CHAPTER 7

PDOS I/O DRIVERS

PDOS I/O drivers are a natural extension of the PDOS file system. If a file name is preceded by a dollar sign, then the PDOS file manager expects the file to be an I/O driver program instead of data.

7.1 DRIVER ENTRY POINTS

PDOS I/O drivers are an extension of the PDOS file system. A I/O driver is designated by a dollar sign preceding a file name. I/O drivers contain position independent (self-relocating) code rather than data.

When an I/O driver is opened, closed, read from, written to, or positioned, the PDOS file manager branches into the channel buffer at specific entry points. This requires that the first ten bytes of the file be reserved for branch vectors and that the driver code be not more than 242 bytes in length.

The following are the driver entry points that must be at the beginning of each driver:

```
         RORG 0
>0000    DROP    JMP OPEN        ;OPEN FILE
>0002    DRCL    JMP CLOSE       ;CLOSE FILE
>0004    DRRD    JMP READ        ;READ FILE
>0006    DRWR    JMP WRITE       ;WRITE DATA
>0008    DRPS    JMP POSIT       ;POSITION FILE
```

The driver must be written in position independent or self-relocating TI9900 assembly code. This simply means that while the code is relocatable, there can be no relocatable tags within the object file.

A common way to make the code self-relocating is to generate a base address and then reference each constant within the program as a displacement beyond the base address.

PDOS passes the base address of the driver buffer in register R10. All labels referenced by address should be defined as the label minus a buffer beginning label plus four. The former makes the label absolute (REL-REL) and the latter skips the file links. Thereafter, all references to the label within the driver are made as displacements beyond register R10.

Extension of PDOS file system

$TTA

Maximum length = 242 bytes

Driver entry points

Position independent code

```
        RORG 0
DROP    ....

        BL @SUB(10)       ;CALL SUBROUTINE
        MOV @C1(10),R1    ;LOAD R1 WITH C1
        ....

SUB     EQU $-DROP+4      ;SUBROUTINE
        LI R1,NUMBER
        ....
        RT
*
C1      EQU $-DROP+4
        DATA 1            ;CONSTANT 1
```

7.2 DRIVER REGISTER USAGE

The PDOS file manager passes all parameters in registers to I/O drivers. A driver uses any of the registers except the link registers R13, R14, and R15. Register R11 contains the return address and must also be preserved. Only level 3 and 4 PDOS primitive calls are allowed in PDOS I/O drivers.

The driver exits via a 'INCT R11' and 'RT' for a normal return and a 'RT' only for an error return. If an error occurs, the error number is returned through register R13 to register R0 of the calling workspace (MOV R0,*R13).

If the driver alters constants within the buffer, then the file altered bit must be set in the file slot so that the buffer is correctly restored when rolled to the disk. This is done by executing the two instructions 'LI R0,>8000' and by 'SOC R0,@12(8)'.

Register R10 points to the beginning of the buffer and can be used as a base register in referencing labels within the driver. A position independent label is defined as '$-DROP+4', where DROP is at relocatable address 0.

The following table describes register usage for each driver entry point:

OPEN	(R4)	= File slot status
	(R8)	= File slot address
	(R9)	= Task control block
	(R10)	= Driver buffer base
	(R11)	= Return address
CLOSE	(R4)	= File slot status
	(R8)	= File slot address
	(R9)	= Task control block
	(R10)	= Driver buffer base
	(R11)	= Return address
READ	(R4)	= File slot status
	(R5)	= Character count
	(R7)	= Buffer address
	(R8)	= File slot address
	(R9)	= Task control block
	(R10)	= Driver buffer base
	(R11)	= Return address

```
                  *     $COUNT        09/01/82
                  *
      01E6        U1C   EQU >01E6     ;UNIT 1
      0000'0000'        RORG 0
0000: 1007        DROP  JMP OPEN      ;0 OPEN
0002: 100E        DRCL  JMP CLOS      ;2 CLOSE
0004: 101A        DRRD  JMP READ      ;4 READ
0006: 101C        DRWR  JMP WRIT      ;6 WRITE
0008: 0200 0046   DRPS  LI R0,70      ;8 POSITION
000C: C740        DRER  MOV R0,*R13   ;ERROR RETURN
000E: 045B              RT
                  *
0010: 04EA 0070   OPEN  CLR @CNT(10)  ;CLEAR COUNTER
0014: 0200 8000         LI R0,>8000
0018: EA00 000C         SOC R0,@12(8) ;FILE ALTERED
001C: 046B 0002   DRRT  B @2(11)      ;RETURN
                  *
0020: C1CB        CLOS  MOV R11,R7    ;SAVE RETURN
0022: 0201 0072         LI R1,MCNT    ;'COUNT='
0026: A04A              A R10,R1      ;ADD BASE
0028: 06AA 0056         BL @PRNT(10)  ;PRINT
002C: C06A 0070         MOV @CNT(10),R1
0030: 2FD6              XCBD          ;CONVERT #
0032: 06AA 0056         BL @PRNT(10)  ;PRINT
0036: 0467 0002         B @2(7)       ;RETURN
                  *
003A: 0200 0048   READ  LI R0,72      ;READ DEVICE
003E: 10E6              JMP DRER      ;  ERROR 72
                  *
0040: 0200 8000   WRIT  LI R0,>8000
0044: EA00 000C         SOC R0,@12(8) ;FILE ALTERED
                  *
0048: D037        WRIT2 MOVB *R7+,R0  ;DONE?
004A: 13E8                JEQ DRRT    ;Y
004C: 05AA 0070         INC @CNT(10)  ;N, COUNT
0050: 10FB              JMP WRIT2
                  *
      0056        PRNT  EQU $-DROP+4  ;PRINT (R1)
0052: D031        PRNT2 MOVB *R1+,R0  ;DONE?
0054: 130A                JEQ PRNT6   ;Y
0056: C329 01E6         MOV @U1C(9),R12
005A: 2FC0        PRNT4 XSWP          ;SWAP
005C: 1F16              TB 22         ;BUSY?
005E: 16FD                JNE PRNT4   ;Y
0060: 1D10              SBO 16        ;N
0062: 3200              LDCR R0,8     ;OUTPUT
0064: 1E10              SBZ 16
0066: 0605              DEC R5        ;DONE?
0068: 16F4                JNE PRNT2   ;N
006A: 045B        PRNT6 RT            ;Y
```

(7.2 DRIVER REGISTER USAGE continued)

```
        WRITE    (R4) = File slot status
                 (R5) = Character count
                 (R7) = Buffer address
                 (R8) = File slot address
                 (R9) = Task control block
                (R10) = Driver buffer base
                (R11) = Return address

     POSITION    (R4) = File slot status
                 (R8) = File slot address
                 (R9) = Task control block
                (R10) = Driver buffer base
                (R11) = Return address
                @4(13),@6(13) = Byte position
```

```
                         *
         0070       CNT  EQU $-DROP+4
  006C: 0000             DATA 0         ;COUNT
         0072      MCNT  EQU $-DROP+4
  006E: 0A0D             BYTE >0A,>0D
  0070: 434F 554E        TEXT +'COUNT='
  0074: 5430 0000
             0000'       END DROP
```

7.3 DRGN - DRIVER CONFIGURATOR

A system utility called DRGN generates a PDOS I/O driver from TI 9900 object code. The program resolves the TI 9900 object code into address-independent binary code , checking for illegal tags, such as relocatable data, and checks for a correct driver size.

DRGN - generate driver

The program operates as follows:

```
        .DRGN
        DRIVER GENERATOR R2.4
        SOURCE FILE=TTO:RB             9900 object file
        DRIVER SIZE=154 BYTES          Print driver size
        DRIVER FILE=TTO                Driver file
        .
```

If no driver file is defined, a file is created and the driver written to the file. DRGN error messages print the offending line followed by the error message.

Define file

Possible error messages include:

```
        CHECK SUM ERROR
        MEMORY SPACE EXCEEDED
        ILLEGAL TAG
        ** EXCEEDS CHANNEL BUFFER SIZE **
        NON-DRIVER FILE ONLY
```

7.4 RESTRICTIONS

The following are restrictions when adding I/O drivers to PDOS:

1) Driver must be written in self-relocating, address-independent TI9900 assembly language.

2) Driver entry points must be in the first ten bytes of the driver.

3) Driver cannot exceed the sector size less 4 link bytes. This results in a maximum length of 252 bytes.

4) A driver can only make level 3 and 4 PDOS primitive calls. This includes event, conversion, and swap primitives.

5) The link registers R13, R14, and R15 must be preserved.

6) The driver must be exited via R11. @2(11) is a normal return and *R11 is an error return. The error number must be moved to register R0 of the calling workspace.

7) Larger drivers can be written, but the excess code must be located elsewhere in memory.