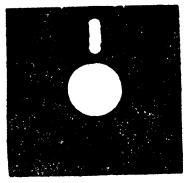


PDOS

PROGRAMMER'S REFERENCE MANUAL

BY:

PAUL R. ROPER

EYRING RESEARCH INSTITUTE INC
1455 West 820 North
Provo, Utah 84601
(801) 375-2434

Published by
EYRING RESEARCH INSTITUTE, INC.
1455 West 820 North
Provo, Utah 84601
(801) 375-2434

PDOS 2.4
PDOS PROGRAMMER'S REFERENCE MANUAL

Written by Paul Ross Roper

Please call (801) 375-2434 for more information.

PD 201-1020 REV D.

CHAPTER 1

INTRODUCTION

PDOS is a powerful multi-user, multi-tasking operating system developed by Eyring Research Institute, Inc., for the Texas Instruments compatible processor family. Chapter 1 is intended to give you a flavor of the operating system environment along with a glossary of terms used throughout this manual.

1.1 HOW TO USE THIS MANUAL

This manual is designed to be a comprehensive introduction to PDOS. It includes instructions for booting, testing, and trouble shooting the system, and covers all monitor commands, assembly primitives, and utilities. Examples and a full demonstration session is also provided. This is accompanied by an audio cassette demonstration tape to make it even more usable.

This manual

Each chapter is marked by a tab, with a table of contents for that chapter located at the tab. You may also find, at some tabs, appropriate summaries of the material in the chapter. These pages are supplementary to the text itself. Since they are not numbered, you may remove them from the binder and use for reference in any way convenient to you.

Tabs

Supplementary pages

You receive the most benefit from this manual if you first read through the table of contents for each chapter and then quickly scan the entire manual for an overview. This would be followed by a more detailed study of those chapters pertaining to your system. The examples to the right of the text are helpful in clarifing various concepts.

First, scan entire manual

This manual is organized in a top down manner: more general and less complex material is covered first. Specific chapter contents are as follows:

Organization of manual

Chapter 1 is an introduction to a PDOS system.

Introduction

Chapter 2 deals with system installation and start up procedures. This includes explanations on various hardware components likely to be found in a PDOS system.

Installation

Chapter 3 describes in detail the PDOS operation system: kernel, file manager, monitor, and floating point module.

PDOS system

Chapter 4 describes the monitor commands.

Monitor commands

Chapter 5 examines the assembly primitives of the PDOS kernel and file manager.

Assembly primitives

Chapter 6 lists the floating point XOP's and how they are used.

Floating point package

Chapter 7 shows how to use and created PDOS I/O drivers which are an extension of the file system.

I/O drivers

Chapter 8 gives a very detailed description of how to add new secondary storage devices to the PDOS boot EPROMs.

Secondary storage DSR's

(1.1 HOW TO USE THIS MANUAL continued)

PDOS BASIC

Chapter 9 covers PDOS BASIC, including a small BASIC primer and examples of more complex BASIC programs.

PDOS BASIC command summary

Chapter 10 is a reference chapter for all BASIC commands, functions, and statements.

Assembler, editor, linker, debugger

Chapter 11 is divided into assembler, editor, linker, and debugger sections.

Run module

Chapter 12 describes and gives examples on how to take your standalone applications and configure an EPROMable run module.

Utilities

Chapter 13 finishes with detailed descriptions of the more common PDOS utilities.

Appendices

The appendices give detailed descriptions of PDOS errors, driver listings, and command summaries.

Quick reference

This manual is written in two columns. The left hand column functions much as does the text of any book. The right hand column functions as an outline of the material in the left hand column plus addition examples and explanations. Use it for quick reference to specific topics.

Reply card

A reply card is also included for your use. While we have done our best to make this manual error free, we know that there will be mistakes, and would appreciate your help in making the next edition better than the current one. Please let us know any major mistakes or suggestions for chapters that need expansion.

Further reference

This manual assumes a moderate amount of computer hardware and software knowledge on your part. It also assumes familiarity for the TM990 board line and the TMS 9900 microprocessor. Such information is available in one or more of the following references:

Cannon, Don L. 1982. FUNDAMENTALS OF MICROCOMPUTER DESIGN - SYSTEM HARDWARE AND SOFTWARE. Dallas, Texas: Texas Instruments.

Texas Instruments Inc. 1979. INTRODUCTION TO MICROPROCESSORS - HARDWARE AND SOFTWARE. Houston, Texas: Texas Instruments.

Texas Instruments Inc. 1981. TM990/101MA MICROCOMPUTER USER'S GUIDE. Houston, Texas: Texas Instruments.

Texas Instruments Inc. 1981. TM990 MICROCOMPUTER CATALOG. Houston, Texas: Texas Instruments.

(1.1 HOW TO USE THIS MANUAL continued)

Texas Instruments Inc. 1978. TM990 POWER BASIC REFERENCE MANUAL. Houston, Texas: Texas Instruments.

Texas Instruments Inc. 1978. TMS 9900 MICROPROCESSOR DATA MANUAL. Houston, Texas: Texas Instruments.

Zarrella, John. 1981. MICROPROCESSOR OPERATING SYSTEMS. Suisun City, California: Microcomputer applications.

NOTATION

›	Hexadecimal number. (e.g., ›1FFF = decimal 8191.)
%	Binary number. (e.g., %0001111111111111 = decimal 8191.)
‹ ›	Parameter used with a PDOS command or primitive. (e.g., DL ‹file name› indicates that the DL command requires a file name as a parameter.)
{ }	Optional. (e.g., SA ‹file name› {,‹attributes›} indicates that the parameter ‹attributes› is optional.)
(Rx)	Indirect addressing. (e.g., (R2) = Buffer refers to register R2 pointing to a buffer.)
^	Control character. (e.g., ^C denotes a hexadecimal ›03 character.)

1.2 PDOS SYSTEM

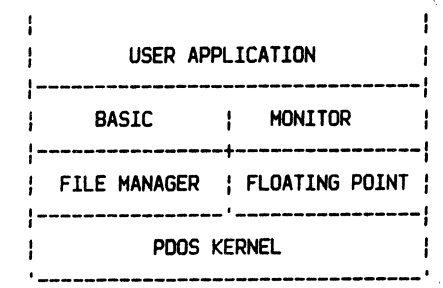
- Real-time, multi-user, multi-tasking
- Prioritized, round-robin scheduling
- Intertask communication and synchronization
- Paged or mapped extended memory modes
- Sequential, random, and shared file management
- Hardware independence
- 9900 layered design
- Complete floating point support
- Configurable, modular, ROMable standalone support

1.2.1 PDOS DESCRIPTION

PDOS is a powerful multi-user, multi-tasking operating system developed by Eyring Research Institute, Inc., for the Texas Instruments compatible processor family. You use PDOS to design and develop scientific, educational, industrial, and business applications.

Multi-user, multi-tasking

PDOS consists of a small, real-time, multi-tasking kernel layered by file management, floating point, and user monitor modules. The 2K byte kernel provides synchronization and control of events occurring in a real-time environment using semaphores, events, messages, mailboxes, and suspension primitives. All user console I/O as well as other useful conversion and housekeeping routines are included in the PDOS kernel.



The file management module supports named files with sequential, random, and shared access. Mass storage device independence is achieved through read and write logical sector primitives. The designer is relieved of real-time and task management problems as well as user console interaction and file manipulation so that efforts are concentrated on the application.

File management module

Assembly language floating point applications are no longer a problem. Conversion modules, assembler directives, and operating system calls allow easy integration of floating point operations into your application programs.

Floating point development

(1.2.1 PDOS DESCRIPTION continued)

PDOS is easily configured for any combination of large or small floppy disks, bubble memory devices, or Winchester mass storage devices. A wide variety of target system configurations are supported for fast development of memory-efficient, cost-effective end products.

Secondary storage

1.2.2 PDOS FUNCTIONAL DESCRIPTION

PDOS KERNEL. PDOS is written in 9900 assembly language for fast, efficient execution. The small kernel provides multi-tasking, real-time clock, event processing, and memory management functions. Ready tasks are scheduled using a prioritized round-robin method. Three XOP vectors are used to interface over 75 system primitives to a user task.

PDOS kernel

MULTI-TASKING EXECUTION ENVIRONMENT. Tasks are the components comprising a real-time application. Each task is an independent program that shares the processor with other tasks in the system. Tasks provide a mechanism that allows a complicated application to be subdivided into several independent, understandable, and manageable modules. Real-time, concurrent tasks are allocated in 1K byte increments.

Multi-tasking execution environment

INTERTASK COMMUNICATION and SYNCHRONIZATION. Semaphores and events provide a low overhead facility for one task to signal another. Events indicate availability of a shared resource, timing pulses, or the occurrence of a hardware or software interrupt. Messages and mailboxes are used in conjunction with system lock, unlock, suspend, and event primitives.

Intertask communication and synchronization

MEMORY REQUIREMENTS. PDOS is very memory efficient. The PDOS kernel, floating point module, file manager, and user monitor utilities require only 8K bytes of memory plus an additional 4K bytes for system buffers and stacks. Most applications are both developed and implemented on the target system. Further memory reduction is achieved by linking the user application to a 2K byte PDOS kernel for a small, ROMable, standalone, multi-tasking module. A fast, 6K byte scientific oriented BASIC interpreter with real-time primitives provides interactive high level language support as well. For large system configurations, PDOS effectively addresses up to a Megabyte of memory in either paged or memory mapped mode.

Memory requirements

(1.2.2 PDOS FUNCTIONAL DESCRIPTION continued)

File management

FILE MANAGEMENT. The PDOS file management module provides sequential, random, read only, and shared access to named files on a secondary storage device. These low overhead file primitives use a linked, random access file structure and a logical sector bit map for allocation of secondary storage. No file compaction is ever required. Files are time stamped with date of creation and last update. Up to 32 files can be simultaneously opened. Complete device independence is achieved through read and write logical sector primitives. Supported devices include floppies, bubble and battery back-up memories, Winchester disks, and streaming tape drives.

Command Line Interpreter

COMMAND LINE INTERPRETER. The PDOS monitor calls the command line interpreter. The CLI parses the command line for multiple commands and parameters. Utilities such as append, define, delete, copy, rename, and show file are resident and execute without destroying current memory programs. Other functions in the PDOS monitor include setting the baud rate of a port; checksumming memory; creating tasks; listing tasks, files and open file status; asking for help; setting file level, file attributes, interrupt mask, and system disk; and directing console output.

Interrupt management

INTERRUPT MANAGEMENT. The PDOS kernel handles user console, system clock, and other designated hardware interrupts. User consoles are interrupt driven with character type ahead. A task can be suspended pending a hardware or software event. PDOS switches control to a task suspended on an external event within 500 microseconds after the occurrence of the event (provided the system mask is high enough.) Otherwise, a prioritized, round-robin scheduling of ready tasks occurs on 8 millisecond intervals.

Portability

PORTABILITY. Software security exists throughout the 9900 product family (including 9940, 9980, 9995, and 99000). PDOS supports all TM990 products from Texas Instruments and Eyring Research Institute Inc., in addition to an expanding list of STD TMS9995 boards.

(1.2.2 PDOS FUNCTIONAL DESCRIPTION continued)

Customer support

CUSTOMER SUPPORT. Numerous support utilities including virtual screen editor, assembler, linker, macroprocessor, EPROMing, disk diagnostics and recovery, and disk cataloging are standard. Single stepping, multiple break points, memory snap shots debugger, task save and restore commands, and error trapping primitives in all high level languages are all provided to aid in program debugging. Free upgrades are available with hotline service to system developers. An optional modem service is provided for fast access to new products.

1.3 PDOS DEMONSTRATION

This section gives you a sample PDOS keyboard session. It is not intended as a start up procedure for new users, but rather, to give the flavor of the PDOS operating system environment.

All entries are terminated by a carriage return <CR> unless otherwise specified. Your entries are all underlined and indicated on those lines with a right bracket (>) in the left column.

Terminal session	Comments
> <CR>	Begin execution of the PDOS boot program via the RESTART.B vector at memory address >FFFC. A carriage return <CR> automatically sets your console port to the correct baud rate.
*PDOS BOOT R2.4	
0-99=BOOT	
100=MEMORY TEST	
101=IAC	
102=BOOT	
103=MAKE BOOT	
104=AUX	
> ?100,57274.....	System memory from >0000 to >DFBB is tested by writing and verifying random numbers throughout memory. A period indicates a complete memory pass without error.
> <RESTART.B><CR>	The RESTART.B vector must be used to stop the memory test.
*PDOS BOOT R2.4	
0-99=BOOT	
100=MEMORY TEST	
101=IAC	
102=BOOT	
103=MAKE BOOT	
104=AUX	
> ?<CR>	A <CR> selects a boot from the main disk.
BOOTED!	
> <CR>	
PDOS/101 R2.4	
ERII, COPYRIGHT 1982	
> DATE=MN,DY,YR 7,8,82	The PDOS banner prompts for date and time. Terminate all entries with a <CR> unless otherwise specified. Date and time numbers can be separated by commas or spaces. Seconds are optional.
> TIME=HR,MN,SC 10,30	
.	

(1.3 PDOS DEMONSTRATION continued)

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

  *0/0   0     3    >6020 >619A >0828 >1005 >6000 >E000 >0080 >0001
> .HE LT
   List Task headings explanation:
         TASK    Task # / spawned task #, current = '*'
         PAGE    CRU memory page number
         TIME    Tics in CPU queue or suspension event
         TB      Task control block pointer
         WS      Workspace pointer
         PC      Program counter
         SR      Status register
         BM      Beginning of task memory
         EM      End of task memory
         CRU     Task output port CRU base
         PORT    Task input port number
> .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
> .DFDSSDFK
  PDOS ERR 53
> .HE 53
    File Not Defined
```

'LT' lists the currently executing tasks.

The 'HE LT' command explains the 'LT' parameters. Notice that task memory begins at >6000 and ends at >E000. The task input port is console port #1 and task output is at CRU base >0080.

When the system first boots, only your task is executing, namely the system task (0).

The 'HE' command is non-destructive (won't affect any current programs) and is used to get explanations of PDOS commands, utilities, and error messages.

Current PDOS commands are listed.

PDOS errors range from 50 to 99.

Error descriptions are listed by 'HE' followed by the error number.

(1.3 PDOS DEMONSTRATION continued)

```
› .LV
  LEVEL=1
```

PDOS supports 256 directory levels for each disk number. Current level is 1.

```
› .SY
  DISK=4
```

The SY command lists the current disk number. PDOS supports up to 128 different logical disk numbers. Each disk number has its own directory and sector allocation bit map. Each disk number corresponds to some physical secondary storage device such as a disk or tape drive.

```
› .SP
  FREE=1185,1022
  USED=6753/6842
```

The SP command lists the current disk's FREE and USED sectors. Each sector corresponds to 256 bytes of data.

```
› .CS.RS.LM
  .RS.LM
  .LM
   FREE=0
```

Multiple commands are entered on the same line by separating commands with periods. Parameters are separated by commas. The command line echoes again for each new command.

```
  .EV
   >0000 >0000 >0000 >0000 >0000 >0000 >0000 >FFFF
```

Events are used for task synchronization. Each event is a single bit. The system events (96-127) are set.

```
› .LS
  DISK=WINCH #4/4                                        FILES=248/512
  LEV  NAME:EXT      TYPE      SIZE      DATE CREATED     LAST UPDATE
   1   ALOAD         SY         8/8      10:14 07/02/82   10:14 07/02/82
   1   ASM           SY        52/52     10:14 07/02/82   10:14 07/02/82
   1   BACKUP        SY         6/6      10:14 07/02/82   10:14 07/02/82
   1   BFIX          SY        11/11     10:14 07/02/82   10:14 07/02/82
   1   BURN302       SY        19/19     10:14 07/02/82   10:14 07/02/82
   1   BURNP         SY         9/9      10:14 07/02/82   10:14 07/02/82
   1   COMP          EX        19/19     10:14 07/02/82   10:14 07/02/82
   1   DDMAP         SY         9/9      10:14 07/02/82   10:14 07/02/82
   1   DDUMP         SY         7/7      10:14 07/02/82   10:14 07/02/82
       ......
```

The directory of any disk is listed to your console by the 'LS' command. The current disk and directory level are used if not specified in the command list. Each file is time stamped with date of creation and date of last update. The file size shows the number of sectors actually used versus the number of sectors allocated to the file from the sector bit map. Hitting any key gives a pause in the listing. Hitting another key continues the listing. The <escape> key terminates the output.

(1.3 POOS DEMONSTRATION continued)

```
> .HE FILES
  Valid file types are as follows:

  AC = Procedure file          C = Contiguous
  OB = 9900 object file        * = Delete protect
  SY = System file             ** = Write protect
  TX = ASCII text file
  BN = Binary file
  EX = BASIC program
  BX = BASIC binary program
  UD = User defined file
> .LS @/4
 DISK=WINCH #4/4                                        FILES=248/512
 LEV  NAME:EXT      TYPE     SIZE      DATE CREATED     LAST UPDATE

  0   $LPT          BN         1/1     10:13 07/02/82   10:13 07/02/82
  0   $TTA          BN         1/1     10:13 07/02/82   10:13 07/02/82
  0   $TTO          BN         1/1     10:13 07/02/82   10:13 07/02/82
  0   $TTS          BN         1/1     10:13 07/02/82   10:13 07/02/82
  10  ADUMP:SR      TX        10/10    10:09 07/02/82   10:09 07/02/82
  5   ADV:DAT       BN C    206/206    10:25 07/02/82   10:26 07/02/82
  5   ADVENT        SY        68/68    10:26 07/02/82   10:26 07/02/82
  1   ALOAD         SY         8/8     10:14 07/02/82   10:14 07/02/82
  2   ALOAD:SR      TX        43/43    10:17 07/02/82   10:17 07/02/82
  10  ART:SR        TX        23/23    10:06 07/02/82   10:06 07/02/82
  1   ASM           SY        52/52    10:14 07/02/82   10:14 07/02/82
      ......
```

The POOS monitor uses the file type in controlling file processing. A file typed as 'OB' contains TI tagged object and is relocatably loaded into task memory and executed; similarly with 'SY' files. 'EX' files are directed to the resident BASIC interpreter, loaded and and executed.

All files on disk number 4 (use your own disk number) are listed by the 'LS @/4' command. The name of the disk listed here is 'WINCH #4'. The current number of files in the disk directory and the directory size are also listed on the same line.

(1.3 PDOS DEMONSTRATION continued)

```
> .SF UPTIME
```

You display a file to your console with the 'SF' or Show File command. 'UPTIME' is an 'EX' or BASIC program.

```
  100  REM UPTIME
  110  DIM D[1],M[2],T[1],W[2]
  120  DATE $D[0]: TIME $T[0]: T=T1C 0
  130  M=$D[0]: D=$D[0;4]: Y=$D[0;7]: C=19
  140  M1=M-2: IF M1<1: M1=M1+12: Y=Y-1: IF Y<0: C=C-1
  150  W=INT[2.6*M1-0.19]+D+Y+INT[Y/4]-2*C+INT[C/4]
  160  W=INT[W-INT[W/7]*7+0.5]: IF W<0: W=W+7
  200  RESTORE W+1: READ $W[0]
  210  DATA "Sunday","Monday","Tuesday","Wednesday"
  220  DATA "Thursday","Friday","Saturday"
  230  RESTORE M: READ $M[0]
  240  DATA "January","February","March","April"
  250  DATA "May","June","July","August","September"
  260  DATA "October","November","December"
  300  PRINT "Today is ";$W[0];", ";$M[0];D;",";C*100+Y;
  310  PRINT ".  The time is ";$T[0];"."
  320  DAY=INT[T/10800000]: T=T-DAY*10800000
  330  HRS=INT[T/450000]: T=T-HRS*450000
  340  MIN=INT[T/7500]: T=T-MIN*7500
  350  SEC=INT[T/125]
  360  PRINT "PDOS has been up for";
  370  IF DAY: PRINT DAY;" days,";
  380  IF HRS: PRINT HRS;" hours,";
  390  IF MIN: PRINT MIN;" minutes, and";
  400  PRINT SEC;" seconds.";
  410  BYE
> .UPTIME
```

'UPTIME' is executed by typing the file name.

```
  Today is Wednesday, July 7, 1982.  The time is 10:45:16.
  PDOS has been up for 17 minutes, and 55 seconds.
> .HE BP
  Format: BP <port>,<rate>[,<base>]

   Where  <port>=1-8  (- sets UNIT 2 base)
          <rate>=110,300,600,1200,2400,4800,9600,19200
          <base>=new 9902 CRU base
```

You change or set port baud rates with the 'BP' command. The range is from 110 to 19200 baud. Each port is initialized to output only at 1200 baud on power up. If another baud rate is required or the port is to be used for input, then the 'BP' command must be used.

```
> .BP 2,1200.IM 5
  .IM 5
```

Port 2 is bauded for input and output at 1200 baud. The CPU interrupt mask is set high enough (using IM) to allow character interrupt inputs from port 2.

```
> .CF UPTIME,$ITA
```

The CF command is used to copy the file

(1.3 PDOS DEMONSTRATION continued)

```
> .BP 2,19200
```

You can change port baud rates at any time.

```
> .CT ,16,,2
  TASK #4
```

A new task (or user) is created by the 'CT' command. The task number is assigned by PDOS. The new task is 16K bytes in size and is assigned as task number 4. Port 2 is used for any task I/O.

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

  *0/0   0     3    >6020 >619A >0828 >C005 >6000 >A000 >0080 >0001
   4/0   0    -97   >A020 >A19A >075A >C405 >A000 >E000 >0180 >0002
```

A summary of the current tasks is listed by the 'LT' command. Since there is no free memory available, memory for the new task is taken from the spawning task's memory. The task time lists as '-97' because the task is asleep pending an input character from port 2.

```
> .LV 99
> .LS
  DISK=WINCH #4/4                                  FILES=248/512
  LEV  NAME:EXT      TYPE      SIZE      DATE CREATED    LAST UPDATE
```

Directory level 99 is selected.

```
> .DF DEMOF,10
> .LS
  DISK=WINCH #4/4                                  FILES=249/512
  LEV  NAME:EXT      TYPE      SIZE      DATE CREATED    LAST UPDATE

   99  DEMOF           C       1/10    10:45 07/08/82  10:45 07/08/82
```

A contiguous file named 'DEMOF' is created. Ten sectors are initially allocated. (This is optional if a non-contiguous file is acceptable.)

```
> .@CF TEMP,DEMOF
> .LT
  TASK  PAGE  TIME    TB    WS    PC    SR    BM    EM   CRU   PORT

  *0/0   0     3    >6020 >619A >0828 >C005 >6000 >9C00 >0080 >0001
   4/0   0    -97   >A020 >A19A >075A >C405 >A000 >E000 >0180 >0002
   5/0   0     1    >9C20 >9DDA >FFCC >9005 >9C00 >A000 >0000 >0000
```

Another way to create a task is with the '@' command. If the command line is preceded by '@', then a new task is created defaulting to 1K of memory and no console port assignment (phantom port). The new task copies, in background, the file 'TEMP' into 'DEMOF'. The task automatically kills itself when completed since it has no input port.

```
> .HE FS
  File Slot usage explanation:
        SLOT    File slot #
        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 EOF sector
        BE      # of bytes in EOF sector
        TN      Task number which opened file
        BF      Buffer pointer
        LF      Lock flag
```

The current files open under PDOS are monitored by the 'FS' command. The file name along with assigned file slot number and parameters are listed.

(1.3 PDOS DEMONSTRATION continued)

```
> .FS
  SLOT NAME              ST    SM    PT    SI    SE    BE    TN    BF    LF

    1  TEMP/4          >0A04 >11B0 >2510 >0009 >001A >006D >0005 >2410 >0000
   32  DEMOF/4         >4104 >1B74 >2614 >0009 >0036 >0030 >0005 >2610 >0000

> .EDIT
  EDIT R2.3x
> *ISTART XPMC            ;OUTPUT MESSAGE
>         DATA MES01
>         XEXT            ;EXIT BACK TO PDOS
> *
> MES01   BYTE >0A,>0D    ;CRLF
>         TEXT 'IT WORKS!!'
>         BYTE 0
>         END START
> $$
> *T$$
  START   XPMC            ;OUTPUT MESSAGE
            DATA MES01
          XEXT            ;EXIT BACK TO PDOS
  *
  MES01   BYTE >0A,>0D    ;CRLF
          TEXT 'IT WORKS!!'
          BYTE 0
          END START

> *GODEMOF$P$GO$H$$
> .ASM DEMOF,#DEMOF:OBJ,#LIST
  ASM R2.4
  SRCE=DEMOF
  OBJ=#DEMOF:OBJ
  LIST=#LIST
  ERR=
  XREF=

  END OF PASS 1
  0 DIAGNOSTICS
  END OF PASS 2
  0 DIAGNOSTICS
```

Assembly language development is very easy. Since it is hard to demonstrate the screen editor on paper, the character edit is used in this example. The '$' is echoed when the <escape> character is entered. This program outputs a message and then exits back to PDOS.

Notice that the assembler supports all PDOS primitive calls.

The file is now ready to be assembled by the utility 'ASM'. The object is written to a new file called 'DEMOF:OBJ'. The '#' in front of the file name tells PDOS to define the file if it is not already defined. A listing of the assembled program is written to a file called 'LIST'.

The list file is displayed to your console with the Show File 'SF' command.

(1.3 PDOS DEMONSTRATION continued)

```
> .SF LIST                                                    Show list file.
                                                    PDOS ASM R2.4
  PAGE: 1               11:03  07/07/82             FILE: DEMOF,WINCH #4

  1  0000: 2F5B         START XPMC          ;OUTPUT MESSAGE
  2  0002: 0006'              DATA MES01
  3  0004: 2FC5               XEXT           ;EXIT BACK TO PDOS
  4                     *
  5  0006: 0A0D         MES01 BYTE >0A,>0D   ;CRLF
  6  0008: 4954 2057 4F52     TEXT 'IT WORKS!!'
     000E: 4B53 2121
  7  0012: 0000               BYTE 0
  8  0013:      0000'         END START

                                                    PDOS ASM R2.4
  PAGE: 2               11:03  07/07/82             FILE: DEMOF,WINCH #4

  18 SYMBOLS
  MES01  R 0006 R0     A 0000   R1     A 0001   R10    A 000A
  R11    A 000B R12    A 000C   R13    A 000D   R14    A 000E
  R15    A 000F R2     A 0002   R3     A 0003   R4     A 0004
  R5     A 0005 R6     A 0006   R7     A 0007   R8     A 0008
  R9     A 0009 START  R 0000
> .SA DEMOF.TX                                               The file attributes are automatically set
> .LS                                                        by the assembler on the object file.  The
  DISK=WINCH #4/4                                FILES=250/512  source file, however, requires the 'SA'
  LEV  NAME:EXT      TYPE      SIZE      DATE CREATED    LAST UPDATE  command to set the text attributes (TX).

   99  DEMOF         TX C       1/10     10:45 07/08/82  10:45 07/08/82
   99  DEMOF:OBJ     OB         1/1      10:45 07/08/82  10:45 07/08/82
> .DEMOF:OBJ                                                 Finally, you execute the new program
  IT WORKS!!                                                 by entering the file name.
> .LT
  TASK  PAGE  TIME    TB    WS    PC    SR    BM    EM   CRU   PORT

  *0/0   0     3    >6020 >619A >0828 >C005 >6000 >A000 >0080 >0001
   4/0   0    -97   >A020 >A19A >075A >C405 >A000 >E000 >0180 >0002
> .KT 4                                                      You terminate a task with the 'KT' command.
> .LT
  TASK  PAGE  TIME    TB    WS    PC    SR    BM    EM   CRU   PORT

  *0/0   0     3    >6020 >619A >0828 >1005 >6000 >E000 >0080 >0001
  .
```

1.4 GLOSSARY

ASCII Literal	ASCII literals create special characters within strings that normally cannot be represented by a single printable character. An ASCII literal is composed of two hex characters within angle brackets.	ASCII Literal
Assembler	A language translator that translates ASCII text into machine code. The input language translates one text line into a single machine instruction.	Assembler
Auto Baud	Auto bauding is a technique used to set a port baud rate by timing the character length using a software program. A single character of a known bit pattern is used.	Auto Baud
Bit Map	A data structure utilized by PDOS for both memory and file space allocation. A single bit in the memory bit map is associated with each block of memory in the system. Likewise, each sector on a logical disk device is associated with a single bit in the sector bit map on the disk header. A 'one' indicates the corresponding sector is allocated, and a 'zero' indicates that the corresponding sector is free.	Bit Map
Blocked	Another term for the suspended task state.	Blocked
Buffer	A temporary block of memory, usually used for message and I/O transfers.	Buffer
Command Line Interpreter	The Command Line Interpreter is a small system software module which parses a line for commands and parameters. The CLI is called by the PDOS monitor.	Command Line Interpreter
Compiler	A language translator that translates the text of a high level language into assembly or machine code.	Compiler
Concurrency	Processes or tasks whose execution overlaps in time. They may be interacting or independent.	Concurrency

(1.4 GLOSSARY continued)

Contention	A situation that occurs when more than one task vies for a single resource.	Contention
CRC	An abbreviation for Cyclic Redundancy Code, an error checking technique that provides a high degree of error detection. It is often used for data transmission links and disk controllers, where burst errors are frequent.	Cyclic Redundancy Code
Create	A system service that initializes a structure by entering information such as its name, size, etc. into system tables. Specifically, PDOS supports task and file creation.	Create
Critical Code	A portion of software that accesses a shared resource and must be protected so that while one task is performing the access (executing the software), no other task is permitted to access the same resource. In most cases, either interrupts are disabled during the execution of this code or the task is locked.	Critical Code
Data Base	A large and complete collection of information that covers a variety of subject areas.	Data Base
Deadlock	A situation that occurs when all tasks within a system are suspended, waiting for resources that have already been assigned to other tasks that are also waiting for additional resources.	Deadlock
Debugger	A system software utility that aids a programmer in locating errors in his software. Functions usually include breakpoints, single stepping, memory inspect and change, disassembly, and assembly.	Debugger
Device	A unit of peripheral hardware such as a printer, terminal, or disk.	Device
Device Driver	A system software module that directly controls the data transfer to and from an I/O peripheral. PDOS device drivers are an extension of the file system.	Device Driver

(1.4 GLOSSARY continued)

Directory	A data structure containing entries for each file in the file system of a storage device. Each directory entry contains information about the file name, access rights, size, date of creation, and last update.	Directory
Disk number	A disk number is used by PDOS to reference a disk device. A single hardware device may be referenced by several disk numbers.	Disk number
DMA	An I/O processor memory access technique whereby the system processor is placed in a hold state while the I/O processor transfers data to or from memory, independent of the system processor and usually at the maximum memory data rate.	Direct Memory Access
Editor	A system utility that permits a programmer to create, modify, concatenate, or delete portions of files on a secondary storage device. Editors operate almost exclusively on text files. Types of editors include character, line, and screen editors.	Editor
End of File	A soft pointer to the end of "known" data within a file.	End of File
Entry Point	The programmer defined address at which a task begins executing.	Entry Point
Event	A condition used to synchronize task execution. An event may have a hardware or software origin. Hardware events result from processor interrupts. Software events are either user or system defined and are used to coordinate system tasks or resources.	Event
Execution Module	The Execution Module consists of the PDOS kernel plus other non-file oriented primitives. This object module is linked with user application tasks to form a ROMable, standalone program for the target processor. Other execution modules are also linked in for high level language support.	Execution Module

(1.4 GLOSSARY continued)

File	A collection of data, normally stored on a storage device such as a disk or tape.	File
File Attributes	File attributes are file status bits indicating the file type, disk storage method, and protection flags.	File Attributes
File Slot	A file slot is a logical I/O channel through which data transfers from an user application to secondary storage or other I/O device. The file slot maintains file status, pointers, and buffers.	File Slot
File System	System software modules that manage files on storage media. Functions include create, delete, rename, read, write, position, protect, etc.	File System
File Type	File types are attributes used by the PDOS monitor in determining how a file is processed.	File Type
First Fit	An algorithm for memory allocation that searches the free list (bit map) only long enough to find an unused memory block that is large enough to satisfy the memory request.	First Fit
Foreground/ Background	A condition within a multi-tasking operating system where critical programs operate in the foreground and execute with high priority while background assemblies, edits, listings, etc., are also going on at a lower priority.	Foreground/Background
Format	A system utility that initializes storage media with information necessary to assure that data can subsequently be read or written without error. This generally entails soft-sectoring disk tracks with address and ID marks which are detected by the hardware controller.	Format
Fragmentation	A condition where main memory or secondary storage is segmented due to dynamic memory allocation and deallocation.	Fragmentation

(1.4 GLOSSARY continued)

Friendly Environment	A software environment in which all software is adequately tested and therefore one task does not interfere with or cause errors in the execution of another task. The operating system cannot prevent intertask conflicts.	Friendly Environment
Garbage Collection	A system utility which reallocates or recovers system resources (such as fragmented memory) for further use.	Garbage Collection
Hard Error	An error which is predictable and repeatable.	Hard Error
High Level Language	A more sophisticated coding language than assembly language. One high level instruction generates many machine instructions. (e.g. FORTRAN, BASIC, PASCAL, etc.)	High Level Language
Hostile Environment	A system software environment in which it is assumed that both hardware and software may fail in any way, and the system is required either to continue running or shut itself down in an orderly manner.	Hostile Environment
In Circuit Emulation	A capability provided on many microcomputer development systems that enables a system designer to use the facilities of the development system to debug prototype hardware and software.	In Circuit Emulation
Index Table	A table utilized for reading and writing random access files with variable record sizes.	Index Table
Initialize	A disk is initialized such that PDOS parameters are available to the file manager. These include disk name, number of directory entries, total number of sectors available, date of initialization, density and sides flags, directory, and sector bit map. Any bad sectors are deallocated from user storage.	Initialize

(1.4 GLOSSARY continued)

Interleaving	A track formatting technique whereby multiple sectors may be read or written sequentially with a minimum of disk latency. This is possible by placing logical sectors on a track in such a way that the time required by the system service routine to process a single sector is less than the time required for the disk to rotate to the start of the next logical sector.	Interleaving
Interleave Factor	The number of physical sectors between a given sector and the next logical sector on a disk track.	Interleave Factor
Interpreter	A language translator that accepts high level language text and translates this text into a special intermediate code that is interpreted by a system program. Usually this intermediate code cannot be directly executed on a general purpose processor.	Interpreter
Interrupt	A signal from an external source that causes the processor to stop execution of the current task, save current task status, and begin executing a system service routine or another user task.	Interrupt
Interrupt Mask	A processor defined variable which limits interrupt levels.	Interrupt Mask
Interval Timer	A hardware clock which generates an interrupt after a specified period of time has elapsed.	Interval Timer
I/O Channel	See File Slot.	I/O Channel
ISAM	An Indexed Sequential Access Method for finding records within a file by means of a number or key in a separate file index.	ISAM
Kernel	The most basic portion of an operating system, usually supporting only task scheduling, communication, coordination, and memory allocation.	Kernel
Linked List	A data structure in which each element contains a pointer to its predecessor or successor (singly linked) or both (doubly linked).	Linked List

(1.4 GLOSSARY continued)

Linker	A system software utility that connects previously assembled/compiled tasks or subroutines into a single object module that can be loaded into memory for execution.	Linker
Loader	A system software utility that moves object code from secondary storage into memory, performing relocation as required.	Loader
Logical Device	A reference to an I/O device by name or number without regard to the exact nature of the I/O device.	Logical Device
Mailbox	A system data structure that handles task communication through global memory buffers.	Mailbox
Memory Bit Map	PDOS uses a memory bit map for memory allocation and deallocation in 1k or 4k byte increments. See Bit Map.	Memory Bit Map
Memory Mapped	A method of implementing system I/O through memory locations.	Memory Mapped
Monitor	A monitor is a set of resident utilities for handling the most common commands of the operating system.	Monitor
Multi-tasking	The ability of an operating system to permit multiple tasks to run concurrently.	Multi-tasking
Multi-user	The ability of an operating system to multi-task and allow multiple users complete system access.	Multi-user
Non-preemptive Scheduling	A scheduling algorithm where a task does not stop executing until it is complete.	Non-preemptive Scheduling
Object Code	The output of an assembler or compiler that can be loaded and executed on the target processor.	Object Code
Open	A system service which allocates a file or resource to a task.	Open

(1.4 GLOSSARY continued)

Operating System	A collection of system software that permits user written tasks to interface to the machine hardware and interact with other tasks in a straightforward, efficient, and safe manner.	Operating System
Overhead	The amount of processing time required by the operating system to perform housekeeping such as paging, swapping, and scheduling. Or, the amount of memory required by the operating system to maintain tasks.	Overhead
Overlay	A technique used to execute programs which are larger than the available memory size in systems without paging or segmentation capabilities.	Overlay
Page	An indivisible segment of memory which facilitates memory management.	Page
Parameter List	A parameter list refers to parameters following a command.	Parameter List
Phantom Port	A user port that has no physical device associated with it.	Phantom Port
Physical Device	A physical device is a hardware unit such as a disk or tape drive. The operating system binds a physical device to a logical device. User routines reference logical devices rather than physical devices.	Physical Device
Position Independent Code	Executable code which runs independent of the physical memory location at which it is loaded.	Position Independent Code
Preemptive Scheduling	A scheduling technique where task scheduling is independent of task completion. Round-robin swapping or high priority tasks can interrupt task execution at any time.	Preemptive Scheduling
Program Counter	A register within the processing element of a computer that contains the address of the next instruction to be executed. It is automatically incremented by the processor and modified by transfer instructions.	Program Counter

(1.4 GLOSSARY continued)

Queue	A data structure in which the first element in is the first element out.	Queue
Random Access	A type of file access in which data may be accessed in a random manner, regardless of its position within the file.	Random Access
Real Time	A type of operating system that supports online equipment having critical time constraints. Events must be handled promptly (i.e., within set timing limits).	Real Time
Real Time Clock	A system clock that indicates actual elapsed time from some reference time.	Real Time Clock
Record	A set of data elements that are logically accessed together.	Record
Reentrant Code	Code that may be executed simultaneously by more than one task. The code cannot be self-modifying and each task must maintain its own data area.	Reentrant Code
Resource	Assets of a computer system that the operating system uses and/or allocates to tasks for their use. These include memory, disk storage, printers, and terminals, as well as processors.	Resource
Response Time	The elapsed time from the entry of a command until its acknowledgement or completion.	Response Time
Retry	An attempt to provide automatic error recovery by executing the failed operation a second time.	Retry
Roll in/ Roll out	Roll in / Roll out functions refer to moving buffers or tasks to and from secondary storage when limited resources are available.	Roll in/Roll out
ROMable Code	Object code that is not self-modifying and uses workspace external to the code.	ROMable Code

(1.4 GLOSSARY continued)

Round-Robin Scheduling	A scheduling method where tasks in the Task List are executed in order, and entries into the list are always put at the end. Each task is given a time limit for execution and executes the full time unless blocked or a swap call is made to the operating system.	Round-Robin Scheduling
Scheduler	A system service that determines which task within the system should be run next.	Scheduler
Sector	The smallest contiguous storage area on a secondary storage medium. PDOS uses 256 byte logical sectors.	Sector
Sector Bit Map	PDOS uses a sector bit map on each secondary storage unit to allocate and deallocate logical sectors. See Bit Map.	Sector Bit Map
Sector Buffer	A buffer associated with a file slot for I/O transfers to and from secondary storage.	Sector Buffer
Semaphore	A "gating" variable that is used to synchronize task operations on shared data.	Semaphore
Sequential File Access	A type of file access where data may only be read or written sequentially, one record at a time.	Sequential File Access
Soft Error	A dynamic error normally caused by some transient condition. Retrying the failed operation often results in successful completion.	Soft Error
Source Code	Source code is ASCII text which is passed through a compiler or assembler to produce object code.	Source Code
Spawn	The spawn process generates a new task or entity. The new task is referred to as the spawned task.	Spawn
Static Priority	A task's execution priority is fixed either when the task is loaded or at time of system generation.	Static Priority

(1.4 GLOSSARY continued)

Status Register	A processor register containing the current executing conditions.	Status Register
Suspended	A task state in which task execution is discontinued pending the occurrence of an event.	Suspended
Swapping	The movement from one task to the next via the scheduler.	Swapping
Synchronization	The process of coordinating the execution of tasks within an operating system.	Synchronization
System Generation	The process of generating, linking, and loading all required system modules together in order to build a new operating system or to update tables in an existing system.	System Generation
System Service	Functions such as timekeeping, memory allocation, and console I/O that the operating system performs for user tasks upon request.	System Service
System Software	Software that is intimately associated with the operating system.	System Software
System Support	Functions or utilities such as language translators, debugging tools, diagnostics, and libraries which enable a system user or programmer to write and test tasks in an efficient manner.	System Support
Target Machine	The final machine on which a program is run.	Target Machine
Task Control Block	A Task Control Block is a block of memory containing the information needed by the operating system to schedule, suspend, and support a task. This includes workspace areas, buffers, port assignments, and other information necessary for the operating system to be reentrant.	Task Control Block
Task List	A system data structure containing a list of tasks within the system. This information includes the minimal amount of data required to suspend and resume task execution.	Task List

(1.4 GLOSSARY continued)

Task Lock	The process of locking a task in the run state such that no other task executes until an unlock task is done.	Task Lock
Task State	The status of a task (i.e., ready, executing, suspended or undefined).	Task State
Throughput	The quantity of information processed by a computer system in a unit time.	Throughput
Time Slice	The smallest time quantity available to the operating system for use in task scheduling.	Time Slice
Trace	A trailing record of a program's execution.	Trace
Unit	A logical gating variable which directs characters to various output destinations.	Unit
Utility	A software program supplied with the operating system which supports program development.	Utility
Wait	A system service that cases a task to be suspended for a specified time or pending the occurrence of an event.	Wait
Wakeup	The act of making a task ready to run after a period of suspension.	Wakeup
Workspace	The scratchpad area used by the central processor for calculations and temporary storage.	Workspace