.

A control C (^C) interrupts this function on a sector boundary, closes both files, and returns to the monitor. This action is reported by the message '

4.4 CHECKSUM PDOS

Format: CS

```
.CS
.EX
*READY
MEMW(090H)=123
BYE
.CS
PDOS ERR=80
.
```

The CHECKSUM command offers a fast check of the PDOS system memory integrity. Memory is summed from memory addresses >0080 to >1FFF. If the result is zero, the command returns to the monitor prompt with no messages. Otherwise, PDOS ERR 80 is reported, indicating that memory has been altered.

Note: Altering interrupt or XOP vectors does not cause a checksum error. PDOS 102 sums memory from >00A0 to >1FFF.

4.5 CREATE TASK

Format: CT <task>,<size>,<time>,<port>

The CREATE TASK command places a new task entry in the PDOS task list. Parameters for the new task include a command line, memory size, number of CPU clock tics, and an I/O port. The new task number is reported after the task is created.

.CT PRGM,20,1,2 TASK #1	Create 20K byte task, port #2, 1 tic

The <task> parameter is the command line for the new task. The string is passed to the new task via a message buffer and hence cannot exceed 50 characters in length. You are reminded of this length by a bell when entering a command line. If the first parameter is omitted, then the PDOS monitor is used. Multiple commands and parameters are passed using parentheses.

.CT HELLO,,1,0 TASK #2	Spawn scheduler
.CT (ASM PRGM:SR,PRGM),10 TASK #3	Spawn background assembler

The amount of memory for the new task is given by <size> and is in 1K byte increments. The system memory bit map is searched for a contiguous block of memory equal to <size>. If the search fails to find a large enough block, then memory is taken from the parent task and allocated to the new task. Default is 1K bytes. (PDOS 102 does not take memory from the parent task. Also, the memory parameter is changed to the next larger 4K byte boundary. The default is 32K bytes.)

```
.CT ,10,,3
TASK #4
```

The <time> parameter specifies the number of clock tics (1/125 second) the task executes before PDOS swaps to the next task. Default is 3 tics or 24 milliseconds.

```
.CT WATCH,,1
TASK #5
```

The <port> parameter assigns to the new task an I/O port. The range is from 0 to 8. Port 0 is the default and is called the phantom port. On the phantom port, all character outputs are ignored while requests for character input result in the task suspending on event 95. More than one task may be assigned to an output port. The input port is a unique assignment and cannot be shared with another task. Input ports are allocated on a first come basis.

```
.CT ,4,,2
TASK #6
.CT ,4,,2
TASK #7
```

After a task is created, the spawned task number is reported. This number is used in killing the new task.

(See also 4.15 KILL TASK and 4.18 LIST TASKS.)

4.6 DEFINE FILE

Format: DF <file>{;<level>}{/<disk>}{,<sectors>}

The DEFINE FILE command creates a new file in the disk directory as specified by the file name <file>. If no <sectors> parameter is included, the file is sequential and one initial sector is allocated to the file.

If <sectors> is nonzero, then a contiguous file is defined and the specified number of contiguous sectors is allocated, linked, and assigned to the file. A contiguous file facilitates random access to the file data since PDOS can directly position to any byte within the file without following sector links.

If a contiguous file is extended past the original allocation length, then the contiguous file attribute is deleted. Therefore, even though contiguous files can be extended, you should allocate enough sectors when the file is first defined to handle all anticipated data. Otherwise, random file access slows down.

The length of a contiguous file is specified in sectors. Each sector contains 252 bytes or characters of data. The file size is given by the number of sectors times 252.

```
.DF FILE1;3/1      Define sequential file on
                     level 3, disk 1
.DF FILE2          Define "FILE2"
.DF FILE3;10,20    Define contiguous file of
                     length 20*252 or 5040
                     bytes on level 10

.DF FILE4;10/1,35
.
```

Bytes = # of sectors x 252

4.7 DELETE FILE

Format: DL <file>

```
.DL FILE1
.DL FILE2/3
.
```

The DELETE FILE command removes from the disk directory the file specified by <file>. All sectors associated with that file are deallocated in the disk's sector bit map and freed for use by other files on the same disk. A file cannot be deleted if it has previously been either delete or write protected. These protection flags must be removed with the 'SA' command before the file can be dropped from the disk.

A sector bit map is maintained by PDOS on each disk so that file creation and deletion does not require a disk compaction routine to recover lost disk space. However, frequent file definitions, deletions, and extensions does create small groups of contiguous sectors which tend to fracture files and make the creation of contiguous files impossible. This is remedied by periodically transferring all files to a newly initialized disk which allocates sectors sequentially for each file.

(See also 13.55 TRANS.)

4.8 SET/RESET EVENT

Format: EV {<event>}

```
.EV
 >0000 >0000 >0000 >0000 >0000 >0000 >0000 >FFFF
.EV 42
WAS 0
.EV
 >0000 >0000 >0020 >0000 >0000 >0000 >0000 >FFFF
.EV -42
WAS 1
.EV
 >0000 >0000 >0000 >0000 >0000 >0000 >0000 >FFFF
.
```

PDOS events are set, reset, or listed with the EV command. A positive <event> parameter sets the event (1), while a negative parameter resets the event (0). If no parameter follows the command, then all 128 events are listed to your console as 8 16-bit hex constants. The first 16 events are shown in the first constant with event 0 being the most significant bit and so forth. (For more information, enter 'HE EVENT'.)

4.9 PDOS BASIC

Format: EX

```
.EX
*READY
BYE
.
```

The PDOS BASIC interpreter is entered via the EX command and exited with the BYE or chain (.) command. A PDOS BASIC program is not altered even though BASIC has been exited and reentered, until another object or BASIC program is executed.

4.10 FILE SLOT USAGE

Format: FS

The FILE SLOT USAGE command lists all files currently open along with file slot information. When the first file is opened, it is assigned slot number 32; as successive files are opened, they are assigned file slots in numerical sequence down to 1. (Read Only Open allocates slots in the opposite order, from 1 to 32.) The file slot maintains information such as the current file pointers and sector indexes. This data is defined as follows:

SLOT	File slot #	Allocated from 32 to 1
NAME	File name/disk #	
ST	File status	
SM	Current sector in memory	
PT	Current file pointer	
SI	Sector index of SM	
SE	Sector index of END-OF-FILE sector	
BE	# of bytes in EOF sector	
TN	Lock Task/Open Task	
BF	Buffer pointer	
LF	Lock flag/# Shared	

File status is defined as:

ST =	>8xxx	Sector altered
	>4xxx	File altered
	>10xx	Driver in channel
	>0Axx	Read only access
	>06xx	Shared random access
	>02xx	Random access
	>01xx	Sequential access
	>xx04	Contiguous file
	>xx02	Delete protect
	>xx01	Write protect

.CT (ASM/4 TIBO:SR,TIB/4),20

SLOT	NAME	ST	SM	PT	SI	SE	BE	TN	BF	LF
1	HLPTX/0	>0A03	>0021	>26AA	>0004	>0004	>0022	>0000	>26BE	>0000
31	TIBO:SR/0	>0102	>004E	>2813	>0006	>001F	>00B8	>0001	>279E	>0000
32	TIB/4	>0100	>0098	>0000	>0000	>0052	>00F1	>0001	>0000	>0000

4.11 EXECUTE

Format: GO {<hex address>}

The GO command executes a program at an absolute memory address or reenters an existing program in memory. When there is no argument, execution begins immediately after the task control block. This is equivalent to the first instruction of a program loaded by PDOS with an entry address of relocatable zero.

If an argument is used, then execution begins at the specified <hex address>. The user workspace is not changed and is located at the beginning of the task control block.

```
.MDUMP 100,110
0064-0073  2F9C 595C 2F9C ....
.GO ,100,110
0064-0073  2F9C 595C 2F9C ....
.ALOAD XBUG
 *IDT=XBUG2.4
 *ABS ADR=>0070
LAST ENTRY ADR=>6000
.GO >6000
XBUG R2.4
?Z
.
```

4.12 HELP

Format: HE <parameter>

The HELP command provides error number explanations, tutorial guides to PDOS, user command parameter formats or definitions, utility program listings, disk usage instructions, or other textual messages associated with system software. HELP can be executed without destroying a BASIC or user program.

The HELP <parameter> is used to search a file named HLPTX on the system disk. All lines beginning with a non-blank character are matched against the <parameter>. If the <parameter> agrees, then all lines immediately following the keyword line that begin with a blank are printed. This continues until another line with a non-blank first character is encountered. If no match is found, the routine does not print anything and returns.

By following the above format, the user can create his own help files for each individual disk. This could include information on how to use the particular application programs on the disk.

```
.HE 1
 Syntax error
.HE HELP
 PDOS helps include: #       Error explanation
                     ERROR   Error number ranges
                     FILE    PDOS file helps
                     ASM     Assembler helps
                     JED     JEDIT commands
                     PDOS    PDOS user commands
                     <file>  Utility description
                     LINKER  LINKER commands
.HE PDOS
 PDOS resident commands are:

 AF - Append file        LM - Available memory
 BP - Baud port          LS - List directory
 CF - Copy file          LT - List tasks
 CS - Checksum           LV - Directory level
 CT - Create task        RC - Reset console
 DF - Define file        RN - Rename file
 DL - Delete file        RS - Reset disk
 EV - Set/Reset event    RT - Restore task
 EX - PDOS Basic         SA - Set attributes
 FS - File slots         SF - Show file
 GO - Execute            SP - Disk space
 HE - Help               ST - Save task
 ID - Init date          SU - Set spool unit
 IM - Interrupt mask     SY - System disk
 KT - Kill task          UN - Set output unit
.
```

4.13 SET SYSTEM DATE/TIME

Format: ID

The SET SYSTEM DATE/TIME command displays the PDOS header and prompts for the date and time. The PDOS header shows the PDOS system type and copyright declaration. Any delimiter can be used to separate date and time parameters. A carriage return <CR> leaves the old date and time.

```
.ID
PDOS/101 R2.4
ERII, COPYRIGHT 1982
DATE=MN,DY,YR 7,8,82
TIME=HR,MN,SC 10,30
.
```

4.14 SET INTERRUPT MASK

Format: IM <interrupt mask>

The SET INTERRUPT MASK command sets the system interrupt mask in the 9900 status register. The range of <interrupt mask> is from 4 to 15. Only the interrupt mask for the current task is set. All other tasks remain the same.

```
.LT
TASK  PAGE  TIME    TB    WS    PC    SR ..

*0/0   0     3    >6020 >619A >082A >1005..
.IM 10
.LT
TASK  PAGE  TIME    TB    WS    PC    SR ..

*0/0   0     3    >6020 >619A >082A >D00A..
.
```

4.15 KILL TASK

Format: KT {<task #>}

.KT 2

.

The KILL TASK command removes a task from the task list and returns the task's memory to the free pool for use by other tasks. Only your current task or a task spawned by your task can be killed.

Each task is assigned a unique task number which is shown by the LIST TASK command. Only the current task (indicated by '*') or those spawned by the current task (indicated by current task number following a "/" character) may be killed. Task #0 is the system task and cannot be killed.

If a minus (-) precedes the task number, then the task's memory is not deallocated to the memory bit map. If the task number is zero, then the current task is killed without deallocating memory. If no parameter is given, then the current task is killed and memory is deallocated.

.KT -2 Kill task # w/o freeing memory

.KT 0 Kill current task w/o freeing memory

All open files associated with the killed task are closed by the KT command.

.KT Kill current task and free memory

Example:

```
.LT
TASK  PAGE  TIME   TB    WS    PC    SR    BM    EM    CRU   PORT

*0/0   0     10  >6020 >629A >06BE >D20F >6000 >A000 >0080 >0001
 1/0   0     4   >8020 >829A >05E8 >220F >A000 >E000 >0180 >0002
.FS
SLOT  NAME         ST    SM    PT    SI    SE    BE    TN    BF

 32   TIB/4      >0100 >0098 >0000 >0000 >0052 >00F1 >0001 >0000
.KT 1
.LT
TASK  PAGE  TIME   TB    WS    PC    SR    BM    EM    CRU   PORT

*0/0   0     10  >6020 >629A >06BE >D20F >6000 >E000 >0080 >0001
.FS
SLOT  NAME         ST    SM    PT    SI    SE    BE    TN    BF
```

4.16 AVAILABLE MEMORY

Format: LM

The AVAILABLE MEMORY command shows you how many K bytes of memory are available for use in creating a new task.

```
.LM
FREE=32
.
```

4.17 LIST DIRECTORY

Format: LS {list string}

The LIST DIRECTORY command displays a selected list of disk file names including directory level, file name and extension, file type, file size, date of creation, and date of last update. Files are selectively listed according to disk number, file type, and/or directory level. The format of the {list string} is defined as follows:

```
.LS {<type>}{<level>}{/<disk>}
     AC       @       0-127
     BN
     BX
     EX
     OB
     SY
     TX
     UD
```

```
.LS {file type}{protection}{level qualifier}{/disk #}

 {file type} = AC  Assign Console file
               BN  Binary file
               BX  PDOS BASIC token file
               EX  PDOS BASIC file
               OB  TI9900 object file
               SY  System file
               TX  Text file
               UD  User defined

{protection} = *   Delete protected
               **  Delete and write protected

{level qualifier} = #  List all files on level #
                    @  List all files

{/disk #} = disk number ranging from 0 to 127
```

Also displayed with each directory listing is the disk name, the number of files stored on the disk and the number of directory entries available. This information is useful in disk maintenance.

```
0-3     303A Floppy
4-7     3314 Winchester
8-11    210 Bubble
```

(4.17 LIST DIRECTORY continued)

The directory entries are not necessarily in alphabetical order but in the order they are stored in the disk directory. If an alphabetical listing is desired, the ORDIR utility orders the directory or the DISCAT utility provides additional directory information in alphabetical order. (See 13.12 DISCAT and 13.45 ORDIR.)

Examples:

```
LS              List all files on current level & disk
LS 2            List all files on level 2 of current disk
LS @            List all files on current disk
LS EX@/5        List all 'EX' files on disk 5
LS OB**1/4      List all write protected 'OB' files
                   on level 1, disk 4
```

Example:

```
.LS
DISK NAME=PAUL #3MD/0                                     FILES=6/64
LEV  NAME:EXT       TYPE     SIZE      DATE CREATED      LAST UPDATE

 1   DOC01          TX *     31/34    15:02 03/03/81   10:13 03/07/81
 1   DOC02          TX *     92/95    12:37 01/05/81   10:21 03/07/81
 1   DOC00          TX *     72/95    12:36 01/05/81   10:11 03/07/81
.LS 0
DISK NAME=PAUL #3MD/0                                     FILES=6/64
LEV  NAME:EXT       TYPE     SIZE      DATE CREATED      LAST UPDATE

 0   JEDY           SY **    25/25    10:42 03/07/81   10:43 03/07/81
 0   $TTA                     1/1     11:12 03/07/81   11:12 03/07/81
.LS @
DISK NAME=PAUL #3MD/0                                     FILES=6/64
LEV  NAME:EXT       TYPE     SIZE      DATE CREATED      LAST UPDATE

 1   DOC01          TX *     31/34    15:02 03/03/81   10:13 03/07/81
 1   DOC02          TX *     92/95    12:37 01/05/81   10:21 03/07/81
 0   JEDY           SY **    25/25    10:42 03/07/81   10:43 03/07/81
 2   PRINTS         EX       21/21    12:40 01/05/81   15:49 03/05/81
 0   $TTA                     1/1     11:12 03/07/81   11:12 03/07/81
 1   DOC00          TX *     72/95    12:36 01/05/81   10:11 03/07/81
.LS @/1
DISK NAME=PAUL #30MD/1                                    FILES=11/64
LEV  NAME:EXT       TYPE     SIZE      DATE CREATED      LAST UPDATE

 1   ASM            SY **    43/43    09:50 02/27/81   09:51 02/27/81
 1   JEDY           SY **    25/25    09:51 02/27/81   09:51 02/27/81
 1   SYFILE         OB        3/4     20:14 02/26/81   20:28 02/26/81
 1   LIST           TX C     41/1000  15:42 02/27/81   15:42 02/27/81
....
```

4.18 LIST TASKS

Format: LT

The LT command displays to the your console all tasks currently in the task list. Task 0 is the system task and is created automatically during system initialization. This task cannot be killed.

Task listing

Your current task is indicated by an '*' preceding the task number. Following the task number is a slash and the parent task number. Subsequent data lists the current status of each task and is defined as follows:

```
*0/0   => current task
 1/0   => spawned task
```

```
TASK = Task # / parent task #, current = '*'.
PAGE = CRU memory page number.
TIME = Tics in CPU queue or suspension event
  TB = Task control block (R9).
  WS = Task Workspace Pointer.
  PC = Task Program Counter.
  SR = Task Status Register.
  BM = Beginning of task memory.
  EM = End of task memory.
 CRU = Task output port CRU base.
PORT = Task input port number.
```

Since BM, EM, CRU, and PORT are local parameters in the task control block, they are not listed for memory pages other than the current page.

Example:

```
.LT
TASK  PAGE  TIME   TB    WS    PC    SR    BM    EM    CRU   PORT

*0/0   0     10   >6020 >629A >06BE >D20F >6000 >8000 >0080 >0001
 1/0   0     4    >8020 >829A >05E8 >220F >8000 >8400 >0180 >0002
 2/0   0     4    >8420 >869A >05E8 >320F >8400 >8800 >0180 >0000
 3/1   1     4    >6020 >629A >05E8 >320F
 4/1   0     4    >8820 >8A9A >06BE >D20F >8800 >9000 >0180 >0000
.KT 1
.LT
TASK  PAGE  TIME   TB    WS    PC    SR    BM    EM    CRU   PORT

*0/0   0     10   >6020 >629A >06BE >D20F >6000 >8400 >0080 >0001
 2/0   0     4    >8420 >869A >05E8 >320F >8400 >8800 >0180 >0000
 3/1   1     4    >6020 >629A >05E8 >320F
 4/1   0     4    >8820 >8A9A >06BE >D20F >8800 >9000 >0180 >0000
.
```

4.19 DIRECTORY LEVEL

Format: LV {<level>}

The DIRECTORY LEVEL command displays or sets the current directory level used in directory listing and file definition. The disk directory is soft partitioned into 256 different groups, facilitating file maintenance. A soft partition means that any file is accessible from any current level.

The DIRECTORY LEVEL command without any argument displays the current directory level. A file defined without a specified directory level is defined on the current level.

If an argument is specified, it is converted to a number and sets the current directory level. The range is from 0 to 255.

```
.LV
LEVEL=3
.LS
DISK NAME=DISK #1/0                                     FILE
LEV  NAME:EXT       TYPE   SIZE    DATE CREATED   LAST

3    FILE1:SRC      OB      2/2   10:04 06/02/80 13:31
3    PROGRAM1:EXT   EX **  10/12  04:50 05/21/80 12:06

.LV 0
.LV
LEVEL=0
.LS
DISK NAME=DISK #1/0                                     FILE
LEV  NAME:EXT       TYPE   SIZE    DATE CREATED   LAST

0    $TTO           OB      1/1   12:35 05/21/80 01:02
0    MAKEBOOT       EX     18/18  09:15 05/22/80 12:02

.DF GAME2:GM;1
.LS @
DISK NAME=DISK #1/0                                     FILE
LEV  NAME:EXT       TYPE   SIZE    DATE CREATED   LAST

0    $TTO           OB      1/1   12:35 05/21/80 01:02
3    FILE1:SRC      OB      2/2   10:04 06/02/80 13:31
1    GAME2:GM       EX      0/1   05:25 06/08/80 05:25
0    MAKEBOOT       EX     18/18  09:15 05/22/80 12:02
3    PROGRAM1:EXT   EX **  10/12  04:50 05/21/80 12:06
```

4.20 RESET CONSOLE

Format: RC

The RESET CONSOLE command is used in an Assigned Console (type=AC) file to terminate the procedure and to revert back to the system console. This allows for a graceful termination of the file commands by closing the file and prompting for a new command. This is important when a task is assigned to the phantom port with a procedure file. If the RC command is not in the procedure file, the task will not kill itself when completed and suspend on the phantom event 95.

```
.
   .SF DO                List procedure file
   LV.SY
   RC
   .SA DO,AC            Set Assigned Console attribute
   .DO                  Invoke procedure file
   .LV.SY
   LEVEL=1
   .SY
   SYS DISK=0
   .RC                  Terminates command file
   .                    Waiting for new command
```

4.21 RENAME FILE

Format: RN <file1>,<file2>
RN <file1>,<level>

The RENAME FILE command changes the file name stored in the disk file directory. The RENAME command may also be used to move a file from one directory level to another. The file <file1> is renamed to <file2>. A disk specification in the second parameter is meaningless. If a number <level> is used instead of <file2>, then <file1> is moved to the new level.

```
.RN FILE1,FILE2
.RN TEMP,PROGRAM2
.RN PROGRAM2,4
.RN FILEN/2,FILEN:BK
.
```

4.22 RESET DISK

Format: RS {<disk #>}

Disk files must be closed at the end of any task so that sector buffers are flushed to the disk, pointers updated in disk directories, and file slots released for further usage. The RESET command either closes all open files associated with your task or closes all open files on a specified disk. The first mode allows your task to terminate itself without affecting the files of other tasks, while the second mode is used before withdrawing a disk from a disk drive.

```
.RS
.RS 2
.
```

RESET also clears the assigned console FILE ID (@>1E0(9)). However, the assigned console file may not be closed if the RESET disk option is used and the file resides on a different disk.

Assigned console reset

4.23 RESTORE TASK

Format: RT <file>

The RESTORE TASK command loads a binary task image as saved by SAVE TASK into user memory. Care must be taken to ensure the same task memory limits! (See 4.27 SAVE TASK.)

```
.RT DUMP1
.GO
```

4.24 SET FILE ATTRIBUTES

Format: SA <file>{,<attributes>}

The SET FILE ATTRIBUTES command associates file attributes with a file in the disk directory. File attributes include file types and protection flags. The "contiguous" attribute cannot be set but is removed with the 'SF' command.

File types are defined as follows:

AC -	Assign console. A file typed 'AC' specifies to the PDOS monitor that all subsequent requests for a console character will be intercepted and the character obtained from the assigned file.	.SA DO,AC	Batch process
BN -	Binary file. A 'BN' file type has no significance to PDOS but aids in file classification.	.SA OUTPUT,BN	Declare as binary data
OB -	9900 object file. All assembly user defined commands are typed as object files. When the file name is entered at the monitor prompt, PDOS loads the file into memory and executes the program.	.SA SPOOL,OB	Must be relocatable object!
SY -	System file. A 'SY' file is generated from an 'OB' file. TI9900 object is condensed into a smaller and faster loading format by the 'SYFILE' utility.	.SA CONFIG,SY	Must be condensed object!
BX -	PDOS BASIC binary file. A BASIC program stored using the 'SAVEB' command is written to a file in pseudo-source token format. Such a file requires less memory than the ASCII LIST format and loads much faster. Subsequent reference to the file name via the PDOS monitor automatically restores the tokens for the BASIC interpreter and begins execution.	SAVEB "PR:BIN"	File is type as "BX"
EX -	PDOS BASIC file. A BASIC program stored using the 'SAVE' command is written to a file in ASCII or LIST format. Subsequent file reference via the PDOS monitor automatically causes the BASIC interpreter to load the file and begin execution.	SAVE "PRGM1"	File is typed as "EX"

(4.24 SET FILE ATTRIBUTES continued)

TX - ASCII text file. A 'TX' type classifies a file as containing ASCII character text. Reference to the file name via the PDOS monitor causes the file to be listed to your console.

.SA LIST,TX   Declare text file

UD - User Defined. A 'UD' file type has no significance to PDOS other than file classification.

.SA PRGM2,UD

The file contiguous flag specifies the file store method. A file can be delete and/or write protected. These parameters follow the file type and are defined as follows:

C - Contiguous. The file is stored in contiguous logical sectors on the disk. This attribute is set during file creation and can only be deleted by using the SA command with a '#' attribute.

.SA DATA,#

* - Delete protect. The file is delete protected and cannot be deleted from the disk.

.SA DATA,*

** - Delete and write protect. The file cannot be deleted nor written to by any PDOS primitive.

.SA PROGRAM,**

All file attributes are cleared by omitting the attribute parameter. Since contiguous files are also linked files, the contiguous flag is removed by specifying a '#' attribute.

.SA FOBJECT   Clear all attributes

.SA CONTIG,#   Clear contiguous type

4.25 SHOW FILE

Format: SF <file name>

The SHOW FILE command displays on your console the disk file as specified by <file name>. The output may be temporarily interrupted at any time by striking any key. Output continues when another key is struck. The ESC key terminates the command at any time.

A 'TX' file type calls the 'SF' command when the file name is processed by the PDOS monitor.

```
.SF PRGM1              PRGM1 listed to CRT
 10  INPUT "N=";N
 20  PRINT " FACTORIAL=";FNFACT[N]
 30  GOTO 10
 100  DEFN FNFACT[I]
 110  IF I<=1: FNFACT=1: FNEND
 120  FNFACT=I*FNFACT[I-1]
 130  FNEND
.
```

4.26 DISK SPACE

Format: SP <disk #>

The DISK SPACE command displays the number of disk sectors free, the largest possible contiguous file, the number of disk sectors used, and the number allocated. All numbers represent decimal sectors. The total size in bytes is the number of sectors times 252.

The <disk #> specifies the disk number. If no parameter is used, then the default disk is displayed.

The FREE parameter shows the number of sectors still available for file storage. This is followed by the largest number of contiguous sectors. This is helpful in defining contiguous files.

The USED parameter shows exactly how much of the disk is truly used versus the amount of disk storage allocated. Some files may have END-OF-FILE markers pointing within the file and not at the end. If these files were copied to another disk, the unused storage would be recovered.

```
.SP
FREE=251,179
USED=7444/7749
.DF PAUL,180
PDOS ERR 55
.DF PAUL,179
.SP
FREE=72,42
USED=7623/7928
.DL PAUL
.SP
FREE=251,179
USED=7444/7749
.SP 6
FREE=39,34
USED=1779/1783
.
```

4.27 SAVE TASK

Format: ST <file>

The SAVE TASK command dumps a binary image of the user task memory to a file. This dump includes the task control block excluding the main workspace. The saved task is again restored by the RESTORE TASK command if the task memory limits have not changed. (See 4.23 RESTORE TASK.)

This command is useful in debugging programs or saving a long execution program so that it can be continued later.

```
STOP AT 40
BYE
.ST SAVE1
.EX
*READY
CONT
....

.RT SAVE1
.EX
*READY
CONT
....
```

4.28 SPOOL UNIT

Format: SU <unit>,<file>

The SPOOL UNIT command sets the spool unit and spool file ID variables in the task control block. Whenever the unit and spool unit variables have corresponding bits, then output is directed to the file specified by the spool file ID variable.

```
.UN 3
        ... Output to main and aux port
.UN 1
.SU 2,LIST
.UN 3
        ... Output to main port and file LIST
```

4.29 SYSTEM DISK

Format: SY {<disk #>}

The disk device identifier is contained within the file name. However, a default or system disk is assigned by the SY command. All file names without the disk identifier default to the system disk. This facilitates multiple users or programs which use temporary files independent of the disk configuration under which they operate.

```
.SY
SYS DISK=0
.SY 1
.SY
SYS DISK=1
.
```

4.30 CONSOLE UNIT

Format: UN <unit #>

The CONSOLE UNIT command sets the console output unit number. The unit number selects where the ASCII output is to be directed. Unit 1 is the system console CRT. Unit 2 is the auxiliary output number.

Each bit of the UNIT variable selects a different output device. Various bits can be assigned to different devices or files with the SU command.

```
.BP  -2,9600        Baud port 2 at 9600 for unit 2
.UN 3               Output to units 1 and 2 (1+2)
```

All further ASCII outputs out main port and AUX port at 9600 baud.