Introduction to PDOS

Seminar Syllabus

PDOS

Realtime Operating System

Introduction to PDOS Training Seminar Syllabus

For use with PDOS 2.6e/f

Product number: 359Ø-1Ø/M Rev B

INTRODUCTION TO PDOS
TRAINING SEMINAR SYLLABUS

OVERVIEW OF SYLLABUS

This syllabus is arranged into four sections. They are as follows:

1. Overview and Class Agenda
2. Session Notes
3. Exercises
4. Sample Answers to Selected Exercises

OVERVIEW OF SEMINAR

This seminar is designed to meet your needs as a PDOS user whether you are a beginner at PDOS or an advanced user. It will be informal and your questions and comments throughout the seminar will be appreciated. The approach for most of the material will begin by discussing the subject and then follow the discussion by doing a 'hands on' exercise.

The first day, we'll concentrate on the development tools of PDOS and how to develop using the system. This day will be of most benefit to you if you have never used a PDOS system.

The second day, the PDOS internals, and high level language tools. The internals will introduce you to the format of the task control blocks, system variables, and the format of disks used by the file manager. This day will build upon the first day by providing greater detail to the functions of the PDOS system. If you are more experienced, you should gain helpful knowledge on how to best use PDOS. The high level languages will cover the unique features of the implemented languages on PDOS and some of the useful extensions to aid in programming under PDOS.

Finally, we'll concentrate on the advanced features of PDOS and how to build down to an application with run modules. Also, we will discuss how to customize a PDOS system along with implementing additional I/O devices. You will gain the most out of this day if you have had hardware and 68000 assembly experience.

At the conclusion of each day, questions and answers will be entertained. At the conclusion of the seminar, tours of Eyring will be available.

In conclusion, the seminar is a general introduction to the PDOS operating system and will not give you a comprehensive in-depth study of PDOS. We are planning future courses to address various aspects of PDOS and supported languages in greater detail.

Agenda

SEMINAR GOAL

1. Beginners should be able to develop on a PDOS system.
2. All should gain an understanding of development tools available.
3. Advanced users should understand where to begin on interfacing to PDOS.

AGENDA

DAY 1

8:ØØ	Continental Breakfast
9:ØØ	Session 1 -- PDOS Overview Session 2 -- Getting Started -The PDOS Monitor
1Ø:3Ø	Break
1Ø:45	Session 3 -- A Development Session -Edit
12:ØØ	Lunch
1:ØØ	Session 3 (con't) -Assemble -Link
2:ØØ	Session 4 -- Advanced Monitor Commands -Procedure Files
3:ØØ	Break
3:15	Session 5 -- Debug
4:ØØ	Session 6 -- Character I/O
5:ØØ	Open for Questions & Answers

DAY 2

8:00 Session 7 -- PDOS Tasking

10:30 Break

10:45 Session 8 -- Advanced PDOS File Manager

12:00 Lunch

1:00 Session 9 -- Languages
- BASIC
- C
- FORTRAN
- Pascal

1:30 Session 10 -- PDOS Internals

2:00 Session 11 -- Hardware Interface
- Task Device Service Routines
- BIOS Device Service Routines
- File Drivers

3:00 Break

3:15 Session 12 -- PDOS Run Modules

4:30 Session 13 -- PDOS System Generation and Installation

5:00 Open for Questions & Answers

ADDITIONAL SESSIONS

Session 14 -- PDOS Customer Services
Session 15 -- Tour of Eyring

Notes

SESSION 1 -- PDOS OVERVIEW

GOALS:

1. You will understand the design objectives of PDOS and the purpose of the operating system.
2. You will be introduced to the PDOS development package.

NOTES:

THE PDOS DESIGN OBJECTIVE

1. Develop on the target hardware.
   * No need for expensive development hardware.
   * No time lost in transferring from host to target.
   * You don't have to work with emulators.
   * You work with actual hardware.

2. Allow easy control and interface to hardware.
   * PDOS allows easy installation of hardware.
   * Hardware may either be controlled at task level or system level.

3. Use little EPROM space.
   * PDOS is small.
   * PDOS can be built down to use only the parts you need for your application.
   * PDOS is written in 68ØØØ assembly.

4. Be fast to provide realtime response.
   * Critical execution paths have been carefully optimized.
   * Low overhead in task switching.
   * Can be event driven with priorities.

THE PDOS DEVELOPMENT PACKAGE

1. Media
   * 5 1/4" disks standard
   * 8" disks
   * EPROMs

2. Documentation
   * PDOS Reference Manual.
   * User guides (future).
   * Installation guide.
   * Application notes.

3. Ready-to-Boot system
   * Bootable disk.

4. Licensed by CPU.
   * Runtime module licenses available separately.

SESSION 2 -- GETTING STARTED

GOALS:

1. You should learn how to boot PDOS on a system.
2. You should learn some basic monitor commands.

NOTES:

BOOTING PDOS

1. Most systems are auto boot; some require firmware boot commands.

2. Most will boot off floppy first, then try other boot locations.

3. Full installation may require EPROMS or setting jumpers. These are described in the installation guide for your hardware. Full installation will be discussed in a future session.

DO EXERCISE 2-1 -- BOOTING PDOS

THE PDOS MONITOR

1. Monitor syntax.

   # -- auto-create file. Preceeds filename string.

   . -- multiple command separator.

   () -- accept the enclosed argument as single argument.

   @ -- filename wildcard; match Ø or more characters.

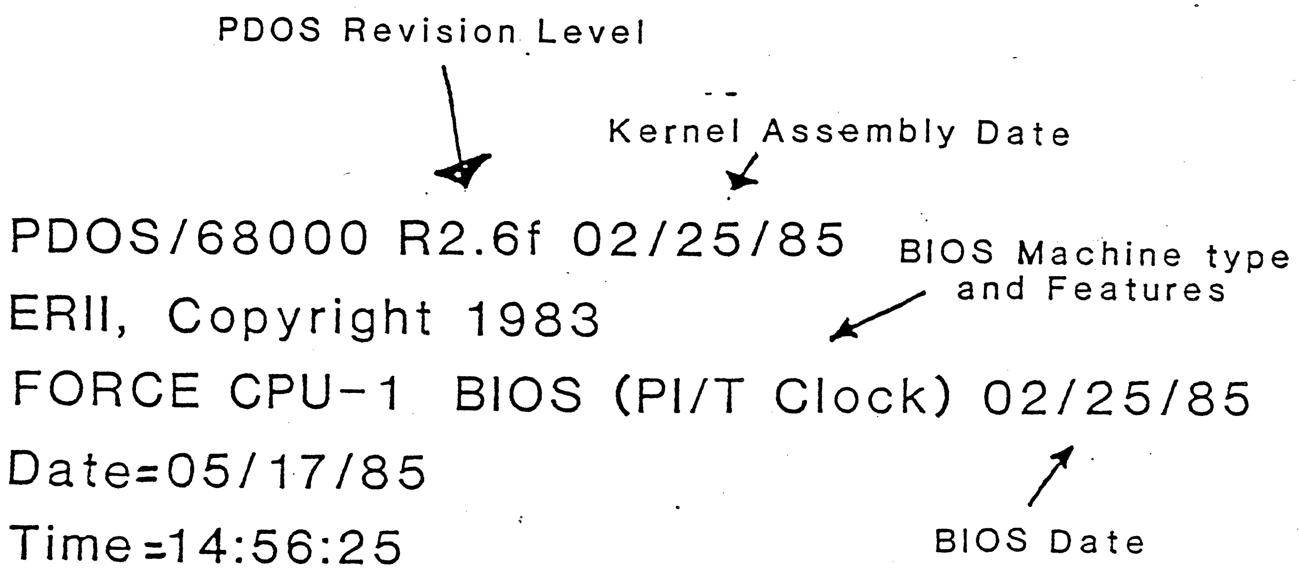
   * -- filename wildcard; match 1 character.

   > -- the monitor prompt.
   2,3>CC <ARG1>,...

2. Helpful Commands

```
HE -- help command
          >HE {<subname>}
ID -- PDOS ID/set date and time.
* MTIME P/B program.

DT -- date/time.
```

DO EXERCISE 2-2 -- HELP/ID



```
PDOS/68000 R2.6f 02/25/85
ERII, Copyright 1983
FORCE CPU-1  BIOS (PI/T Clock) 02/25/85
Date=05/17/85
Time=14:56:25
```

3. PDOS File Handling Commands.

```
AF -- append one file to the end of another
          >AF <source>,<dest>

CF -- copy from one file to another file.
          >CF <source>,<dest>

DF -- define a file.
          >DF <name{,<size>}

DL -- delete a single file.
          >DL <name>
```

DM -- delete multiple files (see wildcards).

```
        >DM <name>    -- Prompts with Y/N/A
```

MF -- make file by allowing text input from the keyboard [ESC] Exits.

```
        >MF <name>
```

RN -- rename a file.

```
        >RN <oldname>,<newname> or
        >RN <name>,<level>
```

SA -- set the PDOS attributes for a file.

```
        >SA <name>[,<attribute>}     -- ie: TX, SY, OB,
                                            AC, BN.
```

SF -- show on a terminal the contents of a file.

```
        >SF <name>
```

LS -- list directory of files. (see wildcards)

```
        >LS <name>    -- LS ;@ Lists all file levels on
        disk.
```

```
Lev  Name:ext     Type Size     Sect  Date created  Last update

1    NAME:1            5/5     Ø2ØD 14:12 Ø1/Ø5/84 14:13 Ø1/Ø5/84
```

4. The PDOS filename format:

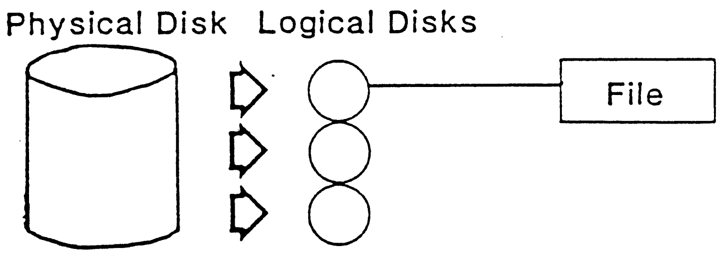
ACCCCCCC:CCC;LLL/DDD

A. Start with an alphabetic character.
B. Add up to seven additional characters.
C. Extension is denoted by colon; if colon present, must have one to three additional characters.
D. Levels are denoted by semicolon and number of level CAUTION: level is a sort key only and not part of filename.
E. Disk unit number denoted by slash followed by disk number.
F. If DISK and/or LEVEL is omitted, then default is assumed.

Table of Recommended File Names

None	EXECUTABLE file
:OBJ	OBJECT
:SR	SOURCE
:MX	S RECORD
:TMP	TEMPORARY
:C	C
:FOR	FORTRAN
:PAS	PASCAL

THE PDOS FILE MANAGER



- A physical disk may be broken down into many logical disks
- A single logical disk contains a directory of files.
- Filenames are 1 to 8 characters with a 3 character optional extension and level.
- Level is used for a work area on the disk (subdirectory).
- Filenames are NOT unique if only a different level number is used.

Directory

Lev	Name:ext
1	DAN:SR
2	DAN:SR

Not possible in standard PDOS.

5. PDOS Disk Commands.

```
LV -- Ø..255 levels. LS for levels
          LV  >LV {<level>}

SY -- Ø..255 disks Sets working disk number up to 4
          SY  >SY {<disk>{,<disk>..}}

SP -- amount of space available on disk.
          SP  >SP {<disk>}
          FREE=total free, largest contiguous sector
          USED=actual use/total allocated
```

6. In future exercises 'NAME' is to be your first name for exercise filenames.

DO EXERCISE 2-3 -- VALID FILENAMES

DO EXERCISE 2-4 -- FILE COMMANDS

DO EXERCISE 2-5 -- DISK COMMANDS

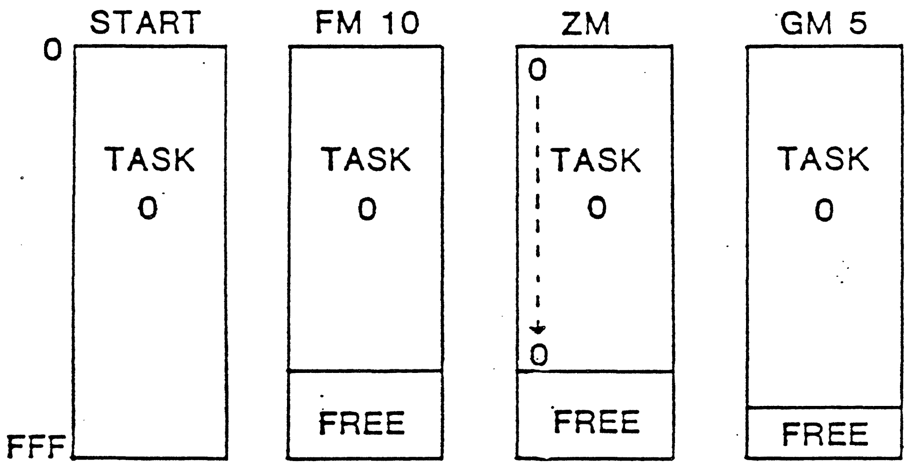
7. PDOS Memory Commands.

```
AM -- amount of memory free to task.
          AM >AM <file1>,<file2>

FM -- free memory from your task.
          FM >FM <kbytes>

GM -- return free memory to your task.
          GM >GM {<kbytes>}

ZM -- zero out your task memory.
          ZM >ZM
```

DO EXERCISE 2-6 -- MEMORY COMMANDS

MEMORY COMMANDS

GM AM FM ZM



AM – List memory adjacent to TASK

GM – Get available memory

FM – Free memory FM –n remove
memory from PDOS allocation map

ZM – Zero task's memory

Memory managed in 2K byte pages

8. Command Line Editing.
   * A line recall feature and editing features are also provided by PDOS.

   [ESC] -- ignore current line.

   ^C -- abort current line.

   ^A -- recall last line.

   ^F -- move right one character.

   ^H -- move left one character

   ^D -- delete character to the right.

   [RUB] -- delete character to the left.

   ^I -- insert mode.

DO EXERCISE 2-7 -- COMMAND LINE EDITING

SESSION 3 -- A DEVELOPMENT SESSION

GOALS:

1. You should learn how to use the PDOS Editor -- MJEDY
2. You should learn how to code, assemble, and link a program.

NOTES:

A. OVERVIEW OF DEVELOPMENT

1. Concept or need.

2. Algorithm or flowchart design.

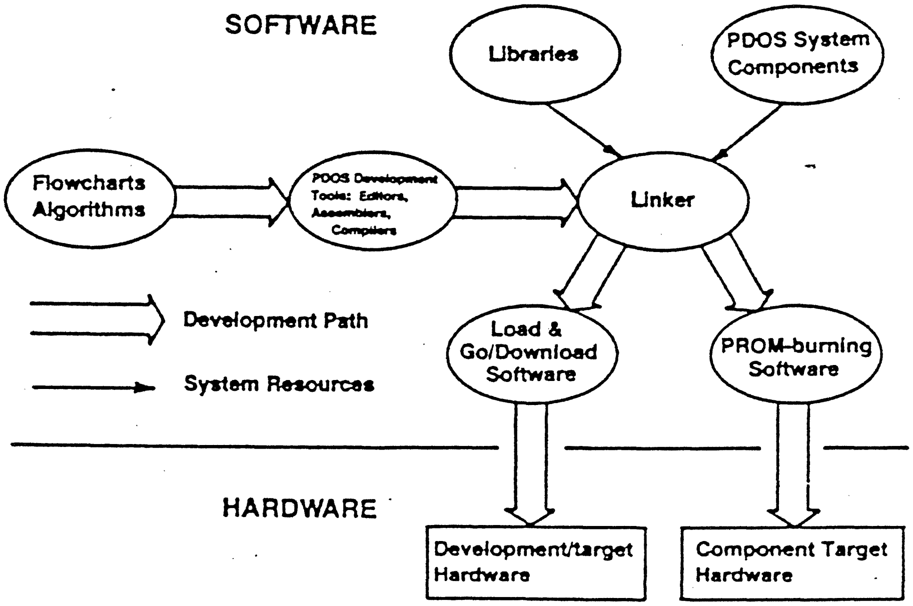
3. Selection of language.
   * BASIC
   * C
   * Pascal
   * FORTRAN
   * Assembly

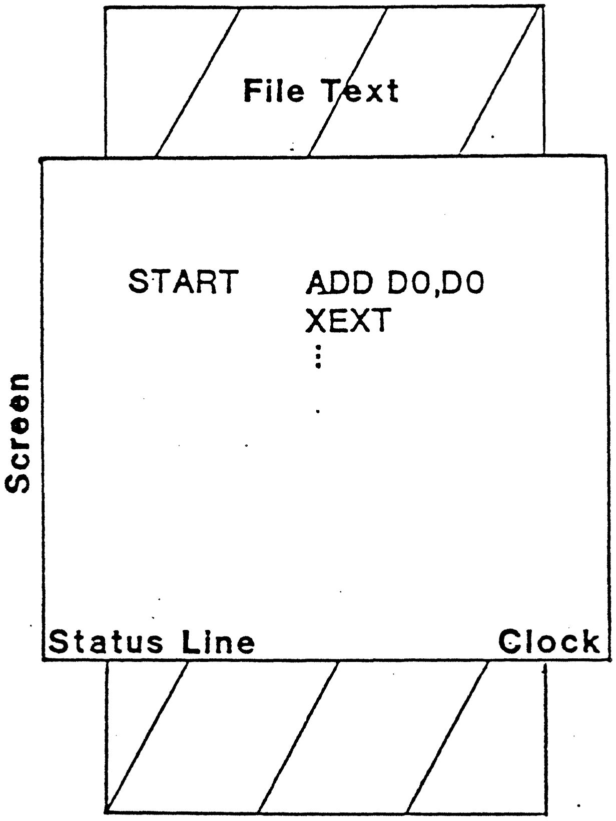
4. Write the program.

5. Assemble the program.

6. Link the program if other modules are involved.

7. Execution and debugging.


SOFTWARE
Libraries
PDOS System Components
Flowcharts Algorithms
PDOS Development Tools: Editors, Assemblers, Compilers
Linker
Development Path
System Resources
Load & Go/Download Software
PROM-burning Software
HARDWARE
Development/target Hardware
Component Target Hardware


File Text
START ADD D0,D0
XEXT
⋮
Screen
Status Line
Clock

* Basic Editing commands
  -- Use of [ESC] key
  -- Control Keys
  -- Help Key [ESC]^F
  -- Cursor movement

```
               ^K
                |
      ^H ------- ^L
                |
               ^J
```

  -- Top / End of text ^T / ^Z
  -- Auto-insert feature
  -- Rubouts [DEL] [^DEL]
  -- PDOS Reference Manual chapter 11-pages 4-12

DO EXERCISE 3-2 -- MJEDY HELP KEY

DO EXERCISE 3-3 -- BASIC EDITING

3. File Interaction.

* ^Write command -- ^Wfilename[ESC]V
* ^Get Command -- ^Gfilename[ESC]V
* On ^W and ^G pressing the ^W or ^G key twice recalls the last filename used.
* ^Quit Command -- ^QV
* ^New buffer Command -- ^NV
* GO from PDOS monitor

DO EXERCISE 3-4 -- FILE INTERACTION

4. Searches and Macros

* Search for a string from the cursor forward
  -- ^S<STRING>[ESC]

* Search for a string backward from the cursor
  -- ^B<STRING>[ESC]

* [TAB], [CR] can be entered in by typing key.

* Other control characters may enter by typing ^V then the control character to find. (The control chars are displayed by adding $2Ø to the char value).

* All search strings are terminated by a [ESC].

* The cursor is placed at the end of the string on a forward search, and at the start of the string on a backward search.

- * To continue search after a string is found, but before any editing is done, type [ESC] again.

- * ^Define a macro -- ^D

- * All commands and key strokes are remembered until ^D is typed again

- * ^Execute a defined macro -- ^E

- * [ESC]^ZN[ESC] execute macro N times (-1 = infinite or until a search fails). ^C will also interrupt a repetitive macro.

DO EXERCISE 3-5 -- GLOBAL SEARCH AND REPLACE

5. Restoring and saving of macros as disk files.

   * Output macro to file -- [ESC]^OFILENAME[ESC]V
   * Input macro from file -- [ESC]^YFILENAME[ESC]V
   * The creation and verification prompt is same as ^G and ^W.

DO EXERCISE 3-6 -- INPUT / OUTPUT OF MACROS

6. Advanced text editing with deletion and pointer commands.

   * ^] - Delete from cursor to EOL
   * ^^ - Delete from cursor to and including the EOL
   * ^P - Place pointer into text
   * ^Y - Insert text in a file at the cursor

THE FOLLOWING COMMANDS WORK WITH A POINTER:

   * ^O - Output to a file from cursor to pointer
   * ^U - Insert into 'up' (cut) buffer from cursor to pointer
   * ^\ - Delete text from cursor to pointer
   * ^A - Insert text at cursor that is in the up buffer
   * ^F - Move cursor to the pointer

NOTE: ALWAYS DELETE THE POINTER BEFORE YOU SAVE YOUR TEXT (THE POINTER IS DELETED AS ANY OTHER CHARACTER).

DO EXERCISE 3-7 -- BLOCK CUT AND PASTE

B. MASM

1. The MASM command line:
   * PDOS Reference Manual 13-5

   ```
   MASM <SOURCE>,{<OBJECT>,<LIST>,<ERROR>,<XREF>}
   ```

   SOURCE = The input text
   OBJECT = The tag object output that may be executed if fully resolved, or output for QLINK.
   LIST = Full listing of the program
   ERROR = List of errors. (Defaults to console if no file specified.)
   XREF = List of symbols and lines where used

NOTES:

A. If only MASM is typed, then you will be prompted for the filenames.

B. PDOS will not create files that do not exist unless you prefix the filenames with a '#'.

DO EXERCISE 3-8 -- RUNNING MASM

2. Program listing format.

   * Use SF command to type your listing on the screen and follow along.

   * Pages 11-16 cover format:

```
                                        68K PDOS Assembler 1Ø/17/84
PAGE: 1         13:42 12/11/84          FILE: DAN:SR,CLASSWORK


 1  Ø/ØØØØØØØØ:AØ8CØØ14        START   XPMC     MESØ1    ;ASK FOR
 2  Ø/ØØØØØØØ4:AØ8Ø                    XGLU
 3  Ø/ØØØØØØØ6:AØ56                    XCDB
 4  Ø/ØØØØØØØ8:C2FCØØ64                MULU.w   #1ØØ,D1
 5  Ø/ØØØØØØØC:AØ8CØØ1F                XPMC     MESØ2    ;X 1ØØ
 6  Ø/ØØØØØØ1Ø:AØ5Ø                    XCBD
 7  Ø/ØØØØØØ12:AØ8A                    XPLC
 8  Ø/ØØØØØØ14:6ØEA                    BRA.S    START
 9                             *
1Ø  Ø/ØØØØØØ16:ØAØD454E5445522Ø MESØ1  DC.B     $ØA,$ØD,'ENTER
11             594F5552204E554D
12             424552203D2ØØØ
13  Ø/ØØØØØØ2D:2Ø782Ø313Ø3Ø2Ø3D MESØ2  DC.B     ' x 1ØØ = ',Ø
14             2ØØØ
15  Ø/ØØØØØØ37:ØØ                      EVEN
16  Ø/ØØØØØØ38:      Ø/ØØØØØØØØ        END      START
```

```
                                     68K PDOS Assembler 1Ø/17/84
PAGE: 2        13:42 12/11/84        FILE: DAN:SR,STANDARDS

DEFINED SYMBOLS:

MESØ1   Ø/ØØØØØØ16   MESØ2   Ø/ØØØØØØ2D   START  Ø/ØØØØØØØØ

EXTERNAL DEFINITIONS: NONE

EXTERNAL REFERENCES: NONE

UNDEFINED SYMBOLS: NONE

UNREFERENCED SYMBOLS: NONE
```

3. External Linkage

* XREF -- Symbol is not in this file; look elsewhere at link time for it.
* XDEF -- Allow this symbol to be used in other files.

DO EXERCISE 3-9 -- EXTERNAL LINKAGE

C. QLINK

* PDOS Reference Manual Chapter 11, page 27, 52-69

1. QLINK Command summary

* HELP - display all QLINK commands
* INPUT - input a single object file
* ZERO - zero out the memory buffer
* LIBRARY - load in only those object files that are needed from a library file
* MAP - display the location and status of memory
* OUTPUT - select for output the memory buffer to a file
* SRECORD - specify srecord format in the output file
* SYFILE - specify binary image format inthe output file
* END - output the memory buffer to the file
* QUIT - return to the PDOS monitor

DO EXERCISE 3-1Ø -- USING QLINK

SESSION 4 -- ADVANCED MONITOR COMMANDS

OALS:

. You should learn how to use additional monitor commands.
. You should learn how to create procedure files.

OTES:

. Procedure Files

* Useful for repetitive command sequences.

* May be "programmed" to allow modification at execution of procedure file.

* Input is directed from a file instead of keyboard.

* Input will be directed to programs using XGCC or XGCR or line edit commands.

* Argument substitution.

  &Ø -- set by program control; usually a status number.

  &1..&9 -- corresponds to argument 1 then 9 of the command line.

  && -- treat character as a single ampersand.

* IF processor.
  = -- IF arguments are equal, do command to right of `.'
  # -- IF arguments are not equal, do command right of `.'.

* GT -- Goto a label string in the file allowing you to skip commands (often used with IF).

* IF-THEN-ELSE

```
IF &1=<string>.monitor then commands.GT ENDIF
monitor else commands
GT ENDIF
ENDIF
```

* Nested command files.
  -- 3 Deep.

* RC vs RS.
  RC -- reset the current procedure file.
  RS -- reset all files open to this task.

SESSION 5 -- DEBUG

GOAL:

You should understand some basic debug commands.

NOTES:

- * Resident debugger.
- * Allows for break points, traces, disassembly, dumping of memory and registers.
- * Re-entrant for tasks.
- * Enter by PB command

DO EXERCISE 5-1 -- DEBUG

- * The debug application note

DO EXERCISE 5-2 -- DEBUG Application Note

SESSION 6 -- CHARACTER I/O

GOALS:

1. You should understand character output redirection under PDOS.
2. You will understand how to baud a port.
3. You will learn how to print and create log files.

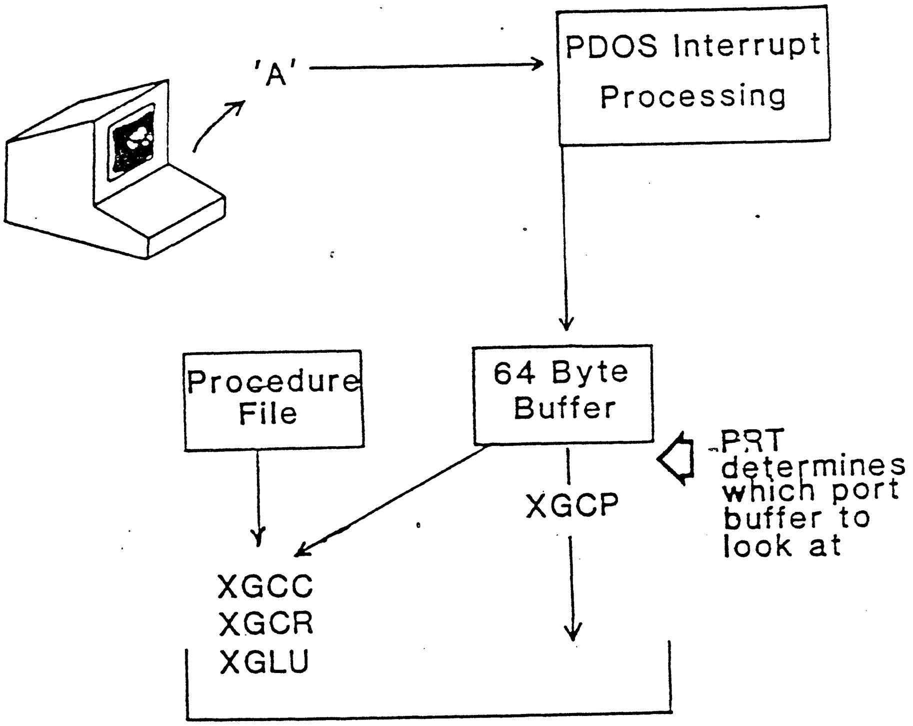
NOTES:

* PDOS Reference Manual Chapter 3

1. The Character Input Path.

* Only one task per input port.
* PRT -- location in TCB where input number is saved.

INPUT PATH


'A'
PDOS Interrupt Processing
64 Byte Buffer
Procedure File
XGCP
PRT determines which port buffer to look at
XGCC
XGCR
XGLU

User Program

2. The Character Output Path.

* More than one task may share output ports.

* What is a unit?
  A unit (in the context of character I/O) is a path on which a character is sent on output. It may be logically linked to a character port, your terminal screen or both. It may also go to a file or nowhere.
  - U1P, U2P

* What is a port / type?
  A port is the physical port on your hardware to which you link other hardware devices. The type is an indicator to PDOS to which output code to use.

* What is a spool?
  It redirects output from a port to a file.

* List task monitor command (LT) list out port assignments.

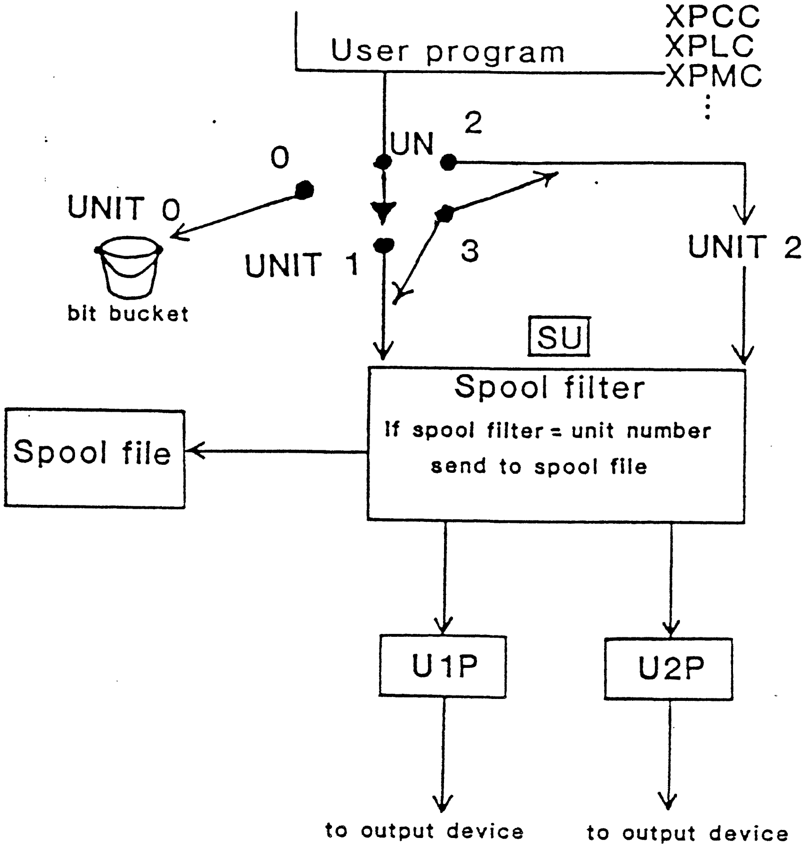
```
Task Prt Tm Event Map Size ........ PRT U1P U2P
*Ø/Ø 64  1          Ø   4Ø4  ........ 1/1 1/1 3/2
```

* The baud port monitor command (BP) sets up the physical port as well as U2P. Also allows to see port setup.

```
Port  Type  f_pi8dbs    Base     Rate   Task
#1     1    ØØØØØØØØ  FFFFCØ71  192ØØ  Ø
```

DO EXERCISE 6-1 -- LT AND BP COMMANDS

OUTPUT



3. I/O Redirection and Log Files

- UN command -- Allows for selection of output path
- SU command -- Directs path to file instead of port
- SF with -Filename -- Used to type files to terminal
- The output flow
- TTA, TTO, TTS driver files

DO EXERCISE 6-2 -- PRINT A FILE

SESSION 7 -- PDOS TASKING

GOALS:

1. You should understand multi-user and multi-tasking.

2. You should become acquainted with the task control block.

3. You should understand task synchronization.

NOTES:

* PDOS Reference Manual Chapter 3

1. The KERNEL

- * Multi-tasking, multi-user scheduling
- * System clock
- * Memory allocation
- * Task synchronization
- * Task suspension
- * Event processing
- * Character I/O processing

2. Tasks

* Creation

Monitor command

```
CT TASK,SIZE,TIME+PRIORITY,PORT
@
Background tasking
```

Father-son relationship

LT Command

```
Task Prt Tm  Event Map Size    PC     SR     TB      BM      EM     PRT U1P U2P
*Ø/Ø 64  1          Ø   368  ØØ233A 2ØØ4 ØØBØØØ ØØCC46 Ø67ØØØ   1/1 1/1 3/2
 1/Ø 64  1    96    Ø    32  ØØ2174 2ØØØ Ø68ØØØ Ø685ØØ Ø7ØØØØ   Ø/Ø 1/1 Ø/Ø
```

DO EXERCISE 7-1 -- BACKGROUND DEVELOPMENT TASK

Primitive

.XCTB

Startup table

* Termination

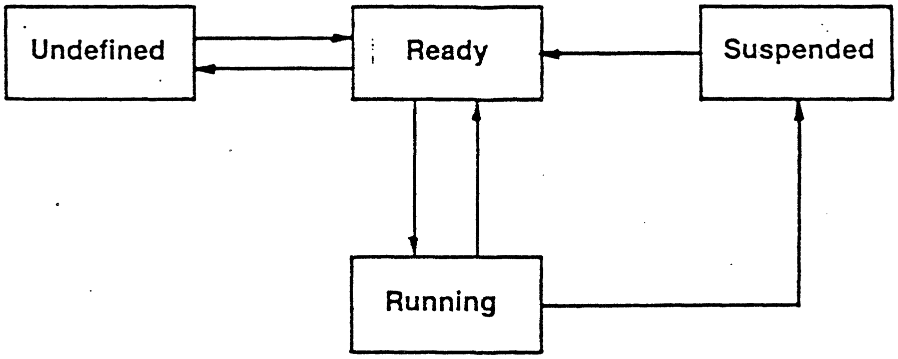
Monitor command
KT
Primitive
XKTB
Auto termination

Caution for Procedure files--MUST USE RC OR RS

DO EXERCISE 7-2 -- PROCEDURE FILES AS TASKS USING RC

* The task cycle

PDOS TASK CYCLE



1. Highest priority READY task always executes
2. Round robin with tasks of same priority

* The task control block

--PDOS Reference Manual Pages 3-6

DO EXERCISE 7-3 -- THE TASK CONTROL BLOCK

3. Inter-task Communication.

* Synchronization

* Events (page 3-12 and 3-13 of PDOS Reference Manual).

  GLOBAL Ø..127

  LOCAL 128

* Event primitives
  XSEF
  XSEV
  XTEF
  XSUI
  EV MONITOR COMMAND

DO EXERCISE 7-4 -- MULTI-TASK EVENT SYNCHRONIZATION

4. Task Priorities

* Highest ready priority always runs. 1 is lowest; 255 is highest.

* The TP command.

* Task lock / unlock.

DO EXERCISE 7-5 -- TASK PRIORITIES

5. Message Buffers

* SM & KM Commands
  SM -1 to father task

* XGTM on each prompt display.

DO EXERCISE 7-6 -- MULTI-TASK MESSAGES

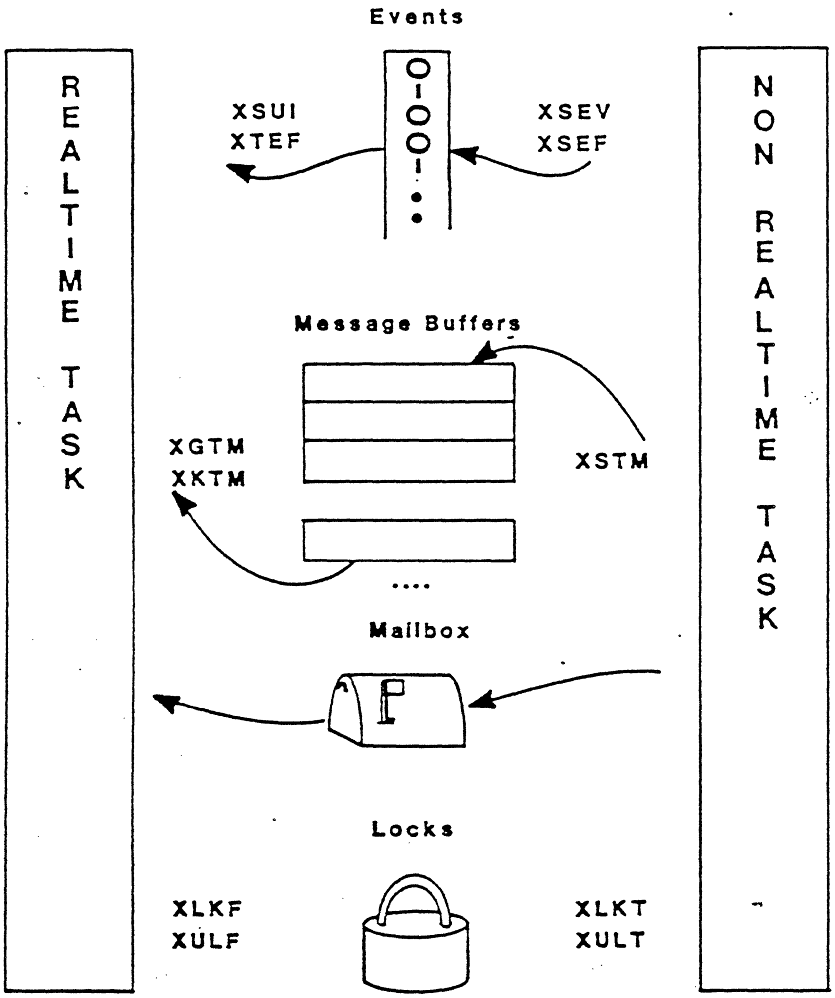
6. Time Delay Feature

* EVENTS 112, 113, 114, 115, 128

* XDEV

DO EXERCISE 7-7 -- TIMER TASK

INTERTASK
SYNCHRONIZATION AND COMMUNICATION



Events

REALTIME TASK

NON REALTIME TASK

XSUI
XTEF

XSEV
XSEF

Message Buffers

XGTM
XKTM

XSTM

Mailbox

Locks

XLKF
XULF

XLKT
XULT

SESSION 8 -- ADVANCED PDOS FILE MANAGER

GOALS:

1. You should understand file structure of PDOS Disks.

2. You should become acquainted with disk utilities.

NOTES:

1. The NERD Standard

- disk Ø..99, 1ØØ..255
- track Ø sides 1 and 2 info
- 96 TPI

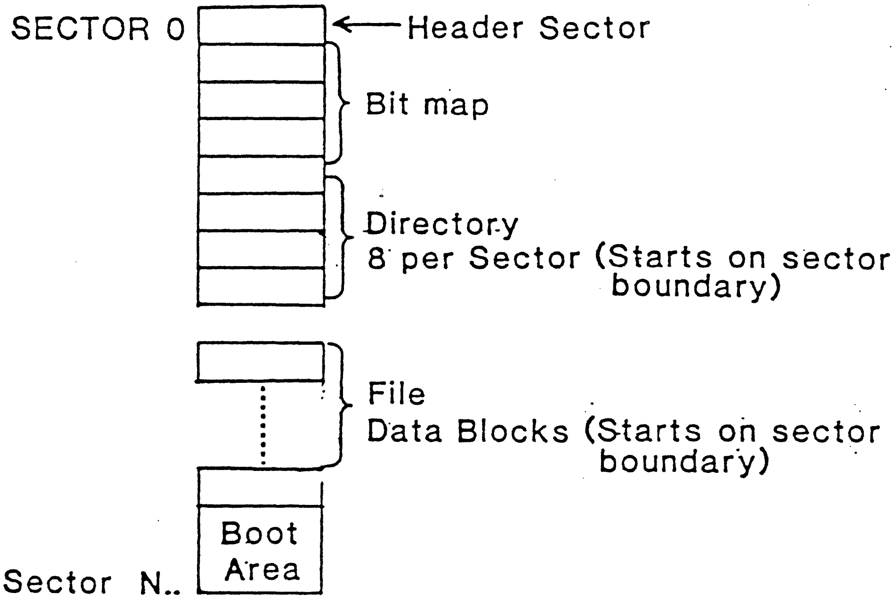
NERD NUMBERING

0... 255	available
0	primary floppy
1	secondary floppy
8	RAM disk
100	primary floppy with ability to R/W track 0

Typical hard disk logical units:

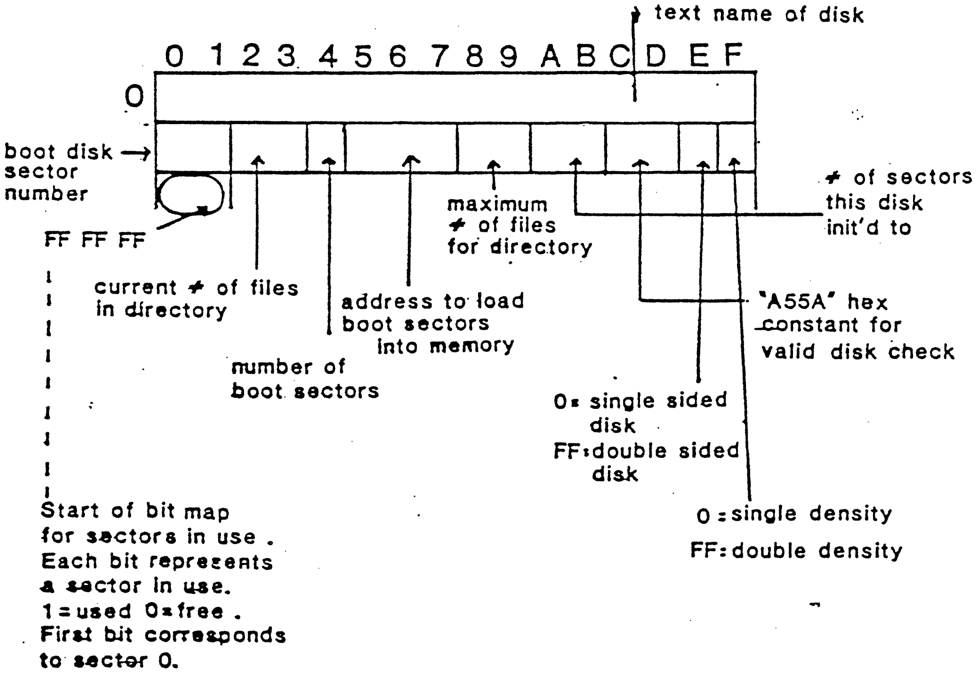
2	1/2 of winch #1
3	1/2 of winch #1
4	floppy image on winch #1
5	1/2 of winch #2
6	1/2 of winch #2
7	floppy image on winch #2

* Sector 0


SECTOR 0
Header Sector
Bit map
Directory
8 per Sector (Starts on sector boundary)
File
Data Blocks (Starts on sector boundary)
Boot Area
Sector N..

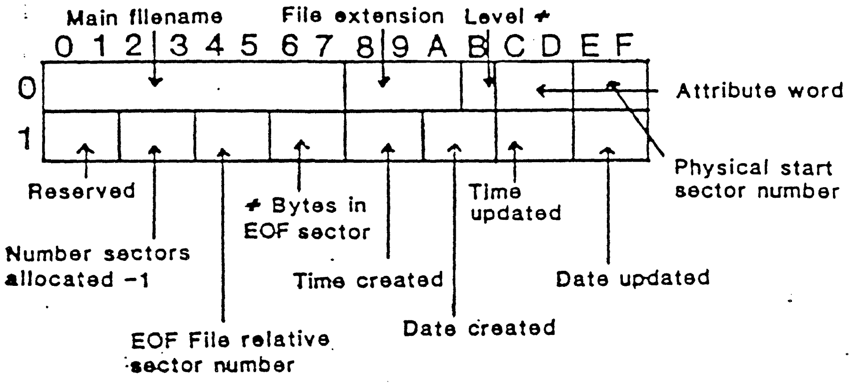
```
ector/Disk=$ØØØØ (Ø)/Ø
ØØ-ØØF  5Ø 44 4F 53 2Ø 32 2E 36 65 2Ø 23 31 ØØ ØØ ØØ ØØ PDOS 2.6e#1....
1Ø-Ø1F  Ø9 4Ø ØØ 5Ø 88 ØØ Ø8 ØØ ØØ AØ Ø9 4Ø A5 5A FF FF .@.P......@%Z..
2Ø-Ø2F  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
...................................
```

SECTOR 0 HEADER


text name of disk
0 1 2 3 4 5 6 7 8 9 A B C D E F
0
boot disk sector number
FF FF FF
current # of files in directory
number of boot sectors
address to load boot sectors into memory
maximum # of files for directory
# of sectors this disk init'd to
'A55A' hex constant for valid disk check
0= single sided disk
FF=double sided disk
0 =single density
FF= double density
Start of bit map for sectors in use. Each bit represents a sector in use. 1=used 0=free. First bit corresponds to sector 0.

*Directory Sectors

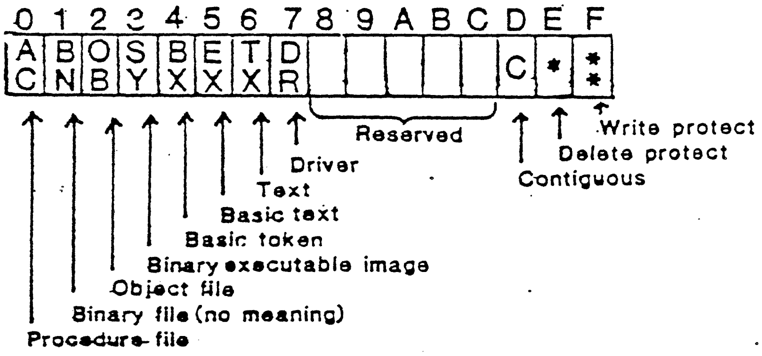
FILE DIRECTORY FORMAT



Time Hours * 256 + minutes

Date (2 digit year * 16+ month) * 32 + day

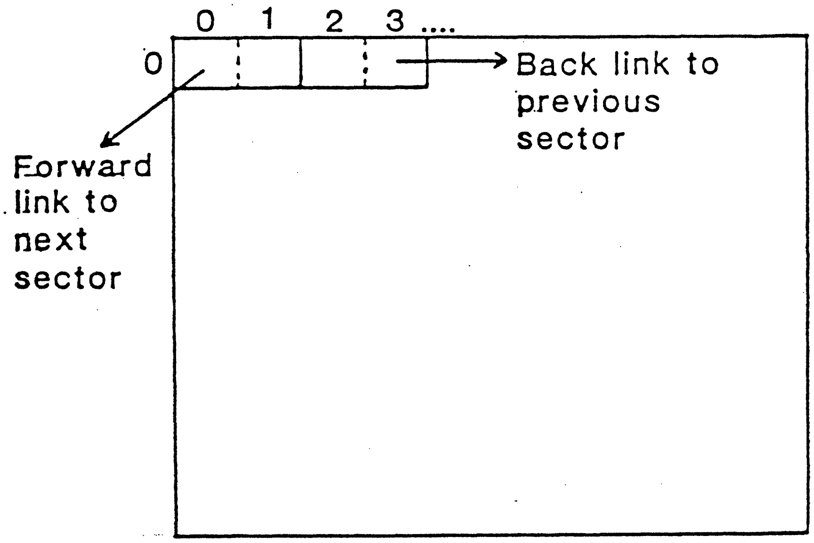
Attribute word=



```
Sector/Disk=$ØØØ2 (2)/Ø
ØØØ-ØØF  41 53 4D ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ 8Ø ØØ ØØ 16 ASM.............
Ø1Ø-Ø1F  ØØ ØØ ØØ ØØ ØØ ØØ ØØ 2E ØE 2Ø A8 97 ØE 2Ø A8 97 ......... (.. (.
Ø2Ø-Ø2F  42 58 52 45 46 ØØ ØØ ØØ ØØ ØØ ØØ Ø4 Ø4 Ø4 Ø1 5Ø BXREF..........P
Ø3Ø-Ø3F  ØØ ØØ ØØ 2A ØØ 2A ØØ 35 ØF 35 A8 F2 14 ØC A9 17 ...*.*.5.5(r..).
...................................

ØAØ-ØAF  43 4C 5Ø 54 58 ØØ ØØ ØØ ØØ ØØ ØØ ØØ Ø2 Ø4 ØØ 17 HLPTX...........
ØBØ-ØBF  ØØ ØØ ØØ 21 ØØ 21 ØØ 88 ØE 2B A8 97 ØE 2C A8 97 ...!.!...+(..,(.
```

*File Sectors

DATA SECTOR



```
ector/Disk=$ØØ16 (22)/Ø
Ø-ØØF   ØØ ØØ ØØ ØØ 4D 41 53 4D 2Ø 26 31 3A 53 52 2C 23 ....MASM &1:SR,#
Ø-Ø1F   4F 42 4A 2F 38 ØD 49 46 2Ø 26 3Ø 2E 52 43 ØD 4D OBJ/8.IF &Ø.RC.M
Ø-Ø2F   53 59 46 4C 2Ø 4F 42 4A 2F 38 2C 23 26 31 ØD 52 SYFL OBJ/8,#&1.R
Ø-Ø3F   43 ØD ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ C...............
Ø-Ø4F   ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ................

ctor/Disk=$ØØ17 (23)/Ø
Ø-ØØF   ØØ 18 ØØ ØØ 48 45 4C 5Ø ØD 2Ø 46 6F 72 2Ø 66 75 ....HELP. For fu
Ø-Ø1F   72 74 68 65 72 2Ø 68 65 6C 7Ø 2C 2Ø 65 6E 74 65 rther help, ente
Ø-Ø2F   72 2Ø 27 48 45 2Ø 27 2Ø 66 6F 6C 6C 6F 77 65 64 r 'HE ' followed

ctor/Disk=$ØØ18 (24)/Ø
Ø-ØØF   ØØ 19 ØØ 17 53 43 41 4C Ø9 Ø9 5Ø 41 53 43 41 4C ....SCAL..PASCAL

ctor/Disk=$ØØ19 (25)/Ø
Ø-ØØF   ØØ 1A ØØ 18 2D 2Ø 52 65 6E 61 6D 65 2Ø 66 69 6C ....- Rename fil

ctor/Disk=$ØØ37 (55)/Ø
Ø-ØØF   ØØ 38 ØØ 36 69 6C 65 73 3A ØD ØD 2Ø 2Ø 2Ø 5Ø 4D .8.6iles:..   PM

ctor/Disk=$ØØ38 (56)/Ø
Ø-ØØF   ØØ ØØ ØØ 37 4F 3A 5Ø 41 53 2Ø 61 6E 64 2Ø 5Ø 48 ...7O:PAS and PH
```

3. File Utilities

* MDDUMP
  - Used to view and edit disks.

* MDDMAP
  - Used to verify disk links for files
  - Sample execution:

```
68K PDOS Disk Diagnostic Mapper Utility Ø9/14/84
  Disk # = Ø
  Output File Name =
  Disk Diagnostic Map

  Disk Name = PDOS 2.6e #1               SECTOR Ø DISK INFORMATION
  Files = 8Ø/16Ø
  Boot sector = 2368
  Boot size/addr = 136/$ØØØ8ØØ
  PDOS Sectors = 2368
  Disk Density = D

 Ø ASM     AC    1/1    14:32 Ø4/23/84  14:32 Ø4/23/84
     22-22 <== Sector numbers

 4 BXREF   EX C 43/43   15:53 Ø7/18/84  2Ø:12 Ø8/23/84
     336-378
```

DO EXERCISE 8-1 -- MDDMAP / MDDUMP

* MCHATLE
  - Change Attributes and levels of selected files.

* MLEVEL
  - Display files by level.

DO EXERCISE 8-2 -- MLEVEL

* MDLOOK
  - Look for possible files on disk.

* MDSAVE
  - Recover all possible files on disk.

DO EXERCISE 9-3 -- MDLOOK / MDSAVE

4. Disk Initialization Procedure.

- xFRMT
  Hard Format -- Function of Hardware device.
  xFRMT program. This is more fully described in Chapter 2 of the PDOS Reference Manual.

- MINIT
  Soft format -- Gets disk ready for PDOS.
  MINIT program. This specifies the number of files and sectors for the disk.

5. Disk Backup Procedure.

* MBACK	-	Sector for sector image
* MTRANS	-	File by file transfer
* MDNAME	-	Gives disk a text name for listings

6. RAM Disk

* Dynamic any memory location
* System Default is unit 8 (monitor changeable)
* FM -N

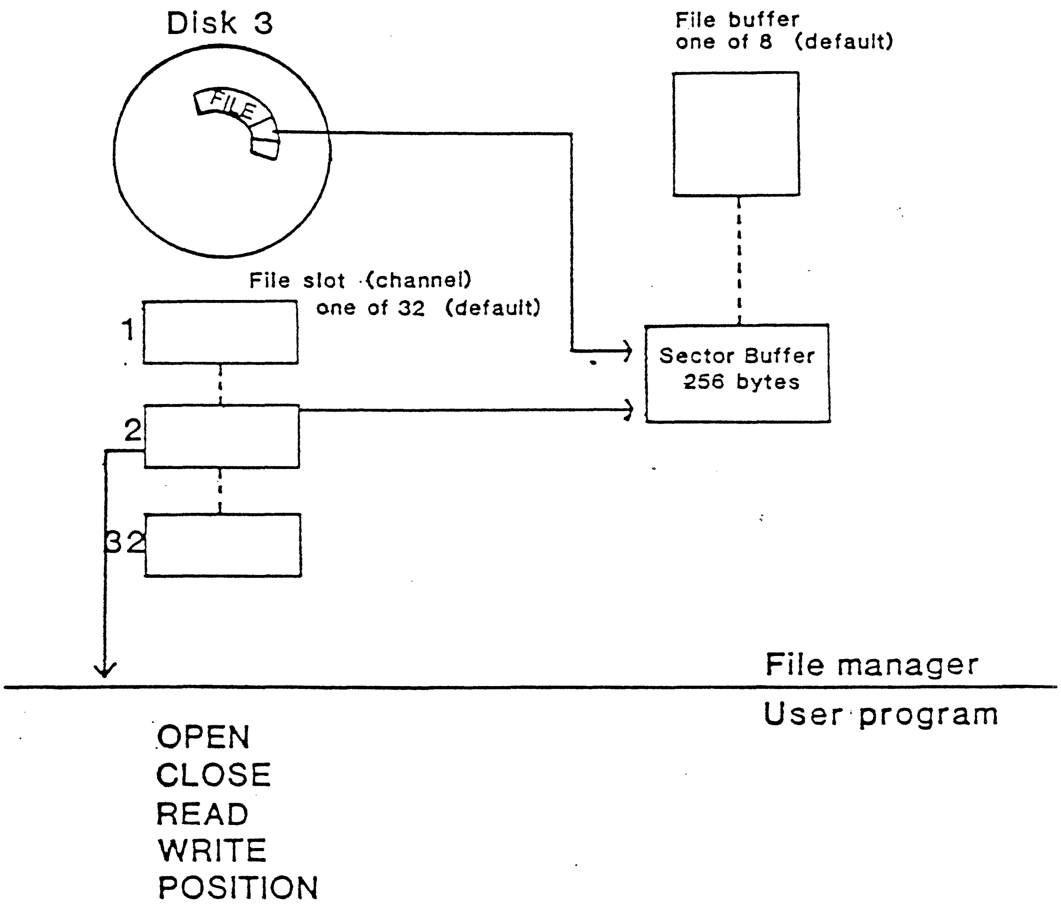
```
  addr=$hhhhhhhh
  RD -U,N,$hhhhhhhh    where U is the desired unit number
    for the disk and N is 4*the amount freed
```

DO EXERCISE 8-4 -- RAM DISK AND MTRANS

7. File Access Modes

* Read Only
* Shared Random Access
* Random Access
* Sequential Access
* Contiguous File
* Delete Protect
* Write Protect
* File Lock
* The RS Command (ARG for disk)
* The FS Command (3-19)
* What is a File Slot?
* 8 Active Buffers
* Most Recently Accessed Queue

FILE MANAGER



Disk 3

FILE

File buffer
one of 8 (default)

File slot (channel)
one of 32 (default)

1

2

32

Sector Buffer
256 bytes

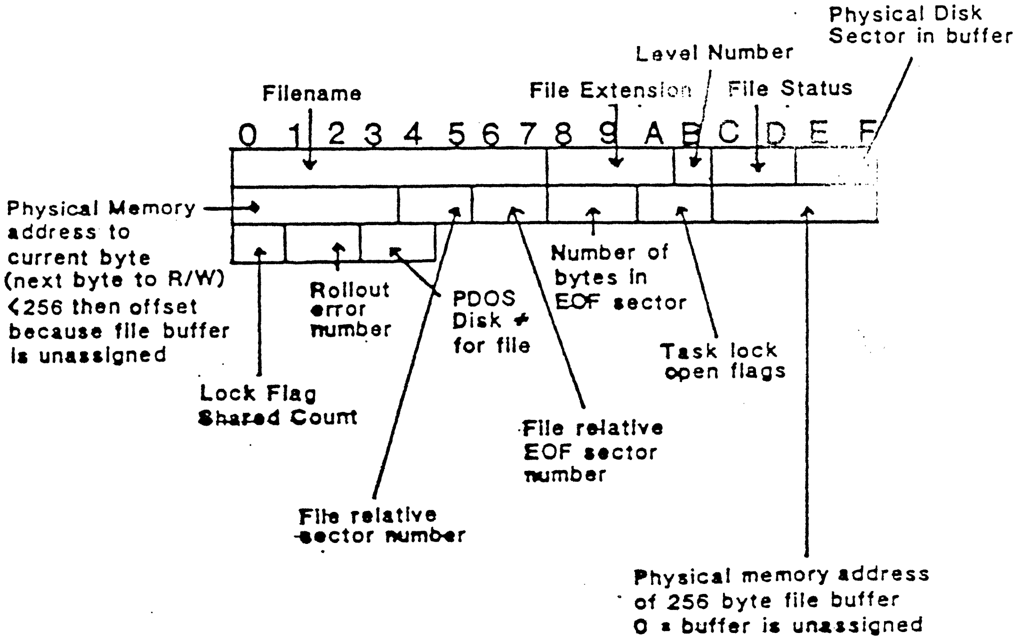
File manager

User program

OPEN
CLOSE
READ
WRITE
POSITION

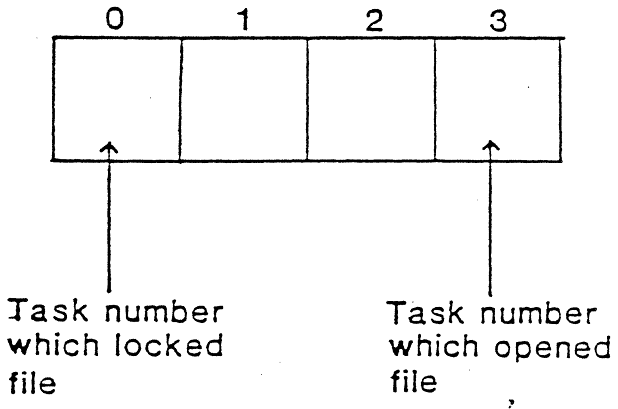
```
1>FS
Slot  Name    ST    SM       PT       SI     EOF       TN       BF       FLGS
32    A:1/1  C100  0057  0000A2DF  0000  0007/0E  0000  0000A25A  00000000

- XLFN          to get slot address
```

FILE SLOT

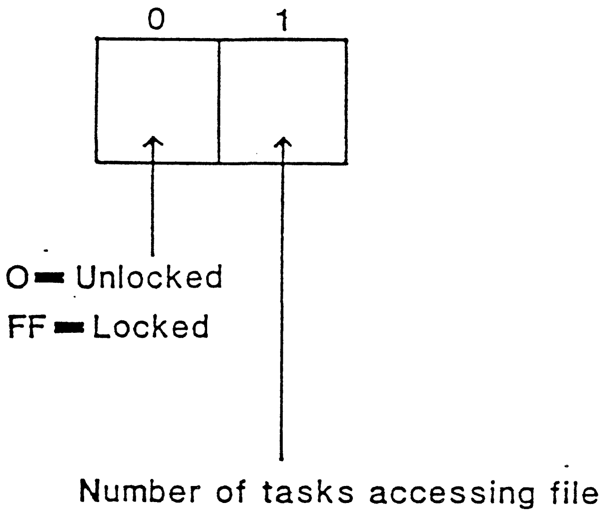


Physical Disk Sector in buffer
Level Number
Filename
File Extension
File Status
0 1 2 3 4 5 6 7 8 9 A B C D E F
Physical Memory address to current byte (next byte to R/W) <256 then offset because file buffer is unassigned
Rollout error number
PDOS Disk # for file
Number of bytes in EOF sector
Task lock open flags
Lock Flag Shared Count
File relative EOF sector number
File relative sector number
Physical memory address of 256 byte file buffer 0 = buffer is unassigned

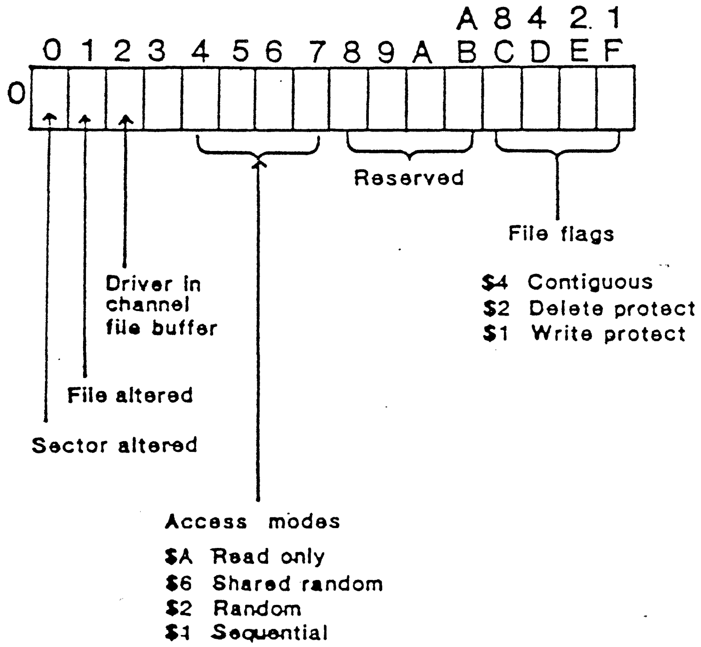
TASK LOCK / OPEN FLAGS



LOCK FLAG/SHARED COUNT



FILE STATUS



- * Direct Disk Access
    - XRSE XWSE
    - BUFFER ADDRESS
    - DISK NUMBER
    - SECTOR NUMBER

DO EXERCISE 8-5 -- DIRECT DISK ACCESS

SESSION 9 -- LANGUAGES

GOALS:

1. You should become acquainted with the various languages available on PDOS and how those languages can use the PDOS system.
2. This is designed only as an overview and not as a detailed discussion of the languages.

NOTES:

1. BASIC

BASIC FEATURES

- * Meaningful, Unlimited length variable names
- * Multiple line, Recursive functions
- * Local function variables
- * Extensive line editing commands
- * Fast 64 bit floating point
- * Context oriented strings
- * Full disk file interface
- * Stand alone run support
- * No 64k byte boundary restrictions
- * Intertask communications arrays
- * Assembly language support
- * Formatted print commands

BASIC ENVIRONMENT

- * Semi-compilied
- * Line editor

BASIC FUNCTIONS AND PROCEDURES

ABS	-- ABSOLUTE VALUE
ADR	-- ADDRESS OF EXPRESSION
ADV	-- VARIABLE DEFINITION ADDRESS (ARRAY VECTOR)
ALOAD	-- LOAD OBJECT FILE
AND	-- TRUE IF OPERANDS NON-ZERO
ATN	-- ARC TANGENT
BAUD	-- SET BAUD RATE OF PORT
BIT	-- TEST/SET BITS IN VARIABLE
BYE	-- RETURN TO MONITOR
CALL	-- ASSEMBLY LANGUAGE INTERFACE
CLEAR	-- ZERO BASIC VARIABLE SPACE
CLOSE	-- CLOSE FILE
COM	-- SHARED MEMORY AREA ACROSS 'RUN' AND 'CHAIN'
COS	-- COSINE

DATA	CONSTANT DEFINITION
DATE	READ/SET SYSTEM DATE
DEFINE	DEFINE FILE
DEFN	DEFINE BASIC FUNCTION
DELAY	SET TIMER TO EXPIRE IN 'N' TICS
DELETE	DELETE FILE
DIM	DIMENSION ARRAY VARIABLE
DISPLAY	COPY FILE TO SCREEN
ELSE	EXECUTE THIS LINE IF PREVIOUS 'IF' WAS FALSE
EQUATE	ASSIGN SIMPLE VARIABLE NAME TO ARRAY ELEMENTS
ERROR	DEFINE ERROR TRAP
ESCAPE	ENABLE ESCAPE TO STOP EXECUTION
EVENT	SET/CLEAR EVENT FLAG
EVF	TEST EVENT FLAG
EXP	EXPONENTIATION
EXTERNAL	DEFINE 'CALL' ENTRY POINTS
FILE	FILE I/O FUNCTIONS (SELECT, READ, WRITE, POSITION, LOCK, UNLOCK)
FILES	DISPLAY FILE DIRECTORY
FNEND	DELIMIT MULTI-LINE USER FUNCTION
FOR	ITERATIVE LOOP STATEMENT
FNPOP	CLEAN-UP AFTER ABORTING FROM USER FUNCTION
FRA	FRACTIONAL PART OF EXPRESSION
FREE	RECOVER SYSTEM MEMORY
GETM	GET TASK MESSAGE
GLOBAL	CREATE SHARED MEMORY AREA BETWEEN TASKS
GOPEN	READ-ONLY OPEN
GOSUB	BASIC SUBROUTINE CALL
GOTO	CONTROL TRANSFER TO LINE NUMBER
IF	CONDITIONAL EXECUTION
IMASK	SET SYSTEM INTERRUPT MASK
INP	INTEGER PART OF EXPRESSION
INPUT	READ DATA FROM CONSOLE
INT	GREATEST INTEGER (FLOOR) OF EXPRESSION
KEY	SAMPLE PORT FOR CHARACTER
LABEL	DEFINE LINE NUMBER LABEL
LAND	LOGICAL 'AND' TWO OPERANDS
LEN	LENGTH OF STRING
LET	VARIABLE ASSIGNMENT
LIST	DISPLAY PROGRAM IN MEMORY
LISTRP	DISPLAY PROGRAM TOKENS IN REVERSE POLISH ORDER
LNOT	LOGICAL COMPLEMENT OF EXPRESSION
LOAD	BRING BASIC PROGRAM INTO MEMORY
LOCAL	DECLARE VARIABLES LOCAL TO USER FUNCTION
LOG	NATURAL LOGARITHM
LOR	LOGICAL 'OR' OF TWO OPERANDS

LXOR	-- LOGICAL 'EXCLUSIVE OR' OF TWO OPERANDS
MAIL	-- GLOBAL ARRAY FOR INTER-TASK COMMUNICATIONS
MCH	-- STRING MATCH FUNCTION
MEM	-- READ/WRITE BYTE IN MEMORY (8 BITS)
MEMW	-- READ/WRITE WORD IN MEMORY (16 BITS)
MEML	-- READ/WRITE LONG WORD IN MEMORY (32 BITS)
MEMP	-- READ/WRITE BASIC VARIABLE ACROSS MEMORY PAGES
NCH	-- NUMERIC VALUE OF CHARACTER
NEW	-- CLEAR PROGRAM AND VARIABLES FROM MEMORY
NEXT	-- END OF 'FOR' LOOP
NOESC	-- DISABLE ESCAPE FROM STOPPING PROGRAM
NOT	-- TRUE WHEN ZERO
ON	-- CASE STATEMENT FOR GOTO, GOSUB, LET
OPEN	-- OPEN FILE FOR SEQUENTIAL ACCESS
OR	-- TRUE IF EITHER OPERAND NON-ZERO
PDOS	-- GET COMMAND LINE
POP	-- CLEAN-UP AFTER ABORTING FROM SUBROUTINE
PRINT	-- PUT DATA TO CONSOLE
PURGE	-- SELECTIVE DELETE OF PROGRAM SEGMENTS
RCP	-- READ CURSOR POSITION
READ	-- READ VALUES FROM 'DATA' STATEMENTS
REM	-- REMARK (COMMENT)
RENAME	-- CHANGE NAME OF FILE
RESET	-- CLOSE ALL FILES
RESTORE	-- DETERMINE WHAT 'DATA' STATEMENT IS READ BY 'READ'
RETURN	-- EXIT FROM SUBROUTINE
RND	-- RANDOM VALUE BETWEEN Ø..1
ROPEN	-- OPEN FILE FOR RANDOM ACCESS
RUN	-- EXECUTE PROGRAM
SAVE	-- WRITE PROGRAM TO DISK (ASCII FORM)
SAVEB	-- WRITE PROGRAM TO DISK (TOKENIZED FORM)
SENDM	-- SEND TASK MESSAGE
SGN	-- RETURN SIGN OF EXPRESSION
SIN	-- SINE OF EXPRESSION
SIZE	-- SHOW SIZE OF PROGRAM, TABLES, AND FREE SPACE
SKIP	-- JUMP OVER 'N' LINES IN PROGRAM
SOPEN	-- OPEN FILE FOR SHARED ACCESS
SPOOL	-- REDIRECT OUTPUT TO FILE
SQR	-- SQUARE ROOT
SRH	-- SEARCH FOR ONE STRING IN ANOTHER
STACK	-- DISPLAY 'GOSUB' STACK ENTRIES
STOP	-- HALT PROGRAM
SWAP	-- DELAY THIS TASK WHILE ANOTHER EXECUTES
SYS	-- ACCESS TO SYSTEM VARIABLES

TAN -- TANGENT OF EXPRESSION
THEN -- EXECUTE THIS LINE IF PREVIOUS 'IF' WAS TRUE
TIC -- CURRENT VALUE OF THE SYSTEM TIMER
TIME -- SET/READ SYSTEM TIME
TRACE -- ENABLE/DISABLE DEBUGGING TRACE
TSK -- RETURN STATUS OF TASK 'N'

UNIT -- REDIRECT OUTPUT TO DIFFERENT PORT

WAIT -- SUSPEND ON EVENT

2. C

C FEATURES

* Full C Language
* Provides direct access to over 100 PDOS system calls
* Supports a subset of UNIX library for portability
* Optimal code generation

C ENVIRONMENT

* Full native code
* Uses PDOS MASM and QLINK

C STANDARD LIBRARY ROUTINES

ALLOC, MALLOC, CALLOC, REALLOC -- DYNAMIC MEMORY ALLOCATION
ATOI, ATOL, ITOA -- ASCII CONVERSION ROUTINES
CLOSE -- CLOSE A FILE SLOT
CREAT -- CREATE A FILE AND OPEN IT
CTYPE -- CHARACTER TYPE MACROS AND FUNCTIONS
EXIT, _EXIT -- CLOSE STREAMS AND EXIT TO MONITOR W/STATUS
FCLOSE -- CLOSE A STREAM
FGETS -- GET A STRING FROM A STREAM
FOPEN -- OPEN A STREAM
FPRINTF, PRINTF, SPRINTF -- FORMATTED OUTPUT ROUTINES

FPUTS -- PUT A STRING TO A STREAM
FREE, MFREE -- DYNAMIC MEMORY DEALLOCATION
FSCANF, SCANF, SSCANF -- FORMATTED INPUT ROUTINES
FSEEK -- POSITION A STREAM
FTELL -- RETURN CURRENT POSITION IN STREAM
GETC -- GET A CHARACTER FROM A STREAM
GETCHAR -- GET A CHARACTER FROM STANDARD INPUT (MACRO)

GETS -- GET A STRING FROM STANDARD INPUT
INDEX -- RETURN POSITION OF CHARACTER IN STRING
LSEEK -- POSITION FILE SLOT
OPEN -- OPEN A FILE SLOT
PUTC -- PUT A CHARACTER TO A STREAM
PUTCHAR -- PUT A CHARACTER TO STANDARD OUTPUT (MACRO)

PUTS -- PUT A LINE TO STANDARD OUTPUT

READ -- READ DATA FROM FILE SLOT
RINDEX -- RETURN POSITION OF CHARACTER IN STRING (f/RIGHT)
RTIME -- RETURN SYSTEM TIME, DATE
SBRK -- ADD 'N' BYTES TO TASK BREAK (MEMORY ALLOCATION)

STRINGS: STRCAT, STRNCAT, STRCHR, STRRCHR, STRCMP,
STRNCMP, STRCPY, STRNCPY, STREND, STRLEN,
STRPBRK, STRSPN, STRTOK -- STRING HANDLING FUNCTIONS

SYSTEM -- SPAWN A TASK AND PASS IT A COMMAND STRING
TTYOPEN -- OPEN A TERMINAL PORT AS A STREAM
UNGETC -- PUSH A BYTE BACK ONTO AN INPUT STREAM
UNLINK -- DELETE A FILE
WRITE -- WRITE DATA TO A PORT/FILE SLOT

C INTERFACE TO PDOS PRIMITIVES

XAPF -- APPEND FILE
XBCP -- BAUD CONSOLE PORT
XBFL -- BUILD FILE LISTING
XBUG -- ENTER DEBUGGER
XCBC -- CHECK FOR BREAK CHAR
XCBD -- CONVERT BINARY TO DECIMAL STRING
XCBH -- CONVERT BINARY TO HEXADECIMAL STRING
XCBP -- CHECK FOR BREAK CHAR/PAUSE
XCBX -- CONVERT BINARY TO DECIMAL IN BUFFER
XCDB -- CONVERT DECIMAL STRING TO BINARY
XCHF -- CHAIN FILE
XCHX -- CONVERT BINARY TO HEXADECIMAL STRING IN BUFFER
XCFA -- CLOSE FILE WITH ATTRIBUTES
XCLF -- CLOSE FILE
XCLS -- CLEAR SCREEN
XCPY -- COPY FILE
XCTB -- CREATE TASK BLOCK
XDEV -- DELAY/SET EVENT
XDFL -- DEFINE FILE
XDLF -- DELETE FILE
XDTV -- DEFINE USER'S VECTOR TABLE
XERR -- ERROR EXIT
XEXT -- EXIT TO MONITOR
XFBF -- FLUSH BUFFER TO DISK
XFFN -- FIX FILE NAME
XFTD -- FIX TIME/DATE
XFUM -- FREE USER MEMORY
XGCC -- GET CHARACTER CONDITIONALLY
XGCP -- GET CHARACTER FROM PORT
XGCR -- GET CHARACTER
XGLB -- GET LINE IN BUFFER
XGLU -- GET LINE IN USER BUFFER
XGML -- GET MEMORY LIMITS
XGNP -- GET NEXT PARAMETER
XGTM -- GET TASK MESSAGE

XGUM -- GET USER MEMORY
XKTB -- KILL TASK
XKTM -- KILL TASK MESSAGE
XLDF -- LOAD FILE
XLER -- LOAD ERROR REGISTER
XLKF AND XULF -- LOCK/UNLOCK FILE
XLKT AND XULT -- LOCK/UNLOCK TASK
XNOP -- NON-EXCLUSIVE RANDOM OPEN FILE
XPBC -- PUT USER BUFFER TO CONSOLE
XPCC -- PUT CONSOLE CHARACTER
XPCL -- PUT CARRIAGE RETURN/LINE FEED TO CONSOLE
XPDC -- PUT DATA TO CONSOLE
XPLC -- PUT LINE TO CONSOLE
XPSC -- POSITION CURSOR
XPSF -- POSITION FILE
XPSP -- PUT SPACE TO CONSOLE
XRBF -- READ BYTES FROM FILE
XRCN -- RESET CONSOLE INPUTS
XRCP -- READ PORT CURSOR POSITION
XRDE -- READ NEXT DIRECTORY ENTRY
XRDM -- DUMP REGISTERS TO CONSOLE
XRDT -- GET DATE/TIME
XRFA -- READ FILE ATTRIBUTES
XRLF -- READ LINE FROM FILE
XRNF -- RENAME FILE
XROO -- OPEN FILE READ-ONLY RANDOM ACCESS
XROP -- OPEN FILE RANDOM ACCESS
XRPS -- READ PORT STATUS
XRSE -- READ SECTOR FROM DISK
XRST -- RESET FILE
XRSZ -- READ SECTOR ZERO
XRTM -- READ TIME
XRTS -- READ TASK STATUS
XRWF -- REWIND FILE
XSEF -- SET/CLEAR EVENT FLAG WITH SWAP
XSEV -- SET EVENT FLAG
XSOP -- OPEN FILE SEQUENTIALLY
XSPF -- SET PORT FLAG
XSTM -- SEND TASK MESSAGE
XSTP -- SET TASK PRIORITY
XSUI -- SUSPEND UNTIL INTERRUPT OR EVENT
XSWP -- SWAP TO NEXT TASK
XSZF -- GET DISK PARAMETERS
XTAB -- TAB TO COLUMN ON SCREEN
XTEF -- TEST EVENT FLAG
XUDT -- UNPACK DATE INTO STRING
XUTM -- UNPACK TIME INTO STRING
XWBF -- WRITE BYTES TO FILE
XWDT -- WRITE DATE/TIME TO SYSTEM CLOCK
XWFA -- WRITE FILE ATTRIBUTES
XWLF -- WRITE LINE TO FILE
XWSE -- WRITE SECTOR

```
     XWTM -- WRITE TIME TO PDOS
     XZFL -- ZERO FILE

ARITHMETIC FUNCTIONS -- UNARY OPERATORS
     CEIL, FLOOR, FABS, FPNEG :
       CEILING, FLOOR, ABSOLUTE VALUE, NEGATION

ARITHMETIC FUNCTIONS -- BINARY OPERATORS
     FPADD, FPCMP, FPDIV, FPMOD, FPMUL, FPSUB :
       FLOATING POINT ADD, COMPARE, DIVIDE,
       MODULO, MULTIPLY, SUBTRACT

CONVERSION FUNCTIONS
     ATOF, ETOA, FTOA :
       ASCII TO FLOATING POINT,
       FLOATING POINT TO ASCII ('E' AND 'F' FORMAT)
     FPLTOF, FPFTOL :
       FLOATING POINT LONG TO FLOAT, FLOAT TO LONG

LOGARITHMIC/EXPONENTIAL FUNCTIONS
     EXP, POW, LOG, LOG1Ø, SQRT :
       EXPONENTIAL, POWER, NATURAL LOG, COMMON LOG, SQUARE ROOT

     FREXP, LDEXP, MODF  -- MISCELLANEOUS EXPONENTIAL

TRIGONOMETRIC FUNCTIONS
     ATAN, ATAN2 : ARCTANGENT
     COS, SIN, TAN : COSINE, SINE, TANGENT
     COSH, SINH, TANH :  HYPERBOLIC COSINE, SINE, TANGENT
```

3. FORTRAN 77

FORTRAN 77 FEATURES

* ANSI FORTRAN 77 by Absoft Corp.
* FULL FORTRAN-77 LANGUAGE FEATURES AVAILABLE
* EXTENSIONS TO FORTRAN-77 FROM ABSOFT:
  - INCLUDE statement
  - dynamic linking
  - parameters in the program statement
  - various allowable comment formats
  - multiple statements per line
  - the SELECT/CASE/CASE DEFAULT/END SELECT construct
  - DO/CYCLE/EXIT/REPEAT construct
  - VIRTUAL arrays
  - TYPE, ACCEPT statements

FORTRAN PROGRAMMING ENVIRONMENT

* SYMBOLIC DEBUGGER
* EXECUTION PROFILER
* LINKER, LIBRARIAN

INTRINSIC FUNCTIONS

INT [INT,IFIX,IDINT] -- CONVERSION TO INTEGER
REAL [REAL,FLOAT,SNGL] -- CONVERSION TO REAL
DBLE -- CONVERSION TO DOUBLE
CMPLX -- CONVERSION TO COMPLEX
ICHAR -- CONVERSION OF CHARACTER TO INTEGER (PASCAL 'ORD')
CHAR -- CONVERSION OF INTEGER TO CHARACTER (PASCAL 'CHR')
AINT [AINT,DINT] -- TRUNCATION
ANINT [ANINT,DNINT] -- NEAREST WHOLE NUMBER
NINT [NINT,IDNINT] -- NEAREST INTEGER
ABS [IABS,ABS,DABS,CABS] -- ABSOLUTE VALUE
MOD [MOD,AMOD,DMOD] -- REMAINDERING
SIGN [ISIGN,SIGN,DSIGN] -- TRANSFER OF SIGN
DIM [IDIM,DIM,DDIM] -- POSITIVE DIFFERENCE
DPROD -- DOUBLE PRECISION PRODUCT
MAX [MAXØ,AMAX1,DMAX1,AMAXØ,MAX1] -- LARGEST VALUE
MIN [MINØ,AMIN1,DMIN1,AMINØ,MIN1] -- SMALLEST VALUE
LEN -- LENGTH OF CHARACTER ENTITY
INDEX -- INDEX OF A SUBSTRING
TRIM -- TRIM TRAILING BLANKS
REPEAT -- STRING REPLICATION
AIMAG -- IMAGINARY PART OF COMPLEX ARGUMENT
CONJG -- CONJUGATE OF A COMPLEX ARGUMENT
SQRT [SQRT,DSQRT,CSQRT] -- SQUARE ROOT
EXP [EXP,DEXP,CEXP] -- EXPONENTIAL
LOG [ALOG,DLOG,CLOG] -- NATURAL LOGARITHM
LOG1Ø [ALOG1Ø,DLOG1Ø] -- COMMON LOGARITHM
SIN [SIN,DSIN,CSIN] -- SINE
COS [COS,DCOS,CCOS] -- COSINE

TAN [TAN,DTAN] -- TANGENT
ASIN [ASIN,DASIN] -- ARCSINE
ACOS [ACOS,DACOS] -- ARCCOSINE
ATAN [ATAN,DATAN] -- ARCTANGENT
ATAN2 [ATAN2,DATAN2] -- ARCTANGENT (X/Y)
SINH [SINH,DSINH] -- HYPERBOLIC SINE
COSH [COSH,DCOSH] -- HYPERBOLIC COSINE
TANH [TANH,DTANH] -- HYPERBOLIC TANGENT
STRING COMPARISONS [LGE,LGT,LLE,LLT]
BYTE [BYTE,WORD,LONG] -- INSPECT MEMORY AT AN ADDRESS

SHIFT -- LOGICAL SHIFT LEFT OR RIGHT

4. PASCAL

PASCAL FEATURES

* Compatible on significant ISO standard
* UCSD string extensions
* Type override
* Origin variables
* 32 bit and 64 bit floating point
* EPROMable and shareable code
* PDOS interface library

PASCAL ENVIRONMENT

* Compiled -- NOT P-Code
* Uses PDOS MASM and QLINK

PASCAL FUNCTIONS AND PROCEDURES

ARCTAN	-- ARC TANGEN
COS	-- COSINE
EXP	-- EXPONENTIATION
LN	-- NATURAL LOGARITHM
SIN	-- SINE
SQR	-- SQUARE
SQRT	-- SQUARE ROOT
TRUNC	-- TRUNCATE TO INTEGER
ROUND	-- ROUND TO INTEGER
ABS	-- ABSOLUTE VALUE
ODD	-- TRUE IF ODD
ORD	-- ORDINAL POSITION OF ELEMENT IN ENUMERATION
CHR	-- INTEGER TO CHARACTER TYPE CONVERSION
SUCC	-- SUCCESSOR TO ELEMENT IN ENUMERATION
PRED	-- PREDECESSOR TO ELEMENT IN ENUMERATION
FLOAT	-- INTEGER TO FLOATING POINT CONVERSION
NEW	-- DYNAMIC VARIABLE ALLOCATION
DISPOSE	-- DYNAMIC VARIABLE DEALLOCATION

PASCAL I/O PROCEDURES AND FUNCTIONS

READ -- READ DATA FROM FILE

READLN -- READ LINE FROM FILE

WRITE -- WRITE DATA TO FILE

WRITELN -- WRITE LINE TO FILE

RESET -- OPEN FILE FOR READING

REWRITE -- OPEN FILE FOR WRITING

CLOSE -- CLOSE FILE

PAGE -- OUTPUT FORM FEED OR CLEAR SCREEN

PUT/GET -- READ/WRITE BINARY TO FILE

SEEK -- POSITION WITHIN FILE

EOF -- TRUE IF AT END OF FILE

EOLN -- TRUE IF AT END OF LINE

(PASCAL OFFERS A SIMILAR INTERFACE TO PDOS PRIMITIVES AS C)

SESSION 1Ø -- PDOS INTERNALS

GOAL:

1. You should become acquainted with some of the basic system variables.

NOTES:

1. SYRAM -- MSYRAM:SR

```
      * How address is obtained
           A5 POINTER
        XGML
        SYS[39]

      * SYRAM
           How to do equates --- PDOS option, INCLUDE file.
           CAUTION: Some locations changing with PDOS 3.Ø

        OFFSET  Ø
BIOS.   DS.L    1               ;ADDRESS OF BIOS ROM
MAIL.   DS.L    1               ;*MAIL ARRAY ADDRESS
RDKN.   DS.W    1               ;*RAM DISK #
RDKS.   DS.W    1               ;*RAM DISK SIZE
RDKA.   DS.L    1               ;*RAM DISK DISK ADDRESS
BFLG.   DS.B    1               ;BASIC PRESENT FLAG
DFLG.   DS.B    1               ;DIRECTORY FLAG
FCNT.   DS.W    1               ;FINE COUNTER
TICS.   DS.L    1               ;32 BIT COUNTER
MON.    DS.B    1               ;MONTH
DAY.    DS.B    1               ;DAY
YRS.    DS.B    6               ;YEAR
HRS.    DS.B    1               ;HOURS
MIN.    DS.B    1               ;MINUTES
SEC.    DS.B    6               ;SECONDS
PATB.    DS.B     16                ;INPUT PORT ALLOCATION TABLE

BRKF.   DS.B    16              ;INPUT BREAK FLAGS
F8BT.   DS.B    16              ;PORT FLAG BITS
UTYP.   DS.B    16              ;PORT UART TYPE
URAT.   DS.B    16              ;PORT RATE TABLE
EVTB.   DS.B    1Ø              ;Ø-79 EVENT TABLE
EVTO.   DS.B    2               ;8Ø-95 OUTPUT EVENTS
EVTI.   DS.B    2               ;96-111 INPUT EVENTS
EVTS.   DS.B    2               ;112-127 SYSTEM EVENTS
EVTZ    EQU     *-EVTB.
        DS.B    128/8           ;TASK 128 EVENTS
EVTM.   DS.L    4               ;EVENTS 112-115 TIMERS
        PAGE
```

```
**************************************************
*       SYRAM (continued)
*
BCLK.   DS.L    1     ;CLOCK ADJUST CONSTANT
TLTP.   DS.L    1     ;TASK LIST POINTER
UTCB.   DS.L    1     ;USER TASK CONTROL BLOCK POINTER
SUIM.   DS.W    1     ;SUPERVISOR INTERRUPT MASK
USIM.   DS.W    1     ;USER INTERRUPT MASK
SPTN.   DS.B    1     ;SPAWN TASK NUMBER (** MUST BE EVEN **)
UTIM.   DS.B    1     ;USER TASK TIME
TPRY.   DS.B    1     ;TASK PRIORITY (** MUST BE EVEN **)
TSKN.   DS.B    1     ;CURRENT TASK NUMBER
L2LK.   DS.B    1     ;LEVEL 2 LOCK (FILE PRIMITIVES, EVENT 12Ø)
L3LK.   DS.B    1     ;LEVEL 3 LOCK (DISK PRIMITIVES, EVENT 121)
TLCK.   DS.B    1     ;TASK LOCK FLAG
        DS.B    1
E122.   DS.B    1     ;BATCH TASK #
E123.   DS.B    1     ;SPOOLER TASK #
E124.   DS.B    1
E125.   DS.B    1
CKSM.   DS.L    1     ;SYSTEM CHECKSUM
PNOD.   DS.W    1     ;PNET NODE #
        DS.B    54-4*NBC           ;SPARES
BCLT.   DS.L    NBC                ;BASIC CALL TABLE
*
RWCL.   DS.W    16                 ;PORT ROW/COLUMN
OPIP.   EQU     *-4
        DS.L    16-1               ;OUTPUT PORT POINTERS
UART.   EQU     *-4
        DS.L    16-1               ;UART BASE ADDRESSES
*
**************************************************
*       THE FOLLOWING CHANGE WITH DIFFERENT CONFIGURATIONS
*
MAPS.   DS.B    MAPZ               ;SYSTEM MEMORY BIT MAP
MAPE.   EQU     *
PORT.   DS.B    (NPS-1)*NCP        ;CHARACTER INPUT BUFFERS
TQUE.   DS.W    NTB+1              ;TASK QUEUE
TLST.   DS.B    NTB*TBZ            ;TASK LIST
TMTF.   DS.L    NTM                ;TO/FROM/INDEX.W
TMBF.   DS.B    TMZ*NTM            ;TASK MESSAGE BUFFERS
DEVT.   DS.B    2+NEV*8            ;DELAYED EVENTS
BSCT.   DS.W    32                 ;BASIC SCREEN COMMAND TABLE
XCHI.   DS.W    NCB                ;CHANNEL BUFFERS QUEUE
XCHB.   DS.B    NCB*BPS            ;CHANNEL BUFFERS
XFSL.   DS.B    NFS*FSS            ;FILE SLOTS
        DS.B    72*3
SYSK.   EQU     *!$Ø7FF+1          ;SYSTEM STACK
SYRAMZ  EQU     SYSK.              ;END OF SYSTEM RAM
        PAGE
```

2. Processor ID Letters

```
MSYRAM--
       |
       |-> BIOS. ------> Ø .. 1
                         2 .. 3
                          I D   LETTERS
        MOVEA.L (A5),AØ         ;POINT TO BIOS
        MOVE.W  4(AØ),DØ        ;GET SYSTEM ID CHARS

        CURRENT LETTER DEFS
        FØ = FORCE CPU 1           MØ = MIZAR 91ØØ
        F2 = FORCE CPU 2           DØ = DATA SUD
        UØ = MOSTEK MATRIX 68      D4 = DY-4
        TØ = VME-1Ø                JØ = T.I. 51ØØ
        S4 = SAGE 2/4              VØ = VME 11Ø
        S6 = STRIDE 42Ø/44Ø/44Ø
        GØ = GMS
```

DO EXERCISE 1Ø-1 -- FIND YOUR PROCESSOR ID

3. Using TICS. for Timing

- * Finding out the tics per second
- * SYS[38]

- * MSYRAM parameter

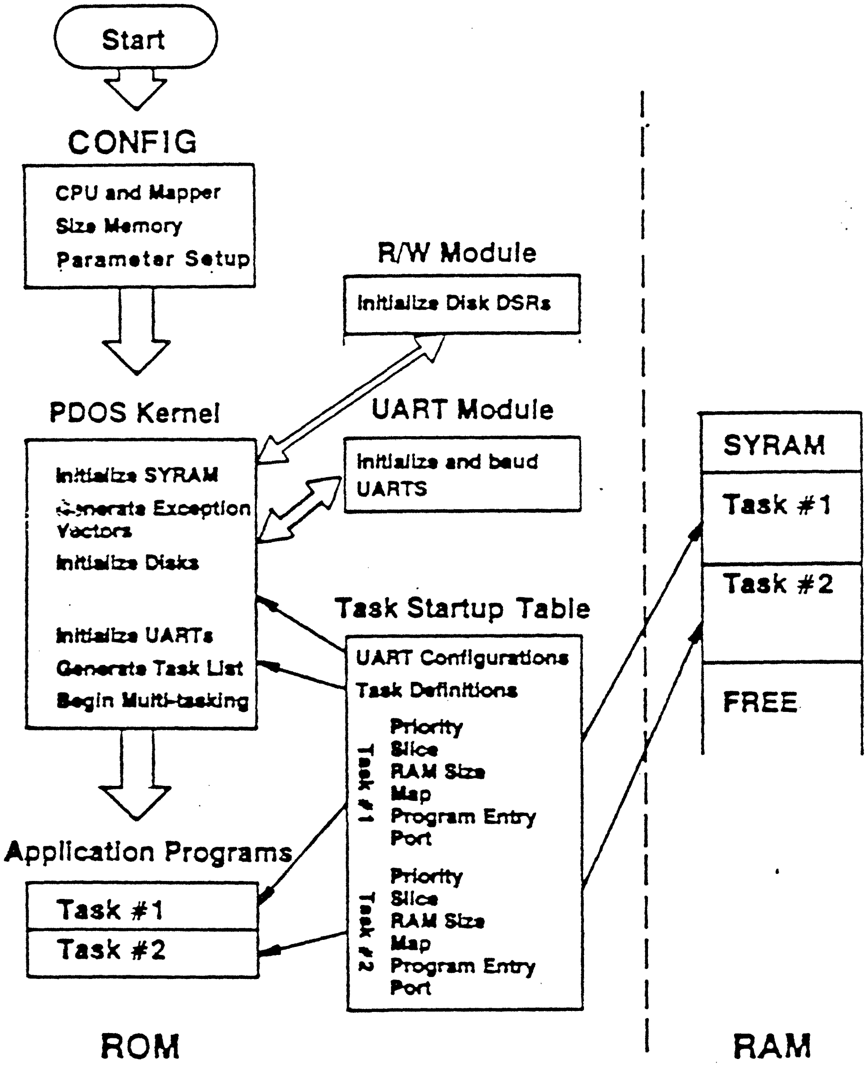
DO EXERCISE 1Ø-2 -- TIMING A LOOP

SESSION 11 -- HARDWARE INTERFACE

GOAL:

You should become acquainted with where to start on interfacing a new character or block device to PDOS.

PDOS SYSTEM INITIALIZATION



NOTES:

1. Task devices.

- Support device directly in task space.
- No operating system intergration needed

2. Driver Files -- Chapter 7

- Disk resident
- Limited to 252 bytes of code (1 PDOS sector)
- See PDOS Reference Manual Chapter 7

3. Basic I/O system -- BIOS

4. Character devices (UARTS) -- Chapter 8

- BIOSU File PDOS Reference Manual Section 8.2
- Device entries
- Get character
- Put character
- Baud port
- Reset port
- Read status
- 4 Unquie types supported

5. Block devices (DISK, R/W Sector) -- Chapter 8

- BIOSW File PDOS Reference Manual Section 8.3
- Interface
- Write Sector
- Read Sector
- Init Disks
- Disk Off
- Error message lists

SESSION 12 -- PDOS RUN MODULES

GOAL:

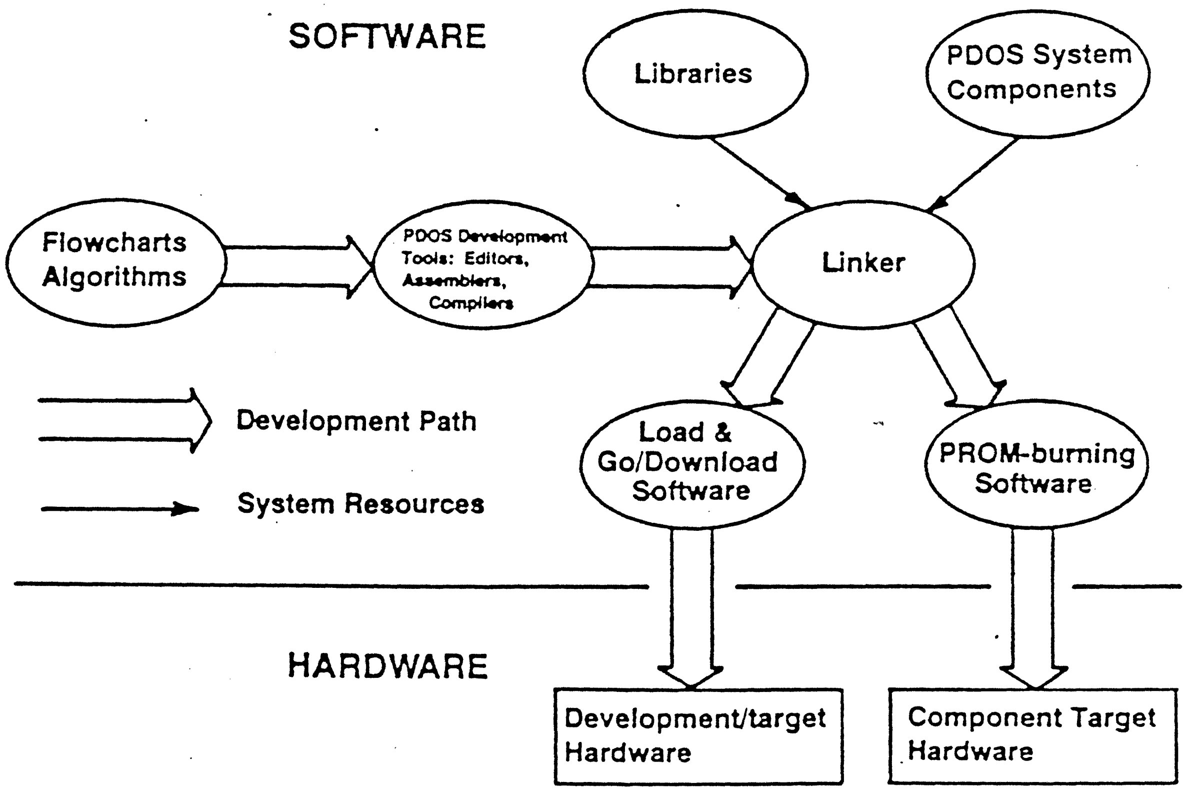
You should be able to build down an application into EPROM for use in a target system.

NOTES:

1. Run module licensing

- * Royalty per CPU
- * Pay only on parts of PDOS used
- * Paid up and source availble

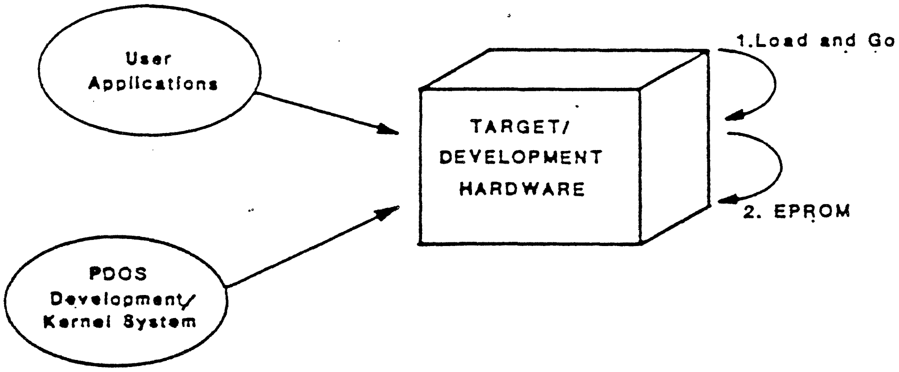
2. Development Cycle Review

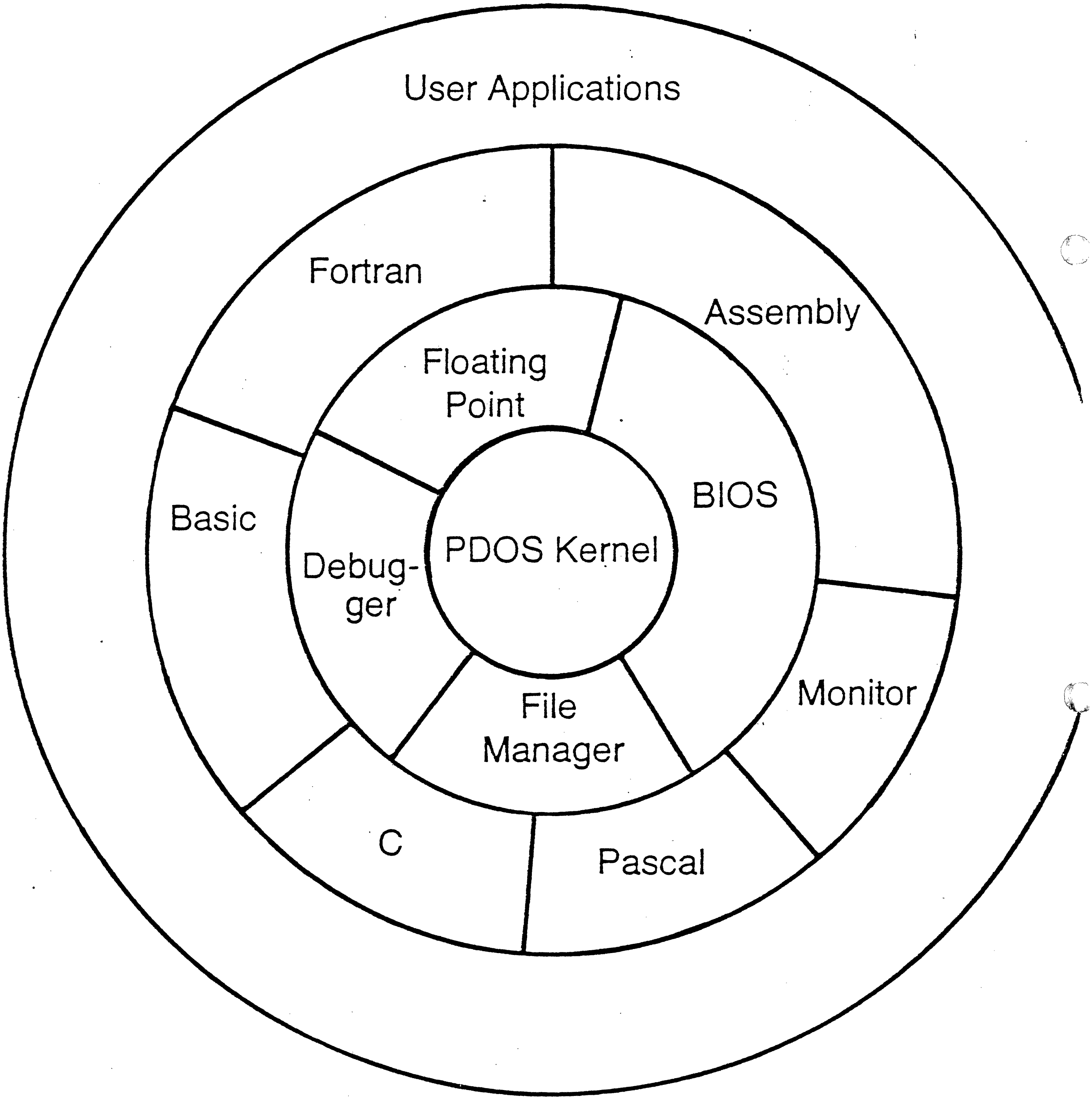

SOFTWARE
Libraries
PDOS System Components
Flowcharts Algorithms
PDOS Development Tools: Editors, Assemblers, Compilers
Linker
Development Path
System Resources
Load & Go/Download Software
PROM-burning Software
HARDWARE
Development/target Hardware
Component Target Hardware

3. Build Down Approach

- * PDOS is complete system for a single CPU
- * In order to extract only the part you need a license to build run modules.

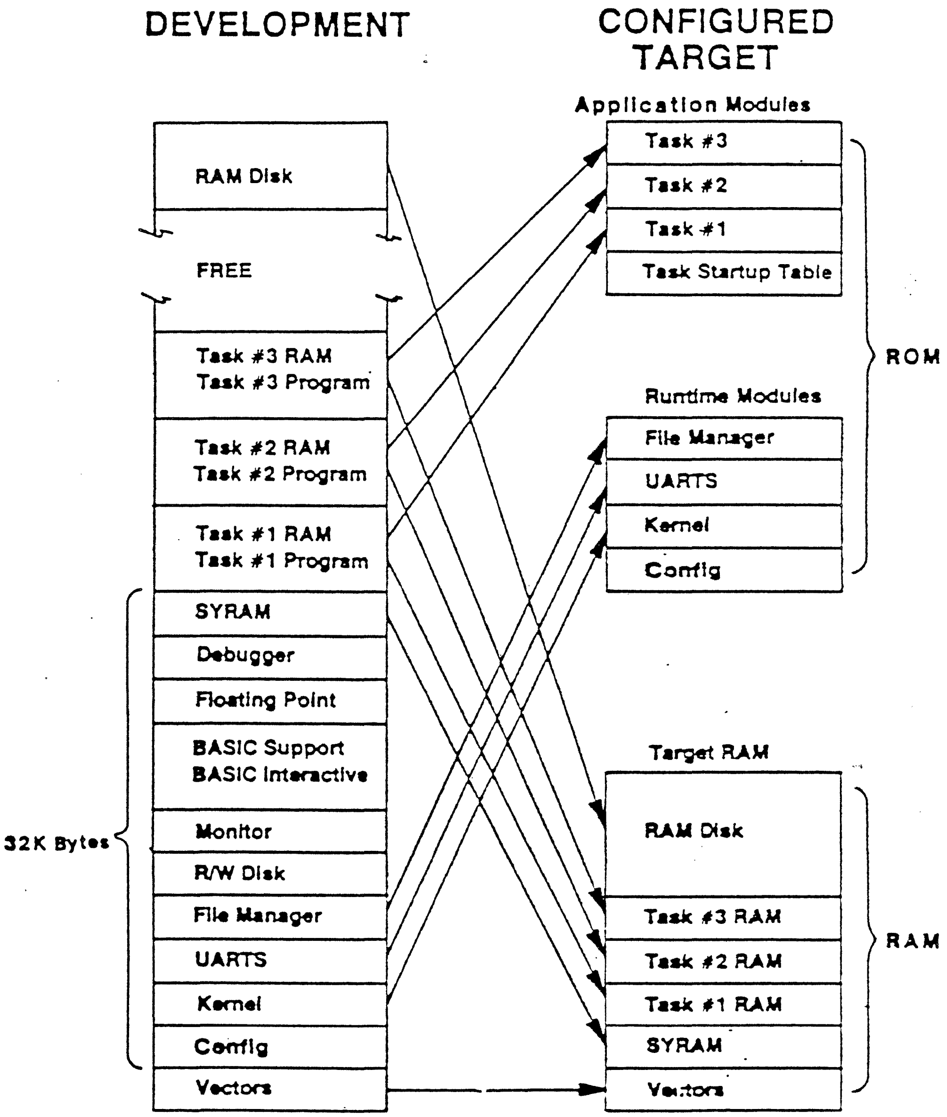
"BUILD-DOWN" SOFTWARE DEVELOPMENT




User Applications
Fortran
Assembly
Floating Point
BIOS
Basic
Debugger
PDOS Kernel
Monitor
File Manager
C
Pascal

4. Demonstration of building a run module system

- * Go over Run Module prelimary document
- * Build a Run module system with exercises work on in class.



* Build a Run module system with exercises work on in class. Instructor demonstration.

SESSION 13 -- PDOS SYSTEM GENERATION AND INSTALLATION

GOALS:

1. You should learn how to configure PDOS for various numbers of TASK Files and other parameters.
2. You will have the opportunity to install PDOS on a target system.

NOTES:

1. Chapter 2 instructions.

- Jumpers
- EPROMs

2. The MSYRAM file.

- User configurable parameters

```
IFUDF   NTB      :NTB EQU 32       ;NUMBER OF TASKS (128 MAXIMUM)
IFUDF   NTM      :NTM EQU 32       ;NUMBER OF TASK MESSAGES
IFUDF   TMZ      :TMZ EQU 64       ;TASK MESSAGE SIZE
IFUDF   NCB      :NCB EQU 8        ;NUMBER OF ACTIVE CHANNEL BUFFERS
IFUDF   NFS      :NFS EQU 32       ;NUMBER OF FILE SLOTS
IFUDF   NEV      :NEV EQU 1Ø       ;NUMBER OF DELAYED EVENTS
IFUDF   NPS      :NPS EQU 16       ;NUMBER OF I/O PORTS
IFUDF   NBC      :NBC EQU 4        ;NUMBER OF BASIC CALL ENTRIES
IFUDF   P2P      :P2P EQU 6        ;INPUT BUFFER SIZE (2^P2P)
IFUDF   MMZ      :MMZ EQU 1<<2Ø*8 ;MAXIMUM MEMORY SIZE (16M MAX.)
```

- How to change value with MASM switch
  - MASM FBIOS:SR/NCB=1Ø,MBIOS

3. Re-assembling and Re-linking the New System.

```
* xDOSGEN

IF &1=Q.GT LINK
MASM MSYRAM:SR,#MSYRAM
MASM xBIOS:SR,#MBIOS
MASM xBIOSU:SR,#MBIOSU
MASM xBIOSW:SR,#MBIOSW
IF &Ø.RC
GT LINK
LINK
QLINK
Z
DEFINE SYRAM,9ØØØ
SECTION Ø,8ØØ
IN MBIOS
IN MBIOSU
IN MBIOSW
IN MPDOS
IN MSYRAM
MAP UFS
MAP UFS,#xMAP
SY
OUTPUT #xDOS
END
QUIT
RC
```

4. Booting the System.

```
* To try new version:
  LO xDOS
  xBOOT

* To build new boot disks use:
  MMKBT
```

DO EXERCISE 13-1 -- INSTALLING PDOS

SESSION 14 -- PDOS CUSTOMER SERVICES

GOALS:

1. You should understand how Eyring can help you in the future.
2. You will be asked to complete an evaluation survey.
3. Possible future classes will be discussed.

NOTES:

- * PDOS Hotline support.
- * Manuals available.
- * Future classes.
- * Application notes.
- * Consulting.

DO EXERCISE 14-1 -- PDOS TRAINING SEMINAR EVALUATION

SESSION 15 -- TOUR OF EYRING

GOAL:

Let you see the Eyring facilities.

NOTES:

1. Software & Systems Marketing Division

Besides PDOS, Eyring markets a number of internally developed software systems:

* The Dynix library automation system.
* The Triton facility management system.
* STAT/1ØØØ -- a statistical analysis package.
* CAD -- computer aided police and emergency dispatch.
* A loan collection system.

2. Custom Software & Systems Development Division

This large division comprises nearly 1ØØ programmers and performs custom software and systems development on contract for large customers and the U.S. government.

3. Communications Division

Eyring has developed two proprietary and classified antennas which are being marketed to the U.S. military. They represent many years of development and are breakthroughs in portability and nuclear hardness.

4. Energy Research Division

Eyring has formed a subsidiary, Hydrocarbons, Inc. to handle the continuing development and marketing of a coal liquefaction process developed at Eyring. This process allows many valuable liquids to be removed from the coal. The remaining char can then be burned and produce the same amount of energy as the original coal would have produced.

5. Technical Services Division

Because of its varied interests and expertise, Eyring has the resources to contract its services to organizations and the U.S. government. Many of these services are currently being rendered at Hill Air Force Base in Ogden, Utah.

Other Eyring offices: Salt Lake City, Utah; Clearfield, Utah.

Exercises

EXERCISE 2-1 -- BOOTING PDOS

Work in groups according to the system to which your terminal is connected. Depending on your system, boot PDOS. (Note: <> are used to denote specific keys, for example: [CR] is a return key).

Force CPU-1/2
1) Turn power on.
2) System will be booted and other terminals started.

VME/1Ø
1) Turn key fully to the right.
2) Turn on power.
3) After TEN BUG message type a '[CR]'.
4) Insert boot diskette.
5) Type 'BO 2[CR]' in response to '>' prompt.
6) System will be booted and other terminals started.

STRIDE
1) Turn on power.
2) On master terminal type 'IH PDOS[CR]' in response to '>' prompt.
3) System will be booted and other terminals started.

Sage II or Sage IV
1) Put boot disk into drive Ø.
2) Turn on power.
3) System will be booted and other terminals started.

GMS system
1) Turn power on.
2) System will be booted and other terminals started.

Mostek
1) Turn power on.
2) Place boot disk in.
3) Type a '[CR]'.
4) In response to prompt, type 'LO 4,1ØØØØ,11ØØØ[CR]'.
5) Next, type 'ES 1ØØØØ[CR]'
6) System will be booted and other terminals started.

EXERCISE 2-2 -- HELP / ID

Enter the ID command and fill in the blanks.

```
     PDOS/68ØØØ  R________   ___/__/___
     ERII, copyright 1983
     _______________________________________
     DATE= ___/__/___
     TIME= ___:__:__
```

Type HE[CR].

You should see the possible help subjects.

Type HE MONITOR

You should see a list of monitor commands.

EXERCISE 2-3 -- VALID FILENAMES

Check which of the following are valid filenames:

[ ] THISISAFILNAME

[ ] 1FILENAME

[ ] FILE NAM

[ ] FILE.NAM

[ ] THE:NAME

[ ] A1234567:EXT

EXERCISE 2-4 -- FILE COMMANDS

Use the MF command to create 2 files. Call the first file NAME:1 and the second file NAME:2. Remember to use the '#' to auto-create your files.

Into file 1 enter the text:

```
THIS IS A TEST OF MF[CR]
ON FILE 1.[CR]
[ESC]
```

Into file 2 enter the text:

```
THIS IS A TEST OF MF[CR]
ON FILE 2.[CR]
[ESC]
```

Set the attributes of both files to 'TX' type.

Type out the files to your screen with the SF command.

Define a third file called NAME:3 with 1Ø blocks.

Rename NAME:3 to NAME:4

Copy NAME:1 to NAME:4

Append NAME:2 to NAME:4

Type out file NAME:4. What do you see?

Define file 1NAME. What happened?

List all files that start with NAME with any extension. What do you see?

Do a DM command to delete all files of the format NAME:*

List all files that start with NAME with any extension. What do you see?

EXERCISE 2-5 -- DISK COMMANDS

Find out how much disk space is on your disk.

Free= 1) ________/ 2) ________

Used= 3) ________/ 4) ________

What do the four numbers mean?

1)

2)

3)

4)

List all files on level 1

List all files on level 2

List all files on disk

EXERCISE 2-6 -- MEMORY COMMANDS

Type AM, how much is free? ______

Type FM 2

Type AM, how much is free? ______

Type GM

Type AM, how much is free? ______

EXERCISE 2-7 -- COMMAND LINE EDITING

Recall the last lines type by pressing ^A several times. What do you notice?

Do a LS command for Level 1.

Recall the line and edit it to list level 2, then execute the command.

Define a file called NAME. Now recall the command and using line edit rename the file to NAME:1; then delete the file.

EXERCISE 3-1 -- SETTING TERMINAL TYPE

Following the instructions on page 13-31 of the reference manual, set your terminal type using the MTERM program.

EXERCISE 3-2 -- MJEDY HELP KEY

Enter MJEDY from the PDOS monitor.

Bring up the help screen on your terminal by pressing [ESC] followed by the ^F key. The menu will be displayed until any other key is pressed.

EXERCISE 3-3 -- BASIC EDITING

Enter the following assembly program:

```
START   XPMNC   MESØ1           ;ASK FOR NUMBER
        XGLU
        XCD
        MULU    #1ØØ,D1
        XPMCMESØ2               ;X 1ØØ
        XCBD
        XPLC
        XPLC
        BRA.S   START
MESØ1   DC.B    $ØA,$ØD,'ENTER YOUR NUMBER = ',Ø
MESØ2   DC.B    ' x 1ØØ = ',Ø
        EVEN
        END     START
```

Now using the arrow and rubout keys move the cursor to the following lines and correct the errors. The bold lines denote error lines.

```
START   XPMC    MESØ1           ;ASK FOR NUMBER
        XGLU
        XCDB
        MULU    #1ØØ,D1
        XPMC    MESØ2           ;X 1ØØ
        XCBD
        XPLC
 (DELETE LINE)
        BRA.S   START
*
MESØ1   DC.B    $ØA,$ØD,'ENTER YOUR NUMBER = ',Ø
MESØ2   DC.B    ' x 1ØØ = ',Ø
        EVEN
        END     START
```

Your program should now appear as follows:

```
START   XPMC    MESØ1           ;ASK FOR NUMBER
        XGLU
        XCDB
        MULU    #1ØØ,D1
        XPMC    MESØ2           ;X 1ØØ
        XCBD
        XPLC
        BRA.S   START
*
MESØ1   DC.B    $ØA,$ØD,'ENTER YOUR NUMBER = ',Ø
MESØ2   DC.B    ' x 1ØØ = ',Ø
        EVEN
        END     START
```

EXERCISE 3-4 -- FILE INTERACTION

Save your created text in a filename using NAME and a :SR extension.

Quit the MJEDY editor.

Re-enter the editor with a GO command.

Save your created text again only instead of typing the filename, press the ^W key twice to recall the last filename used.

Clear the buffer.

Quit the MJEDY editor.

Now re-enter the editor by retrieving the file from the monitor.

You should now have your file in the editor buffer.

EXERCISE 3-5 -- GLOBAL SEARCH AND REPLACE

This exercise will change all 'MES' strings to 'TEX' strings.

Type ^D to begin the macro. You should notice the macro word appear on the status line in place of the time prompt.

Type ^T to place the cursor at the top of the text.

Type ^SMES[ESC]. This will locate the first string.

Press the [DEL] key 3 times to rub out the string 'MES'. Next type in the string 'TEX'.

Type a ^D to end the macro. You should note that the macro word prompt is removed from the screen and the time prompt reappears.

To replace all remaining occurances of 'MES', type [ESC]^Z. You will be prompted with Execute macro '. You enter the number -1 followed by an [ESC]. The macro will execute until the search fails.

If you had wanted to execute the macro once, then you should have typed a ^E instead of the [ESC]^Z command.

EXERCISE 3-6 -- INPUT / OUTPUT OF MACROS

First, save your macro that you defined in Exercise 3-5 into a filename. Use NAME for the filename with a '1:MAC' extension.

Now redo Exercise 3-5 redefining the macro to replace all 'TEX' back to 'MES'. Use the ^E execution on the macro.

Now save this macro out to disk using NAME again for the filename but with a '2:MAC' extension.

You should now be able to recall the first macro and convert 'MES' to 'TEX' and then recall the second macro and convert 'TEX' back to 'MES'.

When you are done, leave the file with the 'MES' text done.

Write the file out.

EXERCISE 3-7 -- BLOCK CUT AND PASTE

Go to the top of the file.

Insert the pointer.

Go to the second line.

Type a ^U to save the text.

Use the ^\ command to cut the text.

Restore the text by the ^A command.

EXERCISE 3-8 -- RUNNING MASM

Run MASM to assemble your program file that you created in part A with MJEDY. For the object output you may use the same name as the source with no extension. For the list file use a ':LST' extension. Remember to prefix these files with a '#' to auto create them. Your screen should be similar to the following when you are done.

```
68K PDOS Assembler R2.6e 1Ø/17/84
ERII, COPYRIGHT 1983
SRC=NAME:SR
OBJ=#NAME
LST=#NAME:LST
ERR=
XRF=
END OF PASS 1  [1 ERROR]
1/4x Ø/ØØØØØØØ8:4AFC4AFC          MUL #1ØØ,D1
END OF PASS 2  [1 ERROR]
```

You will notice that 1 error occurred. This is an illegal opcode instruction. Correct the error with MJEDY by changing the 'MUL' instruction with 'MULU'. Then repeat the MASM step.

Since there are no 'XREF's you can execute this program by typing the command 'NAME' in response to the PDOS prompt. Execute the program and try a few sample numbers.

To return to the PDOS monitor, type an [ESC].

EXERCISE 3-9 -- EXTERNAL LINKAGE

Remove the text strings from your source file and declare the labels for the text strings to be XREF.

```
*
MESØ1   DC.B    $ØA,$ØD,'ENTER YOUR NUMBER = ',Ø
MESØ2   DC.B    ' x 1ØØ = ',Ø
        EVEN
```

Create a new file with the text strings and declare the string labels to be XDEF. Name the new file NAME with a '1:SR' extension.

HINT: Use a pointer in MJEDY and ^O to output the text strings from the original file to the new file and ^\ extension to delete the text from the original.

Reassemble both parts using a ':OBJ' for the object files.

EXTRA: You may want to place a TTL and IDNT in the program and try some of the other directives.

Compare the results by looking at the listing files.

Also type out the object to the screen and translate the tagged object format.

EXERCISE 3-1Ø -- USING QLINK

Execute QLINK from the PDOS monitor.

Input the 2 object modules that you created for exercise 3-9.

Display the map on the screen.

Output the link as a SYfile under as NAME (no extension).

Now execute the file NAME.

The screen should be like the following:

```
1,Ø>QLINK
PDOS 68k Quick Linker 11/Ø1/84
ERII, Copyright 1983
*IN NAME:OBJ
ENTRY ADDRESS=ØØØØØØØØ
*IN NAME1:OBJ
*SYFILE
*MAP

INPUT FILE MAP:
INDEX FILE NAME    TYP IDNT  R  V  DATE   TIME  SECTION ADDRESSES
  1   NAME:OBJ                                Ø/ØØØØØØØØ ØØØØØØ15
  2   NAME1:OBJ                               Ø/ØØØØØØ16 ØØØØØØ37

SECTION GROUPS: NONE

OVERFLOW REFERENCE VALUES: NONE

SECTION        BASE        LOWEST       HIGHEST
   Ø         ØØØØØØØØ     ØØØØØØØØ     ØØØØØØ38

UNRESOLVED EXTERNAL DEFINITIONS: NONE

UNRESOLVED EXTERNAL REFERENCES: NONE
*OUTPUT #NAME
*END
SY FILE: BASE=$ØØØØØØØØ, LENGTH=56
*Q
```

EXERCISE 4-1 -- PROCEDURE COMMAND FILE

PART A:

Use MJEDY to create a procedure file that will assemble a file. Name the file NAME with a ':DOA' extension.

Your command file will pass to the assember 1 argument which is to be the main portion of the file name. You will supply the ':SR', ':OBJ', and ':LST' extensions as part of the procedure command text. Remember to use the '#' on the OBJ and LST files to auto create them.

Hint: For the source part of MASM: &1:SR.

End the file with the RC command to close only the current command file.

Exit the editor. Set the AC attribute and reassemble your two program files.

PART B:

Use MJEDY to create a procedure file that will link the two object files output from part A. Name the filename NAME with a ':DOL' extension.

Output the files to a file with no extension in SYfile format.

The input filenames will be obtained from input arguments.

The output filename will be the same as argument 1.

Display the map after the files have been input.

Relink your program the object files.

EXERCISE 4-1 (Continued)

PART C:

Create a procedure file that has three arguments, the first is for the word 'LINK' or 'ASSM', the second and third are for the filenames (i.e NAME and NAME1). Give the procedure file an ':ALL' extension.

This procedure file will call the procedure file you created for part A if argument &1 = ASSM. It will call the procedure file you created for part B if argument &1=LINK. If argument &1 is blank then both parts are to be done.

HINT: The following psedocode will show you the logic. You will need to use the 'IF' and 'GT' function of the monitor.

```
IF &1=LINK THEN GOTO LINK
IF &1=ASSM THEN GOTO ASSM
NAME:DOA &2,&2,&3
NAME:DOA &3,&2,&3
```

CAUTION: THE ARGUMENTS &Ø..&9 ARE GLOBAL TO ALL PROCEDURE NESTINGS. THE ABOVE WILL CHANGE &1 TO THE VALUE OF &2 AND THEN TO &3. YOU MUST PASS IN ARGUMENTS THAT NEED TO BE PRESERVED. HENCE 'NAME:DOA &2,&2,&3' IS USED TO SAVE &2 AND &3.

```
   GOTO LINK
LINK
   NAME:DOL &2,&3
   STOP
ASSM
   NAME:DOA &2,,&3
   IF &3.NAME:DOA &3
   STOP
```

Test the 3 options of the procedure file.

EXERCISE 4-2 -- SY$STRT FILE

Display the SY$STRT file on your screen.

EXERCISE 5-1 -- DEBUG

Load the program NAME into memory with the LO command and then enter the debugger.

Type the Help key and view the list of commands.

Disassemble the first 1Ø bytes of code.

Search first 1Ø bytes of task memory for a $AØ8Ø. Execute the program with a break point at the MULU instruction. When the break point is reached, dump the registers.

Display memory from +Ø,+4Ø.

Exit to the PDOS monitor.

EXERCISE 5-2 -- DEBUG APPLICATION NOTE

Read the application note and work on some of the examp

EXERCISE 6-1 -- LT AND BP COMMANDS

Type the LT command and note the the output ports for your task.

Type only the BP command note the different information for your tasks port.

My U1P is ____________

My U2P is ____________

My U1P is a type _____ port set at Base ___________ at a

baud rate of ___________.

EXERCISE 6-2 -- PRINT A FILE

Work in teams and find your partners U1P port number. Select each other's port as your unit 2 device. Now take turns and copy the file NAME:SR to the TTA driver. It should appear on your partners terminal.

Next, select output to both your unit 1 and unit 2 devices, then do a LT command. Note the results. Set output to unit 2 and do a LT command. (Note: you will have no chars appear on your terminal). Next, reset back to unit 1.

Spool your unit 2 to a file called NAME:LOG. Select output to both unit 1 and unit 2. Do a LT command and a BP command. Reselect unit 1, then reset your spool file.

Type out the log file to your terminal.

Although output was to a partner's terminal, a printer could have been connected to the port for hard copy output.

EXERCISE 7-1 -- PROCEDURE FILES

Run the procedure file created for Exercise 4-1 Part C to assemble and relink your program as a background task. After you have created your background task, do a LT command to note the father-son relationship. What is your son tasks number?________

EXERCISE 7-2 -- PROCEDURE FILES AS TASKS WITH RC

Create a procedure file named NAME:1 that will do a LT command only. Have the task output to your output port. (This can be done by assigning the task your output port number with the CT command).

Create the task. You should note that the task will do a LT to your terminal. Do a LT command and you will note that the background task is suspended on event 96. (NOTE: Your screen display may be garbaged due to the intermix of characters from the son task and your task.)

Kill the backgound task. Change the procedure file to have a RC or RS at the end. Re-create the task and then do a LT command. You should note that your background task has killed itself.

EXERCISE 7-3 -- THE TASK CONTROL BLOCK

Use the LT command to find your task control block address.

My task block address is ______________________________

Use the debugger to display your task control block.

EXERCISE 7-4 -- MULTI-TASK EVENT SYNCHRONIZATION

Using the MJEDY editor and MASM create the following program use NAME3:SR for the filename. For the program use an event number that is your task number +64 (the task number can be obtained from the LT command).

```
TEVENT   EQU      64+???           ;FOR ??? USE TASK # FROM LT COMMAND
*
START    MOVEQ.L #-TEVENT,D1      ;CLEAR THE EVENT TO Ø
         XSEF                      ;RESET EVENT PAGE 5-91
*
LOOP     MOVEQ.L #TEVENT,D1       ;WAIT FOR EVENT
         XSUI                      ;SUSPEND ON EVENT
         XPMC     MESS1                ;WRITE TO UNIT 1 THE MESSAGE
         BRA      LOOP
*
MESS1     DC.B $ØA,$ØD,'HELLO THERE THIS IS A VERY LONG MESSAGE',Ø
*
         END START
```

Assemble then execute the program as a task with the CT command and specify that the port be the same as your unit 1 port. (This information is all obtained from the list task command). Specify only 2k of memory.

Do the LT command and note the status of your son task. Is the task suspended on an event?

Use the EV command to set the event number, what happened?

Kill the task and verify the result with the list task command.

EXERCISE 7-5 -- TASK PRIORITIES

You should have noted that the string printed intermixed with the prompt message. This is due to the round robin scheduling of PDOS. The problem could be solved by allowing the suspended task to run at a higher priority than your task. Your task priority is defaulted to 64. This can be verified with the LT command.

Recreate the task you ran for Exercise 7-4 at a higher priority than your task and set the event flag.

Change the priority of the son task with the TP command so that the task will run at a lower priorty than your task and set the event flag.

What did you note about the command prompt and string:

Higher Priority________________________

Lower Priority ________________________

Kill the son task.

EXERCISE 7-6 -- MULTI-TASK MESSAGES

Use MJEDY to create a procedure file that will send to your task a message. Remember to place a RC as the last command in the file and to set the file attribute to AC. Run the procedure file as a son task.

Type the carriage return.

What happened?

EXERCISE 7-7 -- TIMER TASK

Redo the program created for exercise 7-4 and add a timeout loop. Save the changes into file NAME5:SR

```
TEVENT  EQU     64+???        ;FOR ??? USE TASK # FROM LT COMMAND
*
START   MOVE.L  #-TEVENT,D1   ;CLEAR THE EVENT TO Ø
        XSEF                  ;RESET EVENT PAGE 5-91
*
LOOP    MOVE.L  #1ØØ*5,DØ     ;WAIT ABOUT 5 SECONDS
        MOVEQ.L #-128,D1      ;USE THE LOCAL EVENT
        XDEV                  ;SET UP A DELAY ENTRY
        LSL.W   #8,D1         ;SHIFT EVENT OVER 1 BYTE
        ADDI.B  #TEVENT,D1    ;WAIT FOR EVENT
        XSUI                  ;SUPPEND ON EVENT
        CMPI.B  #TEVENT,DØ
          BNE.S LAB1
        XPMC     MESS1            ;WRITE TO UNIT 1 THE MESSAGE
*
        BRA     LOOP
*
LAB1    XPMC    MESS2
        BRA     LOOP
*
MESS1   DC.B $ØA,$ØD,'HELLO THERE THIS IS A VERY LONG MESSAGE',Ø
MESS2   DC.B $ØA,$ØD,'TIME OUT',Ø
        EVEN
*
        END START
```

Run as a son task with 2kb of memory and output port same as your tasks. What appears on your screen?

Now set the event flag. What appears on your screen?

Kill the task.

EXERCISE 8-1 -- MDDMAP / MDDUMP

Use MDDMAP to verify that your disk is OK. Note the header information about the disk. Note the starting sector number of one of your text files.

Now run MDDUMP and display sector Ø on the same Disk. Verify some of the header information from MDDMAP with MDDUMP sector Ø. Now dump your text file to the screen by entering the sector number. Enter Alter mode and change a few bytes in your text file. CAUTION: Do NOT change any file links -- the first 4 bytes of the sector. Write the sector back to the disk and Edit the text file with MJEDY. Do you see the changes?

EXERCISE 8-2 --MLEVEL

Execute MLEVEL and list the files on your disk.

EXERCISE 8-3 -- MDLOOK / MDSAVE

Execute MDLOOK to display the possible files on your disk. Select one that looks like a text file and save the text into a file called NAME:TMP. Display NAME:TMP on your console.

EXERCISE 8-4 -- RAM DISK AND MTRANS

Type the RD command to see if you have a RAM disk on your system. If you do, then use MTRANS to transfer all files of NAME:@ to the RAM disk.

Set your disk search list to be units 8 and your current unit. Delete the files of NAME:@ from the RAM disk. Reset the disk list to your previous unit.

EXERCISE 8-5 -- DIRECT DISK ACCESS

Write the following program in file NAME6:SR to read the first sector from one of your source files (The starting sector number can be obtained from the LS command).

```
DISKN    EQU      ???   ;THE DISK NUMBER
SECTN    EQU      $???  ;THE SECTOR NUMBER (REMEMBER THE $ FOR HEX)
*
START    MOVEQ.L  #DISKN,DØ
         MOVE.W   #SECTN,D1
         LEA.L    BUFF(PC),A2
         XRSE
         XEXT
*
BUFF     DS.B     256
         END      START
```

Assemble the program with a list file. From the list file find out the relative address of BUFF (This is on page 2 under DEFINED SYMBOLS).

Now execute the file.

Next use debug (PB command) to look at the buffer. This is done a by doing a memory dump like +e,+1Øe. The 'e' should be the value of the buffer relative address from the list file.

EXERCISE 1Ø-1 -- FIND YOUR PROCESSOR ID

Write a small program to output your processor ID.

Use the following PDOS primitives: XPCL, XPCC, XEXT.

First output a carriage return line feed -- XPCL

Get the processor ID into register DØ

Then output the processor ID -- XPCC (NOTE: the characters will be reversed).

Then output a carriage return line feed -- XPCL

Exit the program -- XEXT

ALTERNATE:

Use BASIC to output the ID using the SYS[] and MEMW functions.

EXERCISE 1Ø-2 -- TIMING A LOOP

Write the following program to time a loop:

```
***********************************************************
*         SAMPLE TIMING COOP.
***********************************************************
*
COUNT     EQU       100000              ;100000 LOOP ITERATIONS
TICS.     EQU       $ØØ14               ;32 BIT COUNTER
*
START     XPMC      MESØ1               ;'START'
          MOVE.L    #COUNT,D6           ;GET COUNTER
          MOVE.L    TICS.(A5),D7        ;GET TICS
*
***********************************************************
*         START OF TEST
***********************************************************
*
LOOP      YOUR CODE HERE
          SUBQ.L    #1,D6               ;DONE?
            BGT.S LOOP                  ;N
*
***********************************************************
*         END OF TEST
***********************************************************
*
TEND      SUB.L     TICS.(A5),D7        ;GET TIME
          NEG.L     D7
          MOVE.L    D7,D1
          XCBM      MESØ2
          MOVEA.L   A1,AØ
*
TEND2     TST.B     (AØ)+               MOVE TO END
               BNE.S TEND2              ;N
          MOVE.B    -(AØ),1(AØ)         ;Y
          MOVE.B    -(AØ),1(AØ)
          MOVE.B    -(AØ),1(AØ)
          MOVE.B    #'.',(AØ)
          XPLC                          ;OUTPUT TIME
          XEXT
*
MESØ1     DC.B      $ØA,$ØD,'BENCHMARK TIMER'
          DC.B      $ØA,$ØD,'START',$ØA,$ØD,Ø
MESØ2     DC.B      'END, TIME=',Ø
          END       START
```

Execute the program.

EXERCISE 13-1 -- INSTALLING PDOS

The installation will proceed on the systems that are available for the seminar.

EXERCISE 14-1 -- PDOS TRAINING SEMINAR EVALUATION

NAME______________________________________DATE_______________

Please take the time to answer the questions on this evaluation. The answers will aid us in improving and developing future courses for PDOS.

YOUR BACKGROUND

NONE

EXTENSIVE

		1	2	3	4	5
1.	My previous experience with PDOS was	□	□	□	□	□
2.	My realtime experience is	□	□	□	□	□
3.	My 68ØØØ processor experience is	□	□	□	□	□
4.	My 99ØØ processor experience is	□	□	□	□	□
5.	My hardware experience is	□	□	□	□	□

THE SEMINAR PRESENTATION

		STRONGLY DISAGREE			STRONGLY AGREE	
		1	2	3	4	5
1.	The instructor was enthusiastic when presenting course material.	□	□	□	□	□
2.	The instructor's use of examples helped to get points across in the lecture.	□	□	□	□	□
3.	The instructor appeared receptive to new ideas and others' viewpoints.	□	□	□	□	□
4.	The instructor generally stimulated class discussion.	□	□	□	□	□
5.	The instructor attempted to cover too much material.	□	□	□	□	□
6.	The instructor generally presented the material too rapidly.	□	□	□	□	□
7.	The instructor appeared to relate the course concepts in a systematic manner.	□	□	□	□	□
8.	The instructor's lecture presentations made for easy note-taking.	□	□	□	□	□
9.	You feel that this course challenged you intellectually.	□	□	□	□	□

1Ø. You have become more competent in this area due to this course.	□	□	□	□	□
11. The direction of the course was adequately outlined.	□	□	□	□	□
12. The course has definite objectives which were attained.	□	□	□	□	□
13. The course is well organized and prepared.	□	□	□	□	□
14. The class sessions are generally instructive and meaningful.	□	□	□	□	□

SESSION RATINGS

	POOR			EXCELLENT	
	1	2	3	4	5
SESSION 1 -- PDOS OVERVIEW	□	□	□	□	□
SESSION 2 -- GETTING STARTED	□	□	□	□	□
SESSION 3 -- A DEVELOPMENT SESSION	□	□	□	□	□
SESSION 4 -- ADVANCED MONITOR COMMANDS	□	□	□	□	□
SESSION 5 -- CHARACTER I/O	□	□	□	□	□
SESSION 6 -- DEBUG	□	□	□	□	□
SESSION 7 -- PDOS TASKING	□	□	□	□	□
SESSION 8 -- PDOS FILE MANAGER	□	□	□	□	□
SESSION 9 -- LANGUAGES	□	□	□	□	□
SESSION 1Ø -- PDOS INTERNALS	□	□	□	□	□
SESSION 11 -- HARDWARE INTERFACE	□	□	□	□	□
SESSION 12 -- PDOS RUN MODULES	□	□	□	□	□
SESSION 13 -- PDOS SYSTEM GENERATION AND INSTALLATION	□	□	□	□	□
SESSION 14 -- PDOS CUSTOMER SERVICES	□	□	□	□	□

I. List the experiences in the course which were of most value and of least value to you.

MOST VALUE	LEAST VALUE

II. What are your specific suggestions for improving the course?

III. What future courses would you like to see?

Answers to Exercises

ANSWERS TO SELECTED EXERCISES

The following are selected sample answers to the exercises. Three type styles will be used to explain the answers where interaction between the computer and you are needed. They are:

Ø,2>	Output from the computer.
RN NAME1:SR,1	Input for you to enter.
This is a comment	Explanatory notes.
SAMPLE OF EXERCISE 2-3	Section headings.

Since your system may be set up differently than the system these exercises were worked out on, you may need to make allowances for minor differences in the output from the computer. This will become more necessary as the exercises move to file access, tasking and advanced concepts.

SAMPLE OF EXERCISE 2-3

Check which of the following are valid filenames:

[ ] THISISAFILNAME

[ ] 1FILENAME

[X] FILE NAM

[X] FILE.NAM

[ ] THE:NAME

[X] A1234567:EXT

Comments:

THISISAFILENAME	Too long
1FILENAME	Starts with a number
FILE NAM	OK, but will need () to be used w/monitor
FILE.NAM	OK, but remember '.' is not for extension
THE:NAME	Extension is more then 3 characters
A1234567:EXT	OK

SAMPLE OF EXERCISE 2-4

```
Ø,2>MF #NAME:1[CR]
THIS IS A TEST OF MF[CR]
ON FILE 1.[CR]
[ESC]
Ø,2>MF #NAME:2[CR]
THIS IS A TEST OF MF[CR]
ON FILE 2.[CR]
[ESC]
Ø,2>SA NAME:1,TX[CR]
Ø,2>SA NAME:2,TX[CR]
Ø,2>SF NAME:1[CR]
THIS IS A TEST OF MF
ON FILE 1.
Ø,2>SF NAME:2[CR]
THIS IS A TEST OF MF
ON FILE 2.
Ø,2>DF NAME:3,1Ø[CR]
Ø,2>RN NAME:3,NAME:4[CR]
Ø,2>CF NAME:1,NAME:4[CR]
Ø,2>AF NAME:2,NAME:4[CR]
Ø,2>SF NAME:4[CR]
THIS IS A TEST OF MF
ON FILE 1.
THIS IS A TEST OF MF
ON FILE 2.
Ø,2>DF 1NAME[CR]
PDOS ERR 5Ø Invalid name
```

```
Ø,2>LS NAME@:@[CR]
```

Your listing may vary from the following in terms of Disk=, Files=, Lev, sect, Date created, Last update. The listing should be the same for Name:ext, type and size should be the same.

```
Disk=CLASSWORK/Ø                                   Files=58/128
Lev   Name:ext        Type      Size      Sect  Date created     Last update
 1    NAME:1          TX         1/1      Ø3D4 17:37 Ø4/Ø4/85 17:45 Ø4/Ø4/85
 1    NAME:2          TX         1/1      Ø3D5 17:39 Ø4/Ø4/85 17:45 Ø4/Ø4/85
 1    NAME:4          TX C       1/1Ø     Ø3D6 17:46 Ø4/Ø4/85 17:47 Ø4/Ø4/85
Ø,2>DM NAME:*[CR]
Delete NAME:1;1/Ø? (Y/N/A)Y
Delete NAME:2;1/Ø? (Y/N/A)Y
Delete NAME:4;1/Ø? (Y/N/A)Y
Ø,2>LS NAME@:@[CR]
```

Your listing may vary from the following.

```
Disk=CLASSWORK/Ø                                   Files=55/128
Lev   Name:ext        Type      Size      Sect  Date created     Last update
```

For the rest of the exercises it is assumed that a [CR] is used to enter commands.

SAMPLE OF EXERCISE 2-5

```
Ø,2>SP
```

Your example may vary from the following.

```
Free=1388,1388
Used=556/962
```

FOUR NUMBERS

1) Total Available

2) Total Available Contiguous

3) Total Currently Used

4) Total Allocated

```
Ø,2>LS ;1
```

Your listing may vary from the following.

```
Disk=CLASSWORK/Ø                                Files=55/128
Lev  Name:ext        Type         Size       Sect  Date created     Last update
 1   APLY            TX            6/7       Ø139 13:5Ø 12/18/84 15:54 12/18/84
 1   DAN             SY            1/1       ØØA4 Ø2:17 ØØ/ØØ/ØØ Ø2:27 ØØ/ØØ/ØØ
```

```
Ø,2>LS ;2
```

Your listing may vary from the following.

```
Disk=CLASSWORK/Ø                                Files=55/128
Lev  Name:ext        Type         Size       Sect  Date created     Last update
Ø,2>LS ;@
```

SAMPLE OF EXERCISE 2-6

```
Ø,2>AM
Free=Ø
Ø,2>FM 2
Addr=ØØØ7Ø8ØØ
Ø,2>AM
Free=2
Ø,2>GM
Ø,2>AM
Free=Ø
```

This number will vary.

SAMPLE OF EXERCISE 2-7

```
Ø,2>[^A]
AM[^A]
GM[^A]
AM[^A]
FM 2[^A]
AM[^A]
LS ;2[^A]
LS ;1[^A]
SP[^A]
LS NAME@:@[^A]
```

```
DM NAME[^A]
AM[^A][ESC]
Ø,2>LS ;1
```

Your listing may vary from the following

```
Disk=CLASSWORK/Ø                            Files=55/128
Lev  Name:ext       Type      Size     Sect  Date created    Last update
 1   APLY           TX         6/7     Ø139 13:5Ø 12/18/84 15:54 12/18/84
 1   DAN            SY         1/1     ØØA4 Ø2:17 ØØ/ØØ/ØØ Ø2:27 ØØ/ØØ/ØØ
Ø,2>[^A]
LS ;1[DEL]2[CR]
Disk=CLASSWORK/Ø                            Files=55/128
Lev  Name:ext       Type      Size     Sect  Date created    Last update

Ø,2>DF NAME[CR]
Ø,2>[^A]
DF NAME[^H][^H][^H][^H][^H][^H][^H]RN[^F][^F][^F][^F][^F],NAME:1
Ø,2>[^A]
RN NAME,NAME:1[^H][^H][^H][^H][^H][^H][DEL][DEL][DEL][DEL][DEL][^H][^H]
[^H]DL[CR]
Ø,2>
```

SAMPLE OF EXERCISE 3-1

```
Ø,2>MTERM L
```

SAMPLE OF EXERCISE 3-2

```
Ø,2>MJEDY
```

Your screen should clear and the clock will appear in the bottom right hand corner of the screen

```
[ESC][^F]
```

Your screen will clear then the help screen will appear will appear. Type any character to return to the edit screen.

SAMPLE OF EXERCISE 3-3

Type in the program as shown and make the corrections

SAMPLE OF EXERCISE 3-4

To save text

```
[^W]NAME:SR[ESC]V
```

To quit MJEDY

```
[^Q]V
```

To re-enter MJEDY

```
Ø,2>GO
```

To save text again

```
[^W][^W][ESC]V
```

To clear the buffer

```
[^N]V
```

To quit MJEDY

```
[^Q]V
```

To re-enter MJEDY and get file from the PDOS monitor

```
Ø,2>MJEDY NAME:SR
```

Your file should be displayed on the screen

SAMPLE FOR EXERCISE 3-9

File NAME:SR should look like the following

```
        IDNT 1,Ø                    ;FOR EXTRA
        XREF MESØ1,MESØ2
*
START   XPMC    MESØ1               ;ASK FOR NUMBER
        XGLU
        XCDB
        MULU    #1ØØ,D1
        XPMC    MESØ2               ;X 1ØØ
        XCBD
        XPLC
        BRA.S   START
*
        END     START
```

File NAME1:SR should look like the following

```
        IDNT    1,Ø
        XDEF    MESØ1,MESØ2
*
MESØ1   DC.B    $ØA,$ØD,'ENTER YOUR NUMBER = ',Ø
MESØ2   DC.B    ' x 1ØØ = ',Ø
        EVEN
        END
```

Assemble both parts

```
Ø,2>MASM NAME:SR,#NAME:OBJ,#NAME:LST
68K PDOS Assembler R2.6e 1Ø/17/84
ERII, Copyright 1983
SRC=NAME:SR
OBJ=#NAME:OBJ
LST=#NAME:LST
ERR=
XRF=
END OF PASS 1
END OF PASS 2
Ø,2>MASM NAME1:SR,#NAME1:OBJ,#NAME1:LST
68K PDOS Assembler R2.6e 1Ø/17/84
ERII, Copyright 1983
SRC=NAME1:SR
OBJ=#NAME1:OBJ
LST=#NAME1:LST
ERR=
XRF=
END OF PASS 1
END OF PASS 2
```

Compare the results by looking at the listing files. Page 2 of the listings will show in NAME that MESØ1 and MESØ2 are 'X'(External) type and in NAME1 that MESØ1 and MESØ2 are 'D' (Defined) type. The **** are also used in NAME to indicate that the value of the location is unknown at assembly time.

```
Ø,2>SF NAME:LST

                                                  68K PDOS Assembler 1Ø/17/84
PAGE: 1         Ø7:16 Ø4/Ø5/85                    FILE: NAME:SR,CLASSWORK

 1                                                XREF.1 MESØ1,MESØ2
 2                                        *
 3  Ø/ØØØØØØØØ:AØ8C****                   START   XPMC     MESØ1          ;ASK FOR
 4  Ø/ØØØØØØØ4:AØ8Ø                               XGLU
 5  Ø/ØØØØØØØ6:AØ56                               XCDB
 6  Ø/ØØØØØØØ8:C2FCØØ64                           MULU.w   #1ØØ,D1
 7  Ø/ØØØØØØØC:AØ8C****                           XPMC     MESØ2          ;X 1ØØ
 8  Ø/ØØØØØØ1Ø:AØ5Ø                               XCBD
 9  Ø/ØØØØØØ12:AØ8A                               XPLC
1Ø  Ø/ØØØØØØ14:6ØEA                               BRA.S    START
11                                        *
12  Ø/ØØØØØØ16:       Ø/ØØØØØØØØ                  END      START

                                                  68K PDOS Assembler 1Ø/17/84
PAGE: 2         Ø7:16 Ø4/Ø5/85                    FILE: NAME:SR,CLASSWORK

DEFINED SYMBOLS:

MESØ1       X   X/ØØØØØØØØ   MESØ2     X   X/ØØØØØØØØ   START    Ø/ØØØØØØØØ

EXTERNAL DEFINITIONS: NONE

EXTERNAL REFERENCES:

MESØ1       X   X/ØØØØØØØØ   MESØ2     X   X/ØØØØØØØØ

UNDEFINED SYMBOLS: NONE

UNREFERENCED SYMBOLS: NONE
```

```
Ø,2>SF NAME1:LST

                                            68K PDOS Assembler 1Ø/17/84
PAGE: 1           Ø     Ø4/Ø5/85            FILE: NAME1:SR,CLASSWORK

 1           ØØØØØØØØØØØØØØ17               XDEF     MESØ1,MESØ2
 2  Ø/ØØØØØØØØ:ØAØD454E5445522Ø MESØ1       DC.B     $ØA,$ØD,'ENTER YOUR
 3           594F5552204E554D
 4           424552203D2ØØØ
 5  Ø/ØØØØØØ17:2Ø782Ø313Ø3Ø2Ø3D MESØ2       DC.B     ' x 1ØØ = ',Ø
 6           2ØØØ
 7  Ø/ØØØØØØ21:ØØ                           EVEN
 8  Ø/ØØØØØØ22:                             END

                                            68K PDOS Assembler 1Ø/17/84
PAGE: 2           Ø7:16 Ø4/Ø5/85            FILE: NAME1:SR,CLASSWOR

DEFINED SYMBOLS:

MESØ1      D    Ø/ØØØØØØØØ    MESØ2      D    Ø/ØØØØØØ17

EXTERNAL DEFINITIONS:

MESØ1      D    Ø/ØØØØØØØØ    MESØ2      D    Ø/ØØØØØØ17

EXTERNAL REFERENCES: NONE

UNDEFINED SYMBOLS: NONE

UNREFERENCED SYMBOLS: NONE
```

SAMPLE OF EXERCISE 3-1Ø

```
Ø,2>QLINK
PDOS 68k Quick Linker 1Ø/1Ø/84
ERII, Copyright 1983
*INPUT NAME:OBJ
ENTRY ADDRESS=ØØØØØØØØ
*INPUT NAME1:OBJ
*MAP

INPUT FILE MAP:
INDEX FILE NAME       TYP IDNT   R  V  DATE  TIME   SECTION ADDRESSES
  1   DAN:OBJ                                    Ø/ØØØØØØØØ ØØØØØØ15
  2   DAN1:OBJ                                   Ø/ØØØØØØ16 ØØØØØØ37

SECTION GROUPS: NONE

OVERFLOW REFERENCE VALUES: NONE

SECTION        BASE        LOWEST      HIGHEST
   Ø         ØØØØØØØØ     ØØØØØØØØ     ØØØØØØ38

UNRESOLVED EXTERNAL DEFINITIONS: NONE

UNRESOLVED EXTERNAL REFERENCES: NONE
*SYFILE
*OUTPUT #NAME
*END
SY FILE: BASE=$ØØØØØØØØ, LENGTH=56
*QUIT
Ø,2>NAME
ENTER YOUR NUMBER = 23[CR] x 1ØØ = 23ØØ
ENTER YOUR NUMBER = 1[CR] x 1ØØ = 1ØØ
ENTER YOUR NUMBER = [ESC]
Ø,2>
```

SAMPLE OF EXERCISE 4-1

PART A:

```
Ø,2>MJEDY
```

Create your file with the following text then write the text to filename NAME:DOA. Quit MJEDY

```
MASM &1:SR,#&1:OBJ,#&1LST
RC
```

Set the attribute to AC type and re-assemble your two source programs

```
Ø,2>SA NAME:DOA,AC
Ø,2>NAME:DOA NAME
Ø,2>MASM NAME:SR,#NAME:OBJ,#NAME:LST
68K PDOS Assembler R2.6e 1Ø/17/84
ERII, Copyright 1983
SRC=NAME:SR
OBJ=#NAME:OBJ
LST=#NAME:LST
ERR=
XRF=
END OF PASS 1
END OF PASS 2
Ø,2>RC
Ø,2>NAME:DOA NAME1
Ø,2>MASM NAME1:SR,#NAME1:OBJ,#NAME1:LST
68K PDOS Assembler R2.6e 1Ø/17/84
ERII, Copyright 1983
SRC=NAME1:SR
OBJ=#NAME1:OBJ
LST=#NAME1:LST
ERR=
XRF=
END OF PASS 1
END OF PASS 2
Ø,2>RC
Ø,2>
```

PART B:

Use MJEDY to create the following file

```
QLINK
INPUT &1:OBJ
INPUT &2:OBJ
MAP
SYFILE
OUTPUT #&1
END
```

```
QUIT
RC
```

Don't forget to set the AC attribute.

```
Ø,2>SA NAME:DOL,AC
```

Re-link your program object files.

```
Ø,2>NAME:DOL NAME,NAME1
Ø,2>QLINK
PDOS 68k Quick Linker 1Ø/1Ø/84
ERII, Copyright 1983
*INPUT NAME:OBJ
ENTRY ADDRESS=ØØØØØØØØ
*INPUT NAME1:OBJ
*MAP

INPUT FILE MAP:
INDEX FILE NAME        TYP IDNT  R  V  DATE TIME  SECTION ADDRESSES
  1   NAME:OBJ                                    Ø/ØØØØØØØØ ØØØØØØ15
  2   NAME1:OBJ                                   Ø/ØØØØØØ16 ØØØØØØ37

SECTION GROUPS: NONE

OVERFLOW REFERENCE VALUES: NONE

SECTION        BASE         LOWEST       HIGHEST
   Ø         ØØØØØØØØ      ØØØØØØØØ      ØØØØØØ38

UNRESOLVED EXTERNAL DEFINITIONS: NONE

UNRESOLVED EXTERNAL REFERENCES: NONE
*SYFILE
*OUTPUT #NAME
*END
SY FILE: BASE=$ØØØØØØØØ, LENGTH=56
*QUIT
Ø,2>RC
Ø,2>
```

ART C:

se MJEDY to create the following file

```
F &1=LINK.GT LINK
F &1=ASSM.GT ASSM
AME:DOA &2,&2,&3
AME:DOA &3,&2,&3
T LINK
NK
ME:DOL &2,&3
:
```

```
ASSM
NAME:DOA &2,,&3
IF &3.NAME:DOA &3
RC
```

Set the attribute to AC type

```
Ø,2>SA NAME:ALL,AC
```

Test ASSM path

```
Ø,2>NAME:ALL ASSM,NAME,NAME1
Ø,2>IF ASSM=LINK.GT LINK
Ø,2>IF ASSM=ASSM.GT ASSM
Ø,2>GT ASSM
NAME:DOA NAME,NAME,NAME1
NAME:DOA NAME1,NAME,NAME1
GT LINK
LINK
NAME:DOL NAME,NAME1
RC
ASSM
Ø,2>NAME:DOA NAME,,NAME1
Ø,2>MASM NAME:SR,#NAME:OBJ,#NAME:LST
68K PDOS Assembler R2.6e 1Ø/17/84
ERII, Copyright 1983
SRC=NAME:SR
OBJ=#NAME:OBJ
LST=#NAME:LST
ERR=
XRF=
END OF PASS 1
END OF PASS 2
Ø,2>RC
Ø,2>IF NAME1.NAME:DOA NAME1
Ø,2>NAME:DOA NAME1
Ø,2>MASM NAME1:SR,#NAME1:OBJ,#NAME1:LST
68K PDOS Assembler R2.6e 1Ø/17/84
ERII, Copyright 1983
SRC=NAME1:SR
OBJ=#NAME1:OBJ
LST=#NAME1:LST
ERR=
XRF=
END OF PASS 1
END OF PASS 2
Ø,2>RC
Ø,2>RC
```

Test LINK path

```
Ø,2>NAME:ALL LINK,NAME,NAME1
```

```
Ø,2>IF LINK=LINK.GT LINK
Ø,2>GT LINK
IF LINK=ASSM.GT ASSM
NAME:DOA NAME,NAME,NAME1
NAME:DOA NAME1,NAME,NAME1
GT LINK
LINK
Ø,2>NAME:DOL NAME,NAME1
Ø,2>QLINK
PDOS 68k Quick Linker 1Ø/1Ø/84
ERII, Copyright 1983
*INPUT NAME:OBJ
ENTRY ADDRESS=ØØØØØØØØ
*INPUT NAME1:OBJ
*MAP

INPUT FILE MAP:
INDEX FILE NAME        TYP IDNT  R  V   DATE  TIME  SECTION ADDRESSES
  1   NAME:OBJ                                      Ø/ØØØØØØØØ ØØØØØØ15
  2   NAME1:OBJ                                     Ø/ØØØØØØ16 ØØØØØØ37

SECTION GROUPS: NONE

OVERFLOW REFERENCE VALUES: NONE

SECTION        BASE        LOWEST       HIGHEST
   Ø         ØØØØØØØØ     ØØØØØØØØ     ØØØØØØ38

UNRESOLVED EXTERNAL DEFINITIONS: NONE

UNRESOLVED EXTERNAL REFERENCES: NONE
*SYFILE
*OUTPUT #NAME
*END
SY FILE: BASE=$ØØØØØØØØ, LENGTH=56
*QUIT
Ø,2>RC
```

Test both (null argument) path

```
Ø,2>NAME:ALL ,NAME,NAME1
Ø,2>IF =LINK.GT LINK
Ø,2>IF =ASSM.GT ASSM
Ø,2>NAME:DOA NAME,NAME,NAME1
Ø,2>MASM NAME:SR,#NAME:OBJ,#NAME:LST
68K PDOS Assembler R2.6e 1Ø/17/84
ERII, Copyright 1983
SRC=NAME:SR
OBJ=#NAME:OBJ
 ST=#NAME:LST
 RR=
 RF=
END OF PASS 1
END OF PASS 2
Ø,2>RC
Ø,2>NAME:DOA NAME1,NAME,NAME1
Ø,2>MASM NAME1:SR,#NAME1:OBJ,#NAME1:LST
68K PDOS Assembler R2.6e 1Ø/17/84
ERII, Copyright 1983
SRC=NAME1:SR
OBJ=#NAME1:OBJ
LST=#NAME1:LST
ERR=
XRF=
END OF PASS 1
END OF PASS 2
Ø,2>RC
Ø,2>GT LINK
LINK
Ø,2>NAME:DOL NAME,NAME1
Ø,2>QLINK
PDOS 68k Quick Linker 1Ø/1Ø/84
ERII, Copyright 1983
*INPUT NAME:OBJ
ENTRY ADDRESS=ØØØØØØØØ
*INPUT NAME1:OBJ
*MAP

INPUT FILE MAP:
INDEX FILE NAME     TYP IDNT   R  V   DATE   TIME  SECTION ADDRESSES
  1   NAME:OBJ                                     Ø/ØØØØØØØØ ØØØØØØ15
  2   NAME1:OBJ                                    Ø/ØØØØØØ16 ØØØØØØ37

SECTION GROUPS: NONE

OVERFLOW REFERENCE VALUES: NONE

SECTION        BASE        LOWEST      HIGHEST
   Ø         ØØØØØØØØ     ØØØØØØØØ     ØØØØØØ38

UNRESOLVED EXTERNAL DEFINITIONS: NONE

UNRESOLVED EXTERNAL REFERENCES: NONE
*SYFILE
*OUTPUT #NAME
*END
SY FILE: BASE=$ØØØØØØØØ, LENGTH=56
*QUIT
Ø,2>RC
```

SAMPLE OF EXERCISE 4-2

```
Ø,2>SF SY$STRT
```

Your start up file may be different

```
MTIME P,85
BP -3,1
LT
DT
RC
```

SAMPLE OF EXERCISE 5-1

These exercise answers will vary from your.

```
Ø,2>LT
Task Prt Tm   Event   Map Size  PC     SR    TB     BM     EM      PRT U1P U2P
 Ø/Ø 64  1    97/-128  Ø 28Ø  ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ  1/1 1/1 B/2
 1/Ø 64  1    98/-128  Ø 2ØØ  Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ  2/1 2/1 B/2
*2/Ø 64  1             Ø  32  ØØ23A6 2ØØ4 Ø69ØØØ Ø6AC46 Ø71ØØØ  7/1 7/1 B/2
```

In this example your are task 2. U1P is 7 and U2P is 11 ($B hex). U1P is a type 1 port.

```
Ø,2>BP
Port   Type   f_pi8dbs    Base     Rate  Task
#1      1     ØØØØØØØØ  FFFFC3Ø1  192ØØ  Ø
#2      1     ØØØØØØØØ  FFFFC311  192ØØ  1
#3      1     ØØØØ1ØØØ  FFFFC341  12ØØ
#4      1     ØØØØØØØØ  FFFFC351  192ØØ
#5      1     ØØØØØØØØ  FFFFC6Ø1  192ØØ
#6      1     ØØØØØØØØ  FFFFC611  192ØØ
#7      1     ØØØØØØØØ  FFFFC641  192ØØ  2
#8      1     ØØØØØØØØ  FFFFC651  192ØØ
#9      1     ØØØØØØØØ  FFFFC681  192ØØ
#1Ø     1     ØØØØØØØØ  FFFFC691  192ØØ
#11     2     ØØØØØØØØ  FFFFC1C1  192ØØ
```

Port 7 (U1P or task 2) is set at a base of $FFFFC641 and a baud rate of 192ØØ

```
Ø,2>
```

SAMPLE OF EXERCISE 5-2

For this exercise I will use Task 1 and 2 as partners. Task 1 is on port 2 and Task 2 is on Port 7 both are running at 192ØØ baud.

Task 2. Select Task 1's port 2 as your U2P

```
Ø,2>BP -2,192ØØ
```

Copy NAME:SR to your partners terminal.

```
Ø,2>CF NAME:SR,TTA
```

Your file text should appear on your partner's terminal. Select output to both unit 1 and unit 2 do a LT command.

```
Ø,2>UN 3
```

The following will appear on both terminals

```
Ø,2>LT
Task Prt Tm  Event    Map Size    PC    SR     TB      BM      EM     PRT U1P U2P
 Ø/Ø 64  1   97/-128 Ø   28Ø   ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ  1/1 1/1 B/2
 1/Ø 64  1   98/-128 Ø   2ØØ   Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ  2/1 2/1 B/2
*2/Ø 64  1           Ø    32   ØØ23A6 2ØØ4 Ø69ØØØ Ø6AC46 Ø71ØØØ  7/1 7/1 B/2
Ø,2>BP
Port   Type   f_pi8dbs    Base     Rate  Task
#1      1    ØØØØØØØØ  FFFFC3Ø1  192ØØ  Ø
#2      1    ØØØØØØØØ  FFFFC311  192ØØ  1
#3      1    ØØØØ1ØØØ  FFFFC341  12ØØ
#4      1    ØØØØØØØØ  FFFFC351  192ØØ
#5      1    ØØØØØØØØ  FFFFC6Ø1  192ØØ
#6      1    ØØØØØØØØ  FFFFC611  192ØØ
#7      1    ØØØØØØØØ  FFFFC641  192ØØ  2
#8      1    ØØØØØØØØ  FFFFC651  192ØØ
#9      1    ØØØØØØØØ  FFFFC681  192ØØ
#1Ø     1    ØØØØØØØØ  FFFFC691  192ØØ
#11     2    ØØØØØØØØ  FFFFC1C1  192ØØ
```

Set to unit 2

```
Ø,2>UN 2
```

Output will only appear on your partners terminal

```
Ø,2>LT
Ø,2>UN 1
```

Spool your unit 2 to a file called NAME:LOG. Select both unit 1 and unit 2 for output and do a LT and BP command. Re-select to unit 1, then reset your spool file

```
Ø,2>SU 2,NAME:LOG
Ø,2>UN 3
Ø,2>LT
Task Prt Tm   Event   Map Size    PC     SR     TB      BM      EM     PRT U1P U2P
 Ø/Ø 64  1    97/-128 Ø   28Ø   ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ   1/1 1/1 B/2
 1/Ø 64  1    98/-128 Ø   2ØØ   Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ   2/1 2/1 B/2
*2/Ø 64  1            Ø    32   ØØ23A6 2ØØ4 Ø69ØØØ Ø6AC46 Ø71ØØØ   7/1 7/1 B/2

Ø,2>BP
Port   Type   f_pi8dbs    Base     Rate   Task
#1      1     ØØØØØØØØ  FFFFC3Ø1  192ØØ   Ø
#2      1     ØØØØØØØØ  FFFFC311  192ØØ   1
#3      1     ØØØØ1ØØØ  FFFFC341  12ØØ
#4      1     ØØØØØØØØ  FFFFC351  192ØØ
#5      1     ØØØØØØØØ  FFFFC6Ø1  192ØØ
#6      1     ØØØØØØØØ  FFFFC611  192ØØ
#7      1     ØØØØØØØØ  FFFFC641  192ØØ   2
#8      1     ØØØØØØØØ  FFFFC651  192ØØ
#9      1     ØØØØØØØØ  FFFFC681  192ØØ
#1Ø     1     ØØØØØØØØ  FFFFC691  192ØØ
#11     2     ØØØØØØØØ  FFFFC1C1  192ØØ
Ø,2>UN 1
```

Reset your spool file.

```
Ø,2>SU Ø
```

Type out the log file to your terminal

```
Ø,2>SF NAME:LOG
was 1
Ø,2>LT
Task Prt Tm   Event   Map Size    PC     SR     TB      BM      EM     PRT U1P U2P
 Ø/Ø 64  1    97/-128 Ø   28Ø   ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ   1/1 1/1 B/2
 1/Ø 64  1    98/-128 Ø   2ØØ   Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ   2/1 2/1 B/2
*2/Ø 64  1            Ø    32   ØØ23A6 2ØØ4 Ø69ØØØ Ø6AC46 Ø71ØØØ   7/1 7/1 B/2
```

SAMPLE OF EXERCISE 6-1

The [CR] will be denoted in this sample to show when and where to type the return key

Load NAME into memory and enter the debugger

```
Ø,2>LO NAME[CR]
Ø;2>PB[CR]
```

Type the help key

```
H
AØ-7     A-reg               #            Mem IAC
B{#,a}   Lst/def break       #,#          Mem dump
DØ-7     D-reg               #,#+         Disassemble
{#}G     Go & break          #,#,#{WL}    Find B/W/L
L        Last dump           #(Ø-7        d(Ax)
O        Offset              #{+-}#       Hex +/-
P        PC
Q        Exit                 -            Open previous
R        Reg dump             LF           Open next
S        Status               +#            # + offset
T        Trace
U        Unit
W{s,sz}  Window               ^D           Disassemble
X        Set breaks & exit
Z        Reset
```

Disassemble the first 1Ø bytes of code.

```
+Ø,+1Ø+
Ø695ØØ/ØØØØ: AØ8C              Aline    $AØ8C
Ø695Ø2/ØØØ2: ØØ14AØ8Ø          ORI.B    #$8Ø,(A4)
Ø695Ø6/ØØØ6: AØ56              Aline    $AØ56
Ø695Ø8/ØØØ8: C2FCØØ64          MULU.W   #$ØØ64,D1
Ø695ØC/ØØØC: AØ8C              Aline    $AØ8C
Ø695ØE/ØØØE: ØØ1FAØ5Ø          ORI.B    #$5Ø,(A7)+
```

Search task memory for a $AØ8Ø

```
+Ø,+1Ø,AØ8ØW
  Ø695Ø4
```

Set a break point at the MULU instruction and execute the program.

```
B1,+8[CR]                           MULU.W  #$ØØ64,D1
G
ENTER YOUR NUMBER = 12[CR]
B> Ø695Ø8/ØØØ8: C2FCØØ64          MULU.W   #$ØØ64,D1
T> Ø695ØC/ØØØC: AØ8C              Aline    $AØ8C
```

When the break point is reached dump the registers

```
RREGISTER DUMP: PC=ØØØ695ØC  SR=ØØØØ  SS=ØØØ693BØ
DØ:ØØØØØØØØ ØØØØØ4BØ ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ
AØ:ØØØØØØØØ ØØØ69ØØ3 ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ ØØØØ9ØØØ ØØØ69ØØØ ØØØ71ØØØ
[ESC]
```

Dump memory from +Ø,+4Ø

```
+Ø,+4Ø[CR]
Ø695ØØ/ØØØØ: AØ8C ØØ14 AØ8Ø AØ56 C2FC ØØ64 AØ8C ØØ1F .......V...d....
Ø6951Ø/ØØ1Ø: AØ5Ø AØ8A 6ØEA ØAØD 454E 5445 522Ø 594F .P..`...ENTER YO
Ø6952Ø/ØØ2Ø: 5552 2Ø4E 554D 4245 522Ø 3D2Ø ØØ2Ø 782Ø UR NUMBER = . x
Ø6953Ø/ØØ3Ø: 313Ø 3Ø2Ø 3D2Ø ØØØØ ØAØD 3436 3ØØD 2Ø2B 1ØØ = ....46Ø. +
```

Return to the PDOS monitor

```
Q
Ø,2>
```

SAMPLE OF EXERCISE 7-1

Create a background son task to reassemble and re-link NAME

```
Ø,2>CT (NAME:ALL ,DAN,DAN1)
*Task #3
```

Do a LT command to note the father/son relationship

```
Ø,2>LT
Task Prt Tm   Event    Map Size    PC     SR     TB     BM     EM     PRT U1P U2P
 Ø/Ø 64  1    97/-128  Ø   28Ø   ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ  1/1 1/1 B/2
 1/Ø 64  1    98/-128  Ø   2ØØ   Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ  2/1 2/1 B/2
*2/Ø 64  1             Ø    48   ØØ23A6 2ØØ4 Ø5DØØØ Ø5EC46 Ø69ØØØ  7/1 7/1 Ø/Ø
 3/2 64  1             Ø    32   ØØ1584 2ØØØ Ø69ØØØ Ø6DBCØ Ø71ØØØ  Ø/Ø Ø/Ø Ø/Ø
```

You should note that task 3's father is task 2 and task 2 is the current task

SAMPLE OF EXERCISE 7-2

Create the NAME:1 as a procedure file with the following text

```
LT[CR]
```

Set the AC attribute

```
Ø,2>SA NAME:1,AC
```

Create the TASK using your console port (U1P) for output. We are assuming for this task that port 7 is U1P

```
Ø,2>CT NAME:1,,,7
,Ø>NAME:1*T
Ø,2>aLs Tk
T    EM      PRT U1P U2P
 Ø/Ø 64  1   97/-128   Ø   296   ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø55ØØØ  1/1 1/1 B/2
 1/Ø 64  1   98/-128   Ø   2ØØ   Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ  2/1 2/1 B/2
 2/Ø 64  1             Ø    48   ØØ23A6 2ØØ4 Ø5DØØØ Ø5EC46 Ø69ØØØ  7/1 7/1 Ø/Ø
*4/2 64  1             Ø    32   ØØ21EØ 2ØØØ Ø55ØØØ Ø555ØØ Ø5DØØØ  Ø/Ø 7/1 Ø/Ø
```

You will see an intermix of output on the screen from the son tasks LT command and your tasks. Now type a LT and note the son task number 4 is still defined and is waiting on event 96, the phantom input port event

```
Ø,2>LT
Task Prt Tm   Event    Map Size    PC     SR     TB     BM     EM     PRT U1P U2P
 Ø/Ø 64  1    97/-128  Ø   28Ø   ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ  1/1 1/1 B/2
 1/Ø 64  1    98/-128  Ø   2ØØ   Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ  2/1 2/1 B/2
*2/Ø 64  1             Ø    48   ØØ23A6 2ØØ4 Ø5DØØØ Ø5EC46 Ø69ØØØ  7/1 7/1 Ø/Ø
```

```
 4/2 64  1   96      Ø   32  ØØ21EØ 2ØØØ Ø55ØØØ Ø555ØØ Ø5DØØØ  Ø/Ø 7/1 Ø/Ø
```

Kill your son task.

```
Ø,2>KT 4
```

Change NAME:1 to have an RC or RS at the end and repeat the task creation and the LT command

```
Ø,2>CT NAME:1,,,7
,Ø>NAME:1*T
Ø,2>aLs Tk
T     EM     PRT U1P U2P
 Ø/Ø 64  1  97/-128  Ø  296  ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø55ØØØ  1/1 1/1 B/2
 1/Ø 64  1  98/-128  Ø  2ØØ  Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ  2/1 2/1 B/2
 2/Ø 64  1           Ø   48  ØØ23A6 2ØØ4 Ø5DØØØ Ø5EC46 Ø69ØØØ  7/1 7/1 Ø/Ø
*4/2 64  1           Ø   32  ØØ21EØ 2ØØØ Ø55ØØØ Ø555ØØ Ø5DØØØ  Ø/Ø 7/1 Ø/Ø
```

Do the LT command and you will note that the son task in not present.

```
Ø,2>LT
Task Prt Tm   Event   Map Size    PC     SR     TB      BM      EM    PRT U1P U2P
 Ø/Ø 64  1   97/-128 Ø   28Ø   ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ  1/1 1/1 B/2
 1/Ø 64  1   98/-128 Ø   2ØØ   Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ  2/1 2/1 B/2
*2/Ø 64  1           Ø    48   ØØ23A6 2ØØ4 Ø5DØØØ Ø5EC46 Ø69ØØØ  7/1 7/1 Ø/Ø
```

SAMPLE OF EXERCISE 7-3

Find the address of your task control block

```
Ø,2>LT
Task Prt Tm   Event   Map Size    PC     SR     TB      BM      EM    PRT U1P U2P
 Ø/Ø 64  1   97/-128 Ø   28Ø   ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ  1/1 1/1 B/2
 1/Ø 64  1   98/-128 Ø   2ØØ   Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ  2/1 2/1 B/2
*2/Ø 64  1           Ø    48   ØØ23A6 2ØØ4 Ø5DØØØ Ø5EC46 Ø69ØØØ  7/1 7/1 Ø/Ø
```

The Task control block for task 2 is at address $Ø5DØØØ to $Ø5DØØØ + $5ØØ.

Use debug to display your task control block. Your display will vary depending on the state of your task

```
Ø,2>PB[CR]
5DØØØ,5D5ØØ[CR]
Ø5DØØØ/FBØØ: 5432 ØØØØ ØØØØ ØØØØ 5Ø41 5379 ØØØØ 2FØ7 T2......PASy../.
Ø5DØ1Ø/FB1Ø: ØØØØ ØØØØ ØØØØ ØØ95 ØCØ2 A91B ØCØ2 A91B ................
Ø5DØ2Ø/FB2Ø: 5433 ØØØØ ØØØØ ØØØØ ØØØØ ØØ79 1ØØ4 2FØ8 T3.........y../.
Ø5DØ3Ø/FB3Ø: ØØØØ ØØØ9 ØØØ9 ØØAC ØCØ2 A91B ØCØ3 A91B ................
Ø5DØ4Ø/FB4Ø: 5433 ØØØØ ØØØØ ØØØØ 5Ø41 5379 ØØØ4 2F12 T3......PASy../.
Ø5DØ5Ø/FB5Ø: ØØØØ ØØØ2 ØØØ2 ØØ2B ØCØ3 A91B ØCØ3 A91B .......+........
Ø5DØ6Ø/FB6Ø: 5443 4F4D 5ØØØ ØØØØ ØØØØ ØØ79 8ØØ4 2F15 TCOMP......y../.
 5DØ7Ø/FB7Ø: ØØØØ ØØØ1 ØØØ1 ØØ83 ØCØ3 A91B ØCØ3 A91B ................
```

```
Ø5DØ8Ø/FB8Ø: 5445 4D5Ø ØØØØ ØØØØ ØØØØ ØØØ1 Ø2ØØ 2F17 TEMP........../.
Ø5DØ9Ø/FB9Ø: ØØØØ ØØ67 ØØØ2 ØØ51 ØE25 A953 ØA19 AA85 ...g...Q.%.S....
Ø5DØAØ/FBAØ: 5445 4D5Ø 31ØØ ØØØØ ØØØØ ØØØ1 Ø2ØØ 2F5Ø TEMP1........./P
Ø5DØBØ/FBBØ: ØØØØ ØØØB ØØØB ØØ5B ØA31 A953 ØCØ8 AA85 .......[.1.S....
Ø5DØCØ/FBCØ: 5445 4D5Ø 4FØØ ØØØØ ØØØØ ØØ28 ØØØ4 2F57 TEMPO......(../W
Ø5DØDØ/FBDØ: ØØØØ ØØØ2 ØØØ2 ØØ47 1Ø2C A96D 1Ø2D A96D .......G.,.m.-.m
Ø5DØEØ/FBEØ: 5445 5354 ØØØØ ØØØØ ØØØØ ØØ81 1ØØØ 2F5A TEST........../Z
Ø5DØFØ/FBFØ: ØØØØ ØØØØ ØØØØ ØØAØ ØBØF A91B ØBØF A91B ................
Ø5D1ØØ/FCØØ: 5Ø42 ØØ33 ØØ54 454D 5ØØØ 31ØØ 37ØØ 4441 PB.3.TEMP.1.7.DA
Ø5D11Ø/FC1Ø: 4E2C 4441 4E31 ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ N,DAN1..........
Ø5D12Ø/FC2Ø: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D13Ø/FC3Ø: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D14Ø/FC4Ø: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D15Ø/FC5Ø: 3ØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØFA ØØØØ ØØØØ Ø...............
Ø5D16Ø/FC6Ø: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D17Ø/FC7Ø: 5Ø42 ØØ4C 54ØØ 554E 2Ø33 ØØ53 552Ø 322C PB.LT.UN 3.SU 2,
Ø5D18Ø/FC8Ø: 5445 4D5Ø ØØ4D 4A45 4459 2Ø54 454D 5Ø31 TEMP.MJEDY TEMP1
Ø5D19Ø/FC9Ø: ØØ4C 54ØØ 474D ØØ4C 54ØØ 4B54 2Ø33 ØØ4B .LT.GM.LT.KT 3.K
Ø5D1AØ/FCAØ: 542Ø 34ØØ 4B54 2Ø35 ØØ43 ØØØØ 2ØØØ ØØØØ T 4.KT 5.C......
Ø5D1BØ/FCBØ: ØØØØ 2ØØØ 323Ø 333Ø ØØ31 423Ø ØØØØ ØØØØ ....ØØØØ.1BØ....
Ø5D1CØ/FCCØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D1DØ/FCDØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D1EØ/FCEØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D1FØ/FCFØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D2ØØ/FDØØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D21Ø/FD1Ø: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D22Ø/FD2Ø: ØØØ6 9ØØØ ØØØØ ØØØØ ØØØØ ØØ73 ØØØØ 13ØØ ...........s....
Ø5D23Ø/FD3Ø: ØØØØ ØØØØ FFFF ØØØ2 Ø851 ØC5Ø ØØØØ 21EØ .........Q.P..!.
Ø5D24Ø/FD4Ø: ØØØØ ØØØØ FFFF 8A6E ØØØF F4ØØ ØØØ5 D1ØØ .......n........
Ø5D25Ø/FD5Ø: FFFF 88Ø7 ØØØ7 ØØØ7 1ØØØ ØAØD 2Ø33 ØØØ9 ............ 3..
Ø5D26Ø/FD6Ø: F88A ØØØØ ØØØØ ØØØ9 F88A ØØØØ ØØØØ ØØØØ ................
Ø5D27Ø/FD7Ø: ØØØF ØØØØ FFØ4 Ø851 ØC5Ø ØØØØ ØØØA ØØØØ .......Q.P......
Ø5D28Ø/FD8Ø: ØØØØ ØØ1Ø ØØØØ ØØØF F8ØØ ØØØØ A75A FFFF .............Z..
Ø5D29Ø/FD9Ø: 88Ø7 ØØØ7 ØØØ7 ØFFC ØØØ7 ØFFC ØØØØ ØØ3D ...............=
Ø5D2AØ/FDAØ: ØØØ7 ØFF8 FFFF ØØØ4 ØØØØ 2ØØ4 ØØØØ ØØØD .......... .....
Ø5D2BØ/FDBØ: ØØØØ ØØ4E ØØØØ ØØØ7 FFFF FFFF ØØØØ 23A6 ...N..........#.
Ø5D2CØ/FDCØ: ØØØØ ØØØ4 ØØØØ ØØAF ØØØØ ØØ4E FFFF 88Ø7 ...r.......Z....
Ø5D2DØ/FDDØ: FFFF C541 ØØØØ 9ØØØ ØØØ5 DØØØ 2ØØØ ØØØØ ...A........ ...
Ø5D2EØ/FDEØ: 3626 FFFF ØØØ4 FFFF ØØØ4 ØØØØ 2ØØ4 ØØØØ 6&.......... ...
Ø5D2FØ/FDFØ: ØØØE ØØØØ ØØC2 ØØØØ ØØØ7 FFFF FFFF ØØØØ ................
Ø5D3ØØ/FEØØ: 23A6 ØØØØ 2ØØØ ØØØØ ØØE6 ØØØØ ØØ21 ØØØØ #... ........8..
Ø5D31Ø/FE1Ø: A65A ØØØ5 D1B3 ØØØØ AEF4 ØØØØ 9ØØØ ØØØ5 .Z..............
Ø5D32Ø/FE2Ø: DØØØ 2ØØØ ØØØ7 ØFFC ØØØØ ØØ2Ø ØØØØ ØØØØ .. .............
Ø5D33Ø/FE3Ø: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØ2 ØØØØ ØØØ7 ................
Ø5D34Ø/FE4Ø: 2ØØ4 ØØØØ ØØØØ 23A6 ØØØØ ØØ2Ø FFFF C641  .....#.... ...A
Ø5D35Ø/FE5Ø: ØØØØ ØBC4 ØØØØ 9158 ØØØ5 D1AE ØØØØ 91ØØ .......X........
Ø5D36Ø/FE6Ø: ØØØØ 9ØØØ ØØØ5 DØØØ 2ØØ4 ØØØØ 23Ø8 ØØØØ ........ ...#...
Ø5D37Ø/FE7Ø: 2Ø3A FFFF FFFF ØØØØ ØØØØ ØØØØ ØØ2C Ø7Ø1 .............,..
Ø5D38Ø/FE8Ø: ØØØØ 2ØØ4 ØØØØ ØØØØ 23A6 ØØØØ ØØ2Ø ØØØ5 .. .....#.... ..
Ø5D39Ø/FE9Ø: D39Ø ØØØ5 D1B4 ØØØØ 266B ØØØ5 D39C ØØØ5 ........&k......
Ø5D3AØ/FEAØ: D5ØØ ØØØØ 9ØØØ ØØØ5 DØØØ ØØØ4 ØØØØ 4EØC ..............N"
Ø5D3BØ/FEBØ: ØØØ5 D324 ØØØØ ØØØØ 2ØØ4 ØØØØ 23A6 ØØØØ ...$.... ...#...
```

```
Ø5D3CØ/FECØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D3DØ/FEDØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D3EØ/FEEØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D3FØ/FEFØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D4ØØ/FFØØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ 5ØØ2 ØØØØ ............P...
Ø5D41Ø/FF1Ø: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØ5 D1Ø3 ØØØ5 ................
Ø5D42Ø/FF2Ø: EC46 ØØØ7 1ØØØ ØØØ5 D5ØØ ØØØØ ØØØØ ØØØØ .F..............
Ø5D43Ø/FF3Ø: ØØØØ ØØ49 Ø42Ø ØØFA ØØØØ ØCØØ 9B59 FFØØ ...I. .......Y..
Ø5D44Ø/FF4Ø: Ø2Ø4 ØØØØ ØØØØ ØØØØ ØØØØ ØØØ2 ØØ2B Ø3Ø7 .............B..
Ø5D45Ø/FF5Ø: Ø2Ø3 Ø7ØØ ØØØØ ØØØØ Ø4ØØ ØØ51 ØØ53 ØØØØ ...........Q.S..
Ø5D46Ø/FF6Ø: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØ5 E93Ø ...............Ø
Ø5D47Ø/FF7Ø: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D48Ø/FF8Ø: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D49Ø/FF9Ø: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D4AØ/FFAØ: ØØØØ ØØØØ ØØØØ 9ØØØ ØØØ5 DØØØ ØØØ7 1ØØØ ................
Ø5D4BØ/FFBØ: ØØØØ ØØØ5 D5ØØ ØØØ5 D5ØØ ØØØ7 1ØØØ ØØØØ ................
Ø5D4CØ/FFCØ: ØØØ5 D5ØØ ØØØØ ØØØØ ØØØ5 DØØØ ØØØ5 D5ØØ ................
Ø5D4DØ/FFDØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D4EØ/FFEØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
Ø5D4FØ/FFFØ: ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ................
```

Return to the PDOS monitor.

```
Q
Ø,2>
```

SAMPLE OF EXERCISE 7-4

Type LT and find out your task number

```
Ø,2>LT
Task Prt Tm   Event   Map Size    PC     SR     TB      BM      EM    PRT U1P U2P
 Ø/Ø 64  1    97/-128 Ø   28Ø   ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ  1/1 1/1 B/2
 1/Ø 64  1    98/-128 Ø   2ØØ   Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ  2/1 2/1 B/2
*2/Ø 64  1            Ø    48   ØØ23A6 2ØØ4 Ø5DØØØ Ø5EC46 Ø69ØØØ  7/1 7/1 Ø/Ø
```

You will note that you are task 2. Use MJEDY and create NAME3:SR with the following text.

```
TEVENT   EQU      64+2              ;FOR 2 FOR TASK 2
*
START    MOVEQ.L #-TEVENT,D1        ;CLEAR THE EVENT TO Ø
         XSEF                       ;RESET EVENT PAGE 5-91
*
LOOP     MOVEQ.L #TEVENT,D1         ;WAIT FOR EVENT
         XSUI                       ;SUPPEND ON EVENT
         XPMC    MESS1              ;WRITE TO UNIT 1 THE MESSAGE
         BRA      LOOP
*
MESS1    DC.B $ØA,$ØD,'HELLO THERE THIS IS A VERY LONG MESSAGE',Ø
*
         END START
```

Assemble the Program

```
Ø,2>MASM NAME3:SR,#NAME3:OBJ
68K PDOS Assembler R2.6e 1Ø/17/84
ERII, Copyright 1983
SRC=NAME3:SR
OBJ=#NAME3:OBJ
LST=
ERR=
XRF=
END OF PASS 1
END OF PASS 2
```

List your task to find out what your U1P port number.

```
Ø,2>LT
Task Prt Tm   Event    Map Size    PC     SR     TB      BM      EM     PRT U1P U2P
 Ø/Ø 64  1    97/-128 Ø   28Ø  ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ  1/1 1/1 B/2
 1/Ø 64  1    98/-128 Ø   2ØØ  Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ  2/1 2/1 B/2
*2/Ø 64  1            Ø    8Ø  ØØ23A6 2ØØ4 Ø5DØØØ Ø61BCØ Ø71ØØØ  7/1 7/1 Ø/Ø
```

You are port 7. Now create the task with 2kb of memory on port 7.

```
Ø,2>CT NAME3:OBJ,2,,7

3:OBJ*Task #3
Ø,2>
```

Now list the tasks and see if your son task is suppended on the right event number.

```
Ø,2>LT
Task Prt Tm   Event    Map Size    PC     SR     TB      BM      EM     PRT U1P U2P
 Ø/Ø 64  1    97/-128 Ø   28Ø  ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ  1/1 1/1 B/2
 1/Ø 64  1    98/-128 Ø   2ØØ  Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ  2/1 2/1 B/2
*2/Ø 64  1            Ø    8Ø  ØØ23A6 2ØØ4 Ø5DØØØ Ø61BCØ Ø71ØØØ  7/1 7/1 Ø/Ø
 3/2 64  1    66      Ø     2  Ø5CDØ8 ØØØØ Ø5C8ØØ Ø5CD38 Ø5DØØØ  Ø/Ø 7/1 Ø/Ø
```

Set the event on and see what happens

```
Ø,2>EV 66
HELLO THERE THIS IS A VERY
Was ØLONG MESSAGE
```

You should note that the message from the son task is intermixed with the father task output. Kill the son task and verify that it is gone.

```
Ø,2>KT 3
Ø,2>LT
Task Prt Tm   Event    Map Size    PC     SR     TB      BM      EM     PRT U1P U2P
 Ø/Ø 64  1    97/-128 Ø   28Ø  ØØC2FØ ØØØØ ØØBØØØ ØØCC46 Ø51ØØØ  1/1 1/1 B/2
 1/Ø 64  1   98/-128 Ø   2ØØ   Ø72298 ØØØØ Ø71ØØØ Ø72EØA ØA3ØØØ   2/1 2/1 B/2
*2/Ø 64  1           Ø    8Ø   ØØ23A6 2ØØ4 Ø5DØØØ Ø61BCØ Ø71ØØØ   7/1 7/1 Ø/Ø
Ø,2>
```

SAMPLE OF EXERCISE 7-5

Recreate the task a higher priority then set the event flag

```
Ø,2>CT NAME3:OBJ,2,65,7
Ø,2>NAME3:OBJ
*Task #3
Ø,2>
Ø,2>NAME3:OBJ
Ø,2>EV 66
HELLO THIS IS A VERY LONG MESSAGE
Was Ø
```

Now lower the task to a lower priority and set the event flag

```
Ø,2>TP 3,63
Ø,2>EV 66
Was Ø
Ø,2>
HELLO THIS IS A VERY LONG MESSAGE
```

You should note that when the son task is higher priority that it completes its output to the console before the father task can output. When the son task is lower priority the father task completes its output first.

Kill the son task.

```
Ø,2>KT 3
```

SAMPLE OF EXERCISE 7-6

Create a procedure file NAME:4 that will return to your task a message

```
SM -1,Hello there from your son task!!
RC
```

Set the attributes to AC type

```
Ø,2>SA NAME:4,AC
```

Run the file and you should see the message when you type the [CR]

```
Ø,2>CT NAME:4,2
 Task 3
 ,2>[CR]
 Task 3: Hello there from your son task!!
```

SAMPLE OF EXERCISE 7-7

Using MJEDY redo the file to look like the following and save the changes into file NAME5:SR.

```
TEVENT  EQU      64+2               ;USE YOUR TASK NUMBER FROM LT COMMAND
*
START   MOVE.L   #-TEVENT,D1        ;CLEAR THE EVENT TO Ø
        XSEF                        ;RESET EVENT PAGE 5-91
*
LOOP    MOVE.L   #1ØØ*5,DØ          ;WAIT ABOUT 5 SECONDS
        MOVEQ.L  #128,D1            ;USE THE LOCAL EVENT
        XDEV                        ;SET UP A DELAY ENTRY
        LSL.W    #8,D1              ;SHIFT EVENT OVER 1 BYTE
        ADDI.B   #TEVENT,D1         ;WAIT FOR EVENT
        XSUI                        ;SUPPEND ON EVENT
        CMPI.B   #TEVENT,DØ
          BNE.S  LAB1
        XPMC     MESS1              ;WRITE TO UNIT 1 THE MESSAGE
        BRA      LOOP
*
LAB1    XPMC     MESS2
        BRA      LOOP
*
MESS1   DC.B $ØA,$ØD,'HELLO THERE THIS IS A VERY LONG MESSAGE',Ø
MESS2   DC.B $ØA,$ØD,'TIME OUT',Ø
        EVEN
*
        END START
```

Assemble the file then run it as a son task with 2kb of memory and output port the same as its father

```
Ø,2>MASM NAME5:SR,#NAME5:OBJ
68K PDOS Assembler R2.6e 1Ø/17/84
ERII, Copyright 1983
SRC=NAME5:SR
OBJ=#NAME5:OBJ
LST=
ERR=
XRF=
END OF PASS 1
1/7a Ø/ØØØØØØØE:728Ø             MOVEQ.L #128,D1   ;USE THE LOCAL EVENT
END OF PASS 2  [1 WARNING]
```

Create son task for program

```
Ø,2>CT NAME5:OBJ,2,,7
*Task #3
Ø,2>
TIME OUT
TIME OUT            Every 5 seconds TIME OUT appears unless event 66 set.
TIME OUTEV 66[CR] Set event 66
HELLO THERE THIS IS A VERY
Was ØLONG MESSAGE
Ø,2>
TIME OUT
TIME OUT
TIME OUTKT 3[CR] Kill the son task
Ø,2>
```