CHAPTER 12

STAND-ALONE RUN MODULE

The PDOS run time module is the stand-alone, multi-tasking kernel of PDOS. The run time executive is broken down into two modules plus three additional modules for BASIC support. These are linked together, along with user assembly language and/or BASIC programs, to yield an EPROMable, customized, stand-alone module which does not require disk interaction.

12.1 RUN MODULES

The PDOS run time module is the stand-alone, multi-tasking kernel of PDOS. The run time executive is broken down into two modules plus three additional modules for BASIC support. These are linked together, along with user assembly language and/or BASIC programs, to yield an EPROMable, customized, stand-alone module which does not require disk interaction.

Stand-alone, multi-tasking kernel

The first module is the PDOS executive program (R$MODA). It contains the interrupt and XOP vectors as well as many PDOS subroutines including character I/O, data conversions, and event processing. This module must be the first file of the link process.

R$MODA = PDOS executive module

The second module is the user configuration module (R$MODC). This module is used to configure the executive module and must be included in the link process. Such items as 9902 CRU bases, system clock interval, 9901 interrupt enables, BASIC assembly link addresses, interrupt mask, CRT control characters, and other system parameters are available for your customization. The execution and configuration modules when combined with an assembly application program constitute a simple run module.

R$MODC = User configuration module

The third module is a floating point package (R$MODF). This optional module is only included when the XOP floating point instructions are used, be it from assembly or a BASIC program. Several different floating point modules are available including a two word (R$MODF2) and hardware 1481 CPU version.

R$MODF = 3 word floating point routines

R$MODF2 = 2 word floating point routines

The fourth module is a stripped down BASIC interpreter (R$MODB). Both the edit and list portions of the interpreter have been deleted as well as all file I/O primitives. As was stated, the floating point module must be included if the BASIC interpreter is used.

R$MODB = BASIC interpreter

The final module contains the BASIC trigonometric functions arctangent, cosine, exponential, natural logarithm, sine, and tangent. If these functions are not used in the BASIC application program, then this module does not need to be including in the link process. A two word version (R$MODT2) allows further speed improvements and memory reduction.

R$MODT = 3 word trigonometric function

R$MODT2 = 2 word trigonometric function

12.2 R$MODA - EXECUTIVE MODULE

The Executive module is the heart of the stand-alone system. It contains the system clock and the multi-tasking routines. Also, many non-disk primitives are included such as console I/O and number conversion routines.

The Executive module must be the first module linked since it must reside at memory address >0000. The user configuration module (R$MODC) must also be linked to define special system parameters. The following is a list of supported PDOS primitives:

XBCP	BAUD CONSOLE PORT
XCBC	CHECK FOR BREAK CHARACTER
XCBD	CONVERT BINARY TO DECIMAL
XCBH	CONVERT BINARY TO HEX
XCBM	CONVERT BINARY TO DECIMAL WITH MESSAGE
XCDB	CONVERT DECIMAL TO BINARY
XCLS	CLEAR SCREEN
XERS	ERROR #, RTWP/SWAP
XFTD	FIX TIME & DATE INTO R0,R1
XGCC	GET CONSOLE CHARACTER CONDITIONAL
XGCR	GET CONSOLE CHARACTER
XGLB	GET LINE IN BUFFER
XGLM	GET LINE IN MONITOR BUFFER
XGLU	GET LINE IN USER BUFFER
XGML	GET MEMORY LIMITS
XGTM	GET TASK MESSAGE
XIPL	INTERRUPT DRIVEN PUT LINE
XLKT	LOCK TASK
XPBC	PUT USER BUFFER TO CONSOLE (*R9)
XPCC	PUT CHARACTER TO CONSOLE
XPCL	PUT CRLF TO CONSOLE
XPLC	PUT LINE TO CONSOLE
XPMC	PUT MESSAGE TO CONSOLE
XPSC	POSITION CURSOR
XRDT	READ DATE
XRTM	READ TIME
XRTS	READ TASK STATUS
XSEF	SET EVENT FLAG
XSER	ERROR RTWP/SWAP
XSTM	SEND TASK MESSAGE
XSUI	SUSPEND UNTIL INTERRUPT
XSWP	SWAP TO NEXT PROCESS
XSWR	RTWP/SWAP
XTAB	TAB
XTEF	TEST EVENT FLAG
XUDT	UNPACK DATE
XULT	UNLOCK TASK
XUTM	UNPACK TIME
XWDT	WRITE DATE
XWTM	WRITE TIME

Error 81 = Not supported

Primitives not supported:

XAPF	APPEND FILE
XCBP	CHECK FOR BREAK OR PAUSE
XCFA	CLOSE FILE WITH NEW ATTRIBUTES
XCHF	CHAIN FILE
XCLF	CLOSE FILE
XCPY	COPY FILE
XCTB	CREATE TASK BLOCK
XDFL	DEFINE FILE
XDLF	DELETE FILE
XERR	MONITOR ERROR CALL
XEXT	EXIT TO MONITOR
XFFN	FIX FILE NAME
XFFE	FIX FILE NAME W/ERROR
XGNP	GET NEXT PARAMETER
XISE	INIT SECTOR
XKTB	KILL TASK BLOCK
XLDF	LOAD FILE
XLFN	LOOK FOR NAME IN FILE SLOTS
XLKF	LOCK FILE
XLST	LIST FILE DIRECTORY
XNOP	OPEN NON-EXCLUSIVE RANDOM
XPSF	POSITION FILE
XRBF	READ BLOCK
XRDE	READ DIRECTORY ENTRY
XRDN	READ DIRECTORY NAME
XRFA	READ FILE ATTRIBUTES
XRLF	READ LINE
XRNF	RENAME FILE
XROO	OPEN READ ONLY RANDOM
XROP	OPEN RANDOM FILE
XRST	RESET FILES
XRSE	READ SECTOR
XRSZ	READ SECTOR ZERO
XRWF	REWIND FILE
XSOP	OPEN SEQUENTIAL FILE
XSZF	SIZE DISK
XULF	UNLOCK FILE
XWBF	WRITE BLOCK
XWFA	WRITE FILE ATTRIBUTES
XWLF	WRITE LINE
XWSE	WRITE SECTOR

(12.2 R$MODA - EXECUTIVE MODULE continued)

R$MODA modules DEF's and REF's are defined as follows:

Module DEF'S	
BERR02	Error handler entry
CLKI	Clock interrupt processor
CLKWS	Clock interrupt workspace
CMPN	Current page number
DAY	Day of system clock
F8B	8-bit input flag base
FCNT	Fine counter
FPAC	Floating point accumulator
FPEAD	Floating point error register
HRS	Hours of system clock
INTWS	Interrupt processing workspace
MAIL	Mail array
MAILP	MAIL[0]
MIN	Minutes of system clock
MON	Month of system clock
OUTAB	Output interrupt spool table
PINT	Console port interrupt processor
PORTS	Character input buffers
SBPTR	Current status block pointer
SEC	Seconds of system clock
SWLOCK	Swap lock
SWTP	Current task list pointer
TICS	Two word counter
TIME	Current task execution timer
TSKN	Current task number
TYPE	Character output routine address
XSUII	Suspended task interrupt processor
YRS	Year of system clock

Module REF's	
BASIC	BASIC entry address
CLSC	Clear screen characters
CR9902	9902 control register
CRUTB	Port CRU base table
CVBD	Convert binary to decimal XOP
CVBI	Convert binary to integer XOP
DTICS	Tics/sec
EFLG	BASIC error flag
EVFX	Evaluate and fix XOP
EVTMB	Event table address
FAD	Floating point add
FDD	Floating point divide
FLDD	FLoating point load
FMD	Floating point multiply
FOPS	Floating point miscellaneous
FSCL	Floating point scale
FSD	Floating point subtract
FSRD	FLoating point store
GOSBE	BASIC error trap address
IMSK	Initial interrupt mask
INTE	9901 interrupt enable mask
MCRU	Paging CRU base address
NTM	Maximum number of task messages
PSSC	Position cursor lead characters
STM	Size of message area
STPS	Number of tics/second
SWTAB	Task list address
TIMC	9901 clock constant
TMBFS	End of task messages
TMD	Destination of task message
TMS	Source of task message
TSKTB	Task generation table
X12PC	XOP 12 program counter
X12WS	XOP 12 workspace
XCLS	Clear screen primitive
XPSC	Position cursor primitive

12.3 R$MODB - BASIC MODULE

The BASIC module is a stripped down BASIC interpreter. When a task is first started, the run executive module checks the first word of the task. If it is a minus one, the BASIC interpreter is called. The following lists the supported commands. If an unsupported command is called, error 34 occurs.

Supported	Not supported
BASE	ALOAD
BAUD	BYE
CALL	CLOSE
CLEAR	DEFINE
DATA	DELETE
DATE	DISPLAY
DEFN	ESCAPE
DIM	FILE
ELSE	GOPEN
EQUATE	LOAD
ERROR	NOESC
EVENT	OPEN
EXTERNAL	PDOS
FNEND	PURGE
FNPOP	RENAME
FOR	RESET
FREE	ROPEN
GLOBAL	RUN
GOSUB	SOPEN
GOTO	SAVE
IF	SAVEB
INPUT	SPOOL
LABEL	TRACE
LET	SYS[4]
LOCAL	SYS[7]
NEXT	SYS[12]
ON	SYS[19]
POP	
PRINT	
READ	
REM	
RESTORE	
RETURN	
SKIP	
STOP	
SWAP	
THEN	
TIME	
UNIT	
WAIT	

Module DEF's:

BASIC	BASIC entry address
CVBD	Conversion XOP 9
CVBI	Conversion XOP 10
EFLG	Error trapping status block index
EVFX	Evaluate and fix XOP 8
FUNBK	Function prep
FUNRT	Function return
GOSBE	Error trapping address

12.4 R$MODC - USER CONFIGURATION MODULE

The user configuration module must be included in a standalone run module. It defines the following for the executive and BASIC modules:

1.	Initial system task list consisting of an entry for each task. An entry consists of a DSEG beginning address (R$DBxx), a DSEG end address (R$DExx), a PSEG beginning address (R$PMxx), and a page/port variable (R$PTxx).	DEF NTB,TSKTB REF R$DB00,R$DE00,R$PM00,R$PT00
2.	Map CRU base address.	DEF MCRU
3.	System clock interval, tics/second, and default tics/task.	DEF TIMC,STPS,DTICS
4.	Interrupt mask and TSM9901 interrupt enable bits.	DEF IMSK,INTE
5.	Event Timers 112 through 115 definitions.	DEF EVTMB
6.	System TMS9902 CRU base addresses and control register constant.	DEF CRUTB,CR9902
7.	Clear screen and position cursor routines with task control block parameters.	DEF XPSC,XCLS DEF CLSC,PSSC
8.	BASIC screen control table address.	DEF SCRNTB
9.	BASIC CALL table address and entries.	DEF ECTAB REF R$SB00,R$SB01,R$SB02,R$SB03
10.	Undefined BASIC routine entry points.	DEF ESCY,NOEY,PURY,RUNY DEF FILY,BYEY,CLSY,GOPY DEF OFFY,DLFY,DSPY,LDPY DEF OPNY,PDSY,RENY,RSTY DEF ROPY,SOPY,SVBY,SAVY DEF SPLY,TRCY,ALDY
11.	XOP 12 definition.	DEF X12WS,X12PC
12.	Task message definition and storage.	DEF NTM,STM,TMD,TMS DEF TMD1,TMS2,TMBFS

12.5 R$MODF - FLOATING POINT MODULE

The floating point module is a single accumulator, IBM format, floating point processor. All the arithmetic required by the BASIC module is included.

The IBM format consists of a sign bit, 7 bits of exponent, and 40 bits of fraction. Zero is represented by 48 bits begin zero. All floating numbers must be normalized for the floating point operations to work correctly.

Other modules may be substituted such as a 4 byte floating point package (R$MODF2) or a hardware floating point processor. The 1481 CPU package is ideally suited for a hardware upgrade to a standalone package.

See Chapter 8 FLOATING POINT PACKAGE.

Module DEF's:

FAD	Floating point add
FADDI	3 word add subroutine
FDD	Floating point divide
FLDD	Load FPAC
FMD	Floating point multiply
FOPS	Miscellaneous operations
FSCL	Scale FPAC
FSD	Floating point subtract
FSRD	Store FPAC
FSUBI	3 word subtract subroutine

Module REF's:

FPAC	Floating point accumulator
FPAC2	
FPAC4	
FPEAD	Floating point error address

12.6 R$MODT - TRIGONOMETRIC MODULE

Often, BASIC applications do no require trigonometric functions and hence, R$MODT has been taken out of the BASIC module and need only be included if the following BASIC functions are used:

ATN[X] = Arctangent
COS[X] = Cosine
EXP[X] = Exponential
LOG[X] = Natural logarithm
SIN[X] = Sine
TAN[X] = Tangent

A two word (4-byte floating point) version (R$MODT2) can be linked in place of R$MODT for further improvements in execution speeds and memory usage. The two word floating point module (R$MODF2) must be used at the same time.

Module DEF's

ATNF	Arctangent
COSF	Cosine
EXPF	Exponential
LOGF	Natural logarithm
SINF	Sine
TANF	Tangent

Module REF's

FPAC	Floating point accumulator
FUNBK	Function prep
FUNRT	Function return
DS0,DS1,DS2	

12.7 LINKING A BASIC PROGRAM TASK

A BASIC program is linked into a standalone run module by the *11 command of the LINKER program. The BASIC program must be in binary (BX) form. When a task is started by the executive module, the first word is checked for the module type. A minus one (-1) indicates a BASIC program.

The LINKER program generates the DEF and REF symbols required to link a BASIC program to the tasking executive module. The task number is assigned by the *14 command of the LINKER. It is your responsibility to ensure consecutive numbering of tasks.

The symbols generated include the program beginning (R$PMxx), the RAM segment beginning (R$DBxx), the RAM segment end (R$DExx), and the page/console port number (R$PTxx).

The RAM requirements must be computed before linking the BASIC module. This generally requires that the program be first "RUN" under PDOS so that all variable space is allocated. Then, the VARS parameter of the SIZE command indicates approximately how much RAM is required if no function recursions are done. More exactly, the RAM requirements are computed as follows:

```
     VARS:<#>     BASIC variable storage
+         80      20 x 4   GOSUB stack
+        144      8 x 18   FOR/NEXT stack
+         40      20 x 2   EXTERNAL table size
+        544      Status block size
+        256      Expression stack (variable)
     --------
[           /1024] rounded to the next
                   highest 1K boundary
```

If function recursion is used by the program, approximately 32 bytes is used by each level of recursion. A trial and error method may be necessary to arrive at the final size.

The resulting value is rounded to the next greater 1K boundary and entered as the <S> parameter of the link command. The <P> parameter is the console port number. (A paged memory system uses the left byte of the <P> parameter for the page #.)

```
.EX
LOAD "HPLOT"
*READY
SIZE
PRGM:424
VNAM:18
VARS:24
FREE:31196
RUN
NUMBER OF SLICES=20

 .
 . .
 . . .
 . . . .
 . . . . .

ESCAPE AT 1020

SIZE
PRGM:424
VNAM:18
VARS:102
FREE:31082


;102 + 80 + 144 + 544 + 256; 1126
```

Hence a 2K task size is required.

12.8 LINKING AN ASSEMBLY LANGUAGE TASK

Four parameters are required to link an assembly task to the run module. These are the program module beginning address (R$PMxx), the data segment beginning address (R$DBxx), the data segment end address (R$DExx), and finally, the console port number (R$PTxx). (xx) is the assigned task number and ranges from 00 to 99. These entries must be defined in module R$MODC.

R$PMxx = Program Module beginning

R$PTxx = Task port #

R$DBxx = Data segment Beginning

R$DExx = Data segment End

Assembly tasks are linked after BASIC task modules. The task number is assigned by the DEF symbols.

The following is an example of how an assembly task can be linked in the run module as task 2 on port 2. Remember, two other tasks must be linked for tasks 0 and 1.

```
*TASK 2 LINKAGE
*
        DEF R$PM02,R$PT02       ;REQ LINK PARAMETERS
        DEF R$DB02,R$DE02
*
R$PT02  EQU 2                   ;PORT #2
*
        RORG 0
R$PM02  EQU $                   ;BEGINNING OF PROGRAM MODULE

        <Assembly program>

        DORG                    ;DATA SEGMENT
        RORG 0
R$DB02  EQU $                   ;BEGINNING OF DSEG
        BSS 32                  ;TASK WORKSPACE
        BSS 512                 ;STATUS BLOCK

        <RAM constants here>

R$DE02  EQU $                   ;END OF DSEG
        DATA 0                  ;FORCE ADDRESS TAG
        END R$PM02
```

Even if no RAM variables are used in the assembly task, a segment of memory must be reserved at the beginning of the data segment (DSEG section) for the primary task workspace and status block. This area is 544 bytes in length. Program variables follow after these definitions. The DSEG segment cannot end with a BSS or BES directive as no assembler address tags would appear in the object. Hence, a DATA directive should be the last statement of the data segment or DATA 0 should follow the R$DExx definition.

544 bytes for system area

12.9 LINKING AN ASSEMBLY SUBROUTINE

Assembly subroutines are linked to other assembly programs through conventional REF's and DEF's. BASIC calls to assembly subroutines must be done through the BASIC CALL table. Reference is made by an index into the table. (See 10.10 CALL)

BASIC CALL table entries are DEF'd out from the user configuration module (R$MODC). Hence, the subroutine entry point name need only match the DEF symbol names of the R$MODC module. Four of these have been predefined as follows:

```
R$SB00 = CALL #0
R$SB01 = CALL #1
R$SB02 = CALL #2
R$SB03 = CALL #3
```

Others may be added as needed, up to 256 entries. (BASIC links the CALL by index to 0 through 255.) The following example shows how a subroutine could be linked to a run module and accessed by BASIC as CALL #1:

```
* SUBROUTINE #1
*
        DEF R$SB01      ;SUBROUTINE LINK
*
        RORG 0
R$SB01  EQU $           ;PROGRAM BEGINNING

        <assembly program>

        RT
*
        DSEG            ;DATA SEGMENT
        RORG 0

        <assembly variables>

        END R$SUB01
```

Notice that no workspace or status block is required. However, all RAM variables must be placed in a DSEG segment.

Conventional linkage

CALL #0-255

12.10 ERROR HANDLING

Error trapping necessary!

Since a run module does not require a terminal, most errors are fatal unless handled by the application program themselves. The ERROR statement can be used in BASIC programs to trap errors to an error handler routine. Assembly programs should process errors returned from PDOS primitives in a similar fashion.

Fatal errors to console

Pulsing LED repeats error number

Fatal errors are reported to the console terminal. They are also indicated a pulsing LED on the CPU. The pulses repeatedly count out the error number, a digit at a time, with a long pulse inbetween. Fatal errors include the use of non-supported run module commands and error calls to PDOS. When such an error occurs, interrupts are disabled and the error indicator repeats forever. No further tasking is done.

As an example, error 27 would look like:

very long OFF
ON, OFF, ON
long OFF
ON, OFF, ON, OFF, ON, OFF, ON, OFF, ON, OFF, ON, OFF, ON
repeat

(A zero digit is indicated by a long ON.)

12.11 RUN MODULE EXAMPLE

```
.EX                                                    Load and save in token format PRGM0:SR
*READY
NEW
*READY
LOAD "PRGM0:SR"
*READY
LIST
 10  BAUD 1,0
 20  PRINT "CALL SUBROUTINE #0.  HERE WE GO....."
 30  CALL #0
 40  PRINT "WE ARE BACK!!!!  COM[0]=";COM[0]
 50  PRINT "CALL SUBROUTINE #1.";
 60  CALL #1
 70  INPUT "ENTER NUMBER:";I: IF I<1: PRINT "TRY AGAIN": GOTO 70
 80  MAIL[0]=I: EVENT 30
 90  IF EVF[30]: SWAP : GOTO 90
 100  PRINT I;" FACTORIAL =";MAIL[1]
 110  GOTO 20
SAVEB "PRGM0"
*READY
NEW
*READY
LOAD "PRGM1:SR"                                        Load and save in token format PRGM1:SR
*READY
LIST
 10  REM PRGM1:SR
 20  WAIT 30  !SUSPEND TASK UNTIL EVENT
 30  IF MAIL[0]>30: GOTO 50
 40  MAIL[1]=FNFACT[MAIL[0]]
 50  EVENT -30  !ACKNOWLEDGE
 60  GOTO 20

 100  DEFN FNFACT[I]
 110  IF I<=1: FNFACT=1: FNEND
 120  FNFACT=I*FNFACT[I-1]
 130  FNEND
SAVEB "PRGM1"
*READY
BYE
.ASM SUBR#0:SR,SUBR#0,TEMP                             Assemble SUBR#0:SR
ASM R2.4
SRCE=SUBR#0:SR
OBJ=SUBR#0
LIST=TEMP
ERR=
XREF=

END OF PASS 1
0 DIAGNOSTICS
END OF PASS 2
0 DIAGNOSTICS
.SF TEMP
```

(12.11 RUN MODULE EXAMPLE continued)

```
                                                    PDOS ASM R2.4
PAGE: 1                 13:06  09/27/82             FILE: SUBR#0:SR,DISK #5

1                       *       SUBR#0:SR       09/27/82
2                       ************************************************
3                       *       BASIC CALL SUBROUTINE #0
4                       *
5                               IDT 'SUBR#0'
7                       *
8                               OPT ?=1         ;RUN MODULE ENABLE
9                       ?       DEF R$SB00      ;CALL #0
10                      *
11                      *       AORG >2240
12                      *       DATA R$SB00
13                      *
14      0000'0000'              RORG 0
15 0000: 2F5B           R$SB00  XPMC            ;PRINT MESSAGE
16 0002: 000A'                    DATA MES1
17 0004: 05A7 0002              INC @2(7)       ;COUNT COM[0]          Increment COM[0] on each call
18 0008: 045B                   RT
19                      *
20 000A: 2A2A 2A2A 2053 MES1    TEXT '**** SUBROUTINE CALL #0 WORKS! ****'
   0010: 5542 524F 5554
   0016: 494E 4520 4341
   001C: 4C4C 2023 3020
   0022: 574F 524B 5321
   0028: 202A 2A2A 2A00
21 002D: 0A0D                   BYTE >0A,>0D
22 002F: 0000                   BYTE 0
23      0030'                   EVEN
24 0030:     0000'              END R$SB00


DEMONSTRATION CALL SUBROUTNE #0                     PDOS ASM R2.4
PAGE: 2                 13:06  09/27/82             FILE: SUBR#0:SR,DISK #5

18 SYMBOLS
MES1    R 000A   R0     A 0000   R1     A 0001   R10    A 000A
R11     A 000B   R12    A 000C   R13    A 000D   R14    A 000E
R15     A 000F   R2     A 0002   R3     A 0003   R4     A 0004
R5      A 0005   R6     A 0006   R7     A 0007   R8     A 0008
R9      A 0009   R$SB00 R 0000
```

(12.11 RUN MODULE EXAMPLE continued)

```
.ASM SUBR#1:SR,SUBR#1,TEMP
ASM R2.4
SRCE=SUBR#1:SR
OBJ=SUBR#1
LIST=TEMP
ERR=
XREF=

END OF PASS 1
0 DIAGNOSTICS
END OF PASS 2
0 DIAGNOSTICS
.SF TEMP
```

(12.11 RUN MODULE EXAMPLE continued)

```
                                                    PDOS ASM R2.4
PAGE: 1                 13:06  09/27/82             FILE: SUBR#1:SR,DISK #5

1                       *       SUBR#1:SR       09/27/82
2                       ********************************************************
3                       *       LIST TASK SUBROUTINE
4                       *
5                       ********************************************************
6                       *
7                               IDT 'SUBR#1'
9                       *
10                              OPT ?=1         ;RUN MODULE ENABLE
11                      *
12                      ?       DEF R$SB01      ;CALL ADDRESS
13                      ?       REF SWTAB,TSKN
14                      *
15      0188            PORT    EQU >0188       ;PORT 1
16      01DC            EUM     EQU >01DC       ;END-USER-MEMORY INDEX
17      01E6            U1CRU   EQU >01E6       ;PORT 1 CRU BASE
18                      *
19                      * PDOS 2.4 ADDRESSES
20                      *
21                      *SWTAB  EQU >2C10       ;TASK LIST ADDRESS
22                      *TSKN   EQU >2FE5       ;CURRENT TASK #
23                      *
24                      *       AORG >2242
25                      *       DATA R$SB01
26                      *
27      0000'0000'              RORG 0
28      0000'           R$SB01  EQU $           ;PROGRAM ENTRY
29 0000: C80B 0000"     LTSK    MOV R11,@SAV1   ;SAVE RETURN ADDRESS
30 0004: 2F5B                   XPMC            ;OUTPUT HEADER
31 0006: 009A'                    DATA LTM1
32 0008: 0203 8542+             LI R3,SWTAB     ;GET SWAP TABLE ADDRESS
33                      *
34 000C: C083           LTSK02  MOV R3,R2
35 000E: 0223 0014              AI R3,20        ;SETUP FOR NEXT TASK
36 0012: C1B2                   MOV *R2+,R6     ;GET PAGE/TICS, DONE?
37 0014: 1333                     JEQ LTSK08    ;Y
38 0016: 2F59                   XPCL            ;N, OUTPUT CRLF
39 0018: 0200 2000              LI R0,' '*256   ;GET BLANK
40 001C: 9CA0 854A+             CB @TSKN,*R2+   ;SPAWNED TASK?
41 0020: 1602                     JNE LTSK04    ;N
42 0022: 0200 2B00              LI R0,'+'*256   ;Y
43                      *
44 0026: 94A0 854A+     LTSK04  CB @TSKN,*R2    ;CURRENT TASK?
45 002A: 1602                     JNE LTSK06    ;N
46 002C: 0200 2A00              LI R0,'*'*256   ;Y, OUTPUT "*
```

(12.11 RUN MODULE EXAMPLE continued)

```
LIST TASK CALL SUBROUTINE                    PDOS ASM R2.4
PAGE: 2              13:06  09/27/82         FILE: SUBR#1:SR,DISK #5

1  0030: 2F58          LTSK06  XPCC            ;OUTPUT TASK ID
2  0032: D072                  MOVB *R2+,R1    ;OUTPUT TASK #
3  0034: 0981                  SRL R1,8
4  0036: 2FD6                  XCBD
5  0038: 2F5A                  XPLC
6  003A: 2F4F                  XTAB
7  003C: 0007                    DATA 7
8  003E: C046                  MOV R6,R1       ;GET PAGE
9  0040: 0981                  SRL R1,8
10 0042: 2FD6                  XCBD
11 0044: 2F5A                  XPLC
12 0046: 2F4F                  XTAB
13 0048: 000D                    DATA 13
14 004A: C046                  MOV R6,R1       ;GET COUNT
15 004C: 06C1                  SWPB R1
16 004E: 0881                  SRA R1,8
17 0050: 2FD6                  XCBD
18 0052: 2F5A                  XPLC
19 0054: 2F4F                  XTAB
20 0056: 0011                    DATA 17
21 0058: C1D2                  MOV *R2,R7      ;SAVE STATUS BLOCK POINTER
22 005A: 06A0 0084'            BL @LTSKN       ;LIST SB,WS,PC,SR
23 005E: 0202 0002"            LI R2,SAV2
24 0062: C102                  MOV R2,R4
25 0064: C507                  MOV R7,*R4      ;STATUS BLOCK PTR
26 0066: 6D20 0098'            S @C32,*R4+     ;GET BEGINNING OF USER MEMORY
27 006A: CD27 01DC             MOV @EUM(7),*R4+
28 006E: CD27 01E6             MOV @U1CRU(7),*R4+
29 0072: C527 0188             MOV @PORT(7),*R4
30 0076: 06A0 0084'            BL @LTSKN       ;OUTPUT BUS,EUS,CRU,PORT
31 007A: 10C8                  JMP LTSK02
32                      *
33 007C: 2F59          LTSK08  XPCL            ;OUTPUT CRLF
34 007E: C2E0 0000"            MOV @SAV1,R11   ;RESTORE R11
35 0082: 045B                  RT
```

(12.11 RUN MODULE EXAMPLE continued)

```
LIST TASK CALL SUBROUTINE                          PDOS ASM R2.4
PAGE: 3                 13:06  09/27/82            FILE: SUBR#1:SR,DISK #5

1                       ************************************************
2                       *       LIST 4 HEX NUMBERS
3                       *
4  0084: 0704           LTSKN   SETO R4
5                       *
6  0086: 0200 203E      LTSKN2  LI RO,>203E
7  008A: 2F58                   XPCC            ;OUTPUT ' >'
8  008C: C072                   MOV *R2+,R1     ;GET DATA
9  008E: 2FD7                   XCBH            ;CONVERT TO HEX
10 0090: 2F5A                   XPLC            ;OUTPUT
11 0092: 0944                   SRL R4,4        ;DONE?
12 0094: 16F8                     JNE LTSKN2    ;N
13 0096: 0458                   RT              ;Y
14                      *
15 0098: 0020           C32     DATA 32
16                      *
17 009A: 0A0D           LTM1    BYTE >0A,>0D
18 009C: 5365 4220 2331         TEXT 'SUB #1 - CURRENT TASK LISTING'
   00A2: 202D 2043 5552
   00A8: 5245 4E54 2054
   00AE: 4153 4B20 4C49
   00B4: 5354 494E 4700
19 00B9: 0A0D                   BYTE >0A,>0D
20 00BB: 5441 534B 2020         TEXT 'TASK  PAGE  TIME    '
   00C1: 5041 4745 2020
   00C7: 5449 4D45 2020
   00CD: 2020
21 00CF: 5342 2020 2020         TEXT 'SB    WS    PC    SR    '
   00D5: 5753 2020 2020
   00DB: 5043 2020 2020
   00E1: 5352 2020 2020
22 00E7: 424D 2020 2020         TEXT 'BM    EM   CRU   PORT'
   00ED: 454D 2020 2043
   00F3: 5255 2020 2050
   00F9: 4F52 5400
23 00FC: 0A0D 0000              BYTE >0A,>0D,0
24       0100'                  EVEN
25                      *
26                      ************************************************
27                      *       SUBROUTINE DATA SEGMENT
28                      *
29       0100'0000"     ?       DSEG            ;DSEG PC FOR RUN MODULE
30       0000"0000"     ?       RORG 0
31                      *
32 0000: 0000           SAV1    DATA 0          ;R11 SAVE
33 0002: 0000 0000      SAV2    DATA 0,0        ;4 WORD SAVE AREA
34 0006: 0000 0000              DATA 0,0
35 000A:       0000'            END R$SB01
```

(12.11 RUN MODULE EXAMPLE continued)

```
LIST TASK CALL SUBROUTINE                    PDOS ASM R2.4
PAGE: 4                 13:06  09/27/82      FILE: SUBR#1:SR,DISK #5

33 SYMBOLS
C32     R 0098   EUM     A 01DC   LTM1    R 009A   LTSK    R 0000
LTSKN   R 0084   LTSKN2  R 0086   LTSK02  R 000C   LTSK04  R 0026
LTSK06  R 0030   LTSK08  R 007C   PORT    A 0188   R0      A 0000
R1      A 0001   R10     A 000A   R11     A 000B   R12     A 000C
R13     A 000D   R14     A 000E   R15     A 000F   R2      A 0002
R3      A 0003   R4      A 0004   R5      A 0005   R6      A 0006
R7      A 0007   R8      A 0008   R9      A 0009   R$SB01  R 0000
SAV1    D 0000   SAV2    D 0002   SWTAB   E 0000   TSKN    E 0000
U1CRU   A 01E6


.ASM TASK02:SR,TASK02,TEMP
ASM R2.4
SRCE=TASK02:SR
OBJ=TASK02
LIST=TEMP
ERR=
XREF=

END OF PASS 1
0 DIAGNOSTICS
END OF PASS 2
0 DIAGNOSTICS
.SF TEMP
```

(12.11 RUN MODULE EXAMPLE continued)

```
                                                    PDOS ASM R2.4
PAGE: 1                 13:06  09/27/82             FILE: TASK02:SR,DISK #5

1                       *       TASK02:SR       09/27/82
2                       ************************************************
3                       *       CURSOR ADDRESSING DEMO (RUN AS TASK #2)
4                       *
5                               IDT 'TASK #2'
7                       *
8                               OPT ?=1         ;RUN MODULE ENABLE
9                       *
10                      ?       DEF R$PM02,R$PT02       ;REQ LINK PARAMETERS
11                      ?       DEF R$DB02,R$DE02
12                      ?       REF INTHS,XSUII         ;SUSPEND TASK
13                      *
14      0000'0014       ?       AORG >0014              ;LEVEL 5 INTERRUPT
15 0014: 8462+853A+     ?       DATA INTHS,XSUII        ;WAKE TASK ROUTINE
16                      *
17      0002            R$PT02  EQU 2           ;PORT 2 (AUX PORT)
18                      *
19      0018 0000'              PSEG
20      0000'0000'              RORG 0
21      0000'           R$PM02  EQU $           ;PROGRAM ENTRY ADDRESS
22 0000: 04C1           SUB     CLR R1          ;NO CRU BASE CHANGE
23 0002: 0205 0002              LI R5,R$PT02    ;GET PORT #
24 0006: 04C6                   CLR R6          ;19.2K BAUD
25 0008: 2F49                   XBCP            ;BAUD CONSOLE PORT
26                      *
27 000A: 0201 0005      SUB02   LI R1,5         ;SUSPEND TASK
28 000E: 2FC7                   XSUI            ;  ON LEVEL 5
29 0010: 020C 0180              LI R12,>0180    ;POINT TO AUX PORT
30 0014: 1D12                   SBO 18          ;ACKNOWLEDGE INTERRUPT
31 0016: 0204 00C8              LI R4,200       ;DO 200 TIMES
32                      *
33 001A: 06A0 003A'     SUB04   BL @RANDG       ;GET RANDOM #
34 001E: 3820 0054'             MPY @C72,R0     ;GET COLUMN
35 0022: C080                   MOV R0,R2
36 0024: 06A0 003A'             BL @RANDG       ;GET 2ND RANDOM #
37 0028: 3820 0052'             MPY @C24,R0     ;GET ROW
38 002C: C040                   MOV R0,R1
39 002E: 2F5D                   XPSC            ;POSITION CURSOR
40 0030: 2F5B                   XPMC            ;OUTPUT MESSAGE
41 0032: 0056'                    DATA MES01
42 0034: 0604                   DEC R4          ;DONE?
43 0036: 15F1                     JGT SUB04     ;N
44 0038: 10E8                   JMP SUB02       ;Y, SUSPEND TASK AGAIN
```

(12.11 RUN MODULE EXAMPLE continued)

```
DEMONSTRATION TASK #2                          PDOS ASM R2.4
PAGE: 2                13:06  09/27/82         FILE: TASK02:SR,DISK #5

 1                     ************************************************
 2                     *       GENERATE PSEUDO RANDOM SEQUENCE
 3                     *
 4                     *       LINEAR CONGRUENTIAL SEQUENCE:
 5                     *       X[N+1] = (X[N] * A + C)  MOD 2^16
 6                     *
 7                     *USE MOST SIGNIFICANT K-BITS IF <16 ARE REQUIRED
 8                     *
 9                     *
10 003A: C020 0220"    RANDG   MOV @RANDS,R0   ;GET RANDOM SEED
11 003E: C0C0                  MOV R0,R3
12 0040: 0AB3                  SLA R3,11       ;N*2^11
13 0042: A0C0                  A R0,R3         ;N*2^11+N
14 0044: 0A23                  SLA R3,2
15 0046: A003                  A R3,R0         ;N*2^11+5N
16 0048: 0220 3619             AI R0,13849     ;(N*A+13849.)  MOD 2^16
17 004C: C800 0220"            MOV R0,@RANDS   ;STORE NEW SEED
18 0050: 045B                  RT
19                     *
20                     ************************************************
21                     *       PROGRAM CONSTANTS
22                     *
23 0052: 0018          C24     DATA 24
24 0054: 0048          C72     DATA 72
25                     *
26 0056: 5044 4F53     MES01   TEXT 'PDOS'
27 005A: 0000                  BYTE 0
28       005C'                 EVEN
29                     *
30                     ************************************************
31                     *       TASK DATA SEGMENT
32                     *
33       005C'0000"    ?       DSEG            ;DSEG PC FOR RUN MODULE
34       0000"0000"    ?       RORG 0
35                     *
36       0000"         R$DB02  EQU $           ;BEGINNING OF DSEG
37       0000"0020"    ?       BSS 32          ;TASK WORKSPACE
38       0020"0220"    ?       BSS 512         ;STATUS BLOCK
39 0220: 0000          RANDS   DATA 0          ;RANDOM SEED
40       0222"         R$DE02  EQU $           ;END OF DSEG
41 0222:      0000'            END SUB
```

(12.11 RUN MODULE EXAMPLE continued)

```
.LINK                                                    Link modules together
LINKER R2.4
*<CR>
COMMAND         DESCRIPTION

0,<FILE>        OPEN OUTPUT FILE
1,<FILE>        LINK FILE
2{,<FILE>}      LIST UNDEFINED REFS
3{,<FILE>}      LIST MULTIPLY DEFINED DEFS
4{,<FILE>}      LIST LINK MAP
5               OUTPUT PARTIAL LINK
6{,<ADR>}       OUTPUT OVERLAYS AND START TAG
7               EXIT TO PDOS
8{,<ADR>}       LIST/SET PSEG BASE ADDRESS
9{,<ADR>}       LIST/SET DSEG BASE ADDRESS
10              RESTART
11,<FL>,<S>,<P> LOAD BASIC BINARY MODULE
12,<M>          0=NO DSEG, 1=NORMAL, 2=DSEG=>PSEG
13,<FILE>       LIBRARY
14{,DEFAULT}    SET DEFAULT TASK NUMBER
15{,FLAG}       OBJECT MODE, 0=OFF, 1=ON
*9,>4000                                                 Set RAM base at >4000
WAS >0000
*12,0                                                    Do not output any DSEG tags
WAS >0000
*0,TEMP                                                  Output to TEMP
*1,R$MODA                                                Load PDOS executive module
*1,R$MODB                                                Load BASIC module
*1,R$MODC                                                Load configuration module
*1,R$MODF                                                Load floating point routines
*11,PRGM0,2,1                                            Load BASIC program PRGM0 with 2K bytes
                                                           on port 1
MODULE SIZE: 268
*11,PRGM1,2,0                                            Load BASIC program PRGM1 with 1K bytes
MODULE SIZE: 182                                           with no port
*1,TASK02                                                Load task #2
*1,SUBR#0                                                Load BASIC assembly program #0
*1,SUBR#1                                                Load BASIC assembly program #1
*2                                                       Output all undefined DEF's
UNDEFINED DEF ENTRIES:
ATNF   >0000  COSF   >0000  EXPF   >0000  LOGF   >0000  R$DB03 >0000
R$DE03 >0000  R$PM03 >0000  R$PT03 >0000  R$SB02 >0000  R$SB03 >0000
SINF   >0000  TANF   >0000
*3                                                       Output any multiply defined DEF's
MULTIPLY DEFINED DEF ENTRIES: NONE
*4,LINKMAP                                               Output link map to LINKMAP
*6                                                       Set start tag at >0000
START TAG = >0000
*7                                                       Exit to PDOS
```

(12.11 RUN MODULE EXAMPLE continued)

```
.SF LINKMAP                                                    Look at link map

LINK MAP FILE
TIME=13:08:00
DATE=09/27/82
FILE=TEMP
FILE  NAME:EXT          IDT         ENTRY  [--PSEG---]  [--DSEG---]  DEF  REF

   1  R$MODA            'R$MA2.4 '  >0000  >0000 >09F4  >4000 >41F6   31   46
   2  R$MODB            'R$MB2.4 '  >09F4  >09F4 >22E8  >41F6 >41F6   11   64
   3  R$MODC            'R$MC2.4 '  >22E8  >22E8 >23B2  >41F6 >43E8   49   21
   4  R$MODF            'R$MF2.4 '  >23B2  >23B2 >26F8  >43E8 >43E8    8    2
   5  PRGM0             'R$PMEX00'  >26F8  >26F8 >2804  >43E8 >4BE8    4    0
   6  PRGM1             'R$PMEX01'  >2804  >2804 >28BA  >4BE8 >53E8    4    0
   7  TASK02            'TASK #2 '  >28BA  >28BA >2916  >53E8 >560A    4    2
   8  SUBR#0            'SUBR#0  '  >2916  >2916 >2948  >560A >560A    1    0
   9  SUBR#1            'SUBR#1  '  >2948  >2948 >2A48  >560A >5614    1    3

DEF      F/T   VALUE    REFERENCES                                    EPROM required from >0000 to >2A48
                                                                      RAM required from >4000 to >5614
ALDY     3 P   >23A4    #2 P >0ADA
ATNF       U   >0000    #2 P >1A60
BASIC    2 P   >09F4    #1 P >0174                                    BASIC entry address
BERR02   1 P   >01FA    #2 P >1512                                    Error processor
BYEY     3 P   >23A4    #2 P >0ADC
CLKI     1 P   >0314                                                  Clock interrupt processor
CLKWS    1 D   >41D6                                                  Clock workspace
CLSC     3 P   >232C    #1 P >013C                                    Clear screen characters
CLSY     3 P   >23A4    #2 P >0ADE
CMPN     1 D   >41DE    #2 P >1AAE                                    Current memory page #
COSF       U   >0000    #2 P >1A64
CR9902   3 P   >232A    #1 P >00B0   #1 P >0950                      9902 control register constant
CRUTB    3 P   >2318    #1 P >00A4                                    9902 CRU base address table
CVBD     2 P   >1E14    #1 P >0066
CVBI     2 P   >1E0A    #1 P >006A
DAY      1 D   >4048
DFFY     3 P   >23A4    #2 P >0AE2
DLFY     3 P   >23A4    #2 P >0AE4
DS0      2 A   >0142
DS1      2 A   >0148
DS2      2 A   >014E
DSPY     3 P   >23A4    #2 P >0AE6
DTICS    3 A   >0003    #1 P >00BC                                    Default tics/task
ECTAB    3 P   >239C    #2 P >0CD4
EFLG     2 A   >0132    #1 P >0200
ESCY     3 P   >23A4    #2 P >0AA6
EVFX     2 P   >15A2    #1 P >0062
EVTMB    3 P   >2310    #1 P >0278                                    Event timers 112 through 115
EXPF       U   >0000    #2 P >198E   #2 P >1A6C
F8B      1 D   >4176
FAD      4 P   >2308    #1 P >004A
FCNT     1 D   >41DC                                                  Fine counter
```

(12.11 RUN MODULE EXAMPLE continued)

```
FDD      4 P   >264C    #1 P >0056
FILY     3 P   >23A4    #2 P >0AE8
FLDD     4 P   >23BE    #1 P >0042
FMD      4 P   >254A    #1 P >0052
FOPS     4 P   >24A0    #1 P >005E
FPAC     1 D   >419E    #2 P >1924   #2 P >19A6   #2 P >1A94   #2 P >1AB2oating point accumulator
                        #2 P >1CBA   #2 P >1D56   #2 P >1D82   #2 P >1D98
                        #2 P >1DE0   #2 P >1DF8   #2 P >22B0
FPAC2    1 D   >41A0    #2 P >1920   #2 P >19A2   #2 P >1A0E   #2 P >1A2C
                        #2 P >1C8C   #2 P >1CCA   #2 P >1CEA   #2 P >1CFA
                        #2 P >1D24   #2 P >1D2E   #2 P >1D52   #2 P >21A0
FPAC4    1 D   >41A2    #2 P >1DF4
FPEAD    1 D   >41A4                                          Floating point error register
FSCL     4 P   >26CA    #1 P >005A
FSD      4 P   >23CA    #1 P >004E
FSRD     4 P   >23B2    #1 P >0046
FUNBK    2 P   >1DDE
FUNRT    2 P   >1CB8
GOPY     3 P   >23A4    #2 P >0AE0
GOSBE    2 P   >1106    #1 P >0206
HRS      1 D   >4040
IMSK     3 A   >0005    #1 P >00CE                            Interrupt mask
INTE     3 P   >230A    #1 P >015A
INTWS    1 D   >417E    #7 A >0014                            Interrupt Workspace
LDPY     3 P   >23A4    #2 P >0AEA
LOGF       U   >0000    #2 P >197E   #2 P >1A78
MAIL     1 D   >4000    #2 P >0BE6
MAILP    1 D   >4004
MCRU     3 A   >0980    #1 P >00EC   #1 P >03A4   #2 P >1AA0   #2 P >1AC0
MIN      1 D   >4042
MON      1 D   >4046
NOEY     3 P   >23A4    #2 P >0ABC
NTM      3 A   >0008    #1 P >02B0   #1 P >051C
OPNY     3 P   >23A4    #2 P >0AEC
OUTAB    1 D   >4166
PDSY     3 P   >23A4    #2 P >0AEE
PINT     1 P   >0552
PORTS    1 D   >404C
PSSC     3 P   >234E    #1 P >0142                            Position cursor characters
PURY     3 P   >23A4    #2 P >0AC4
R$DB00   5 D   >43E8    #3 P >22E8                            Tasks DSEG beginning addresses
R$DB01   6 D   >4BE8    #3 P >22F0
R$DB02   7 D   >53E8    #3 P >22F8
R$DB03     U   >0000    #3 P >2300
R$DE00   5 D   >4BE8    #3 P >22EA                            Tasks DSEG ending addresses
R$DE01   6 D   >53E8    #3 P >22F2
R$DE02   7 D   >560A    #3 P >22FA
R$DE03     U   >0000    #3 P >2302
R$PM00   5 P   >26F8    #3 P >22EC                            Tasks PSEG beginning addresses
R$PM01   6 P   >2804    #3 P >22F4
R$PM02   7 P   >28BA    #3 P >22FC
R$PM03     U   >0000    #3 P >2304
```

(12.11 RUN MODULE EXAMPLE continued)

```
R$PT00   5 A   >0001   #3 P >22EE                                  Tasks page/port definitions
R$PT01   6 A   >0000   #3 P >22F6
R$PT02   7 A   >0002   #3 P >22FE
R$PT03     U   >0000   #3 P >2306
R$SB00   8 P   >2916   #3 P >239C                                  BASIC CALL subroutine addresses
R$SB01   9 P   >2948   #3 P >239E
R$SB02     U   >0000   #3 P >23A0
R$SB03     U   >0000   #3 P >23A2
RENY     3 P   >23A4   #2 P >0AF0
ROPY     3 P   >23A4   #2 P >0AF4
RSTY     3 P   >23A4   #2 P >0AF2
RUNY     3 P   >23A4   #2 P >0ACC
SAVY     3 P   >23A4   #2 P >0AFA
SBPTR    1 D   >41E8   #2 P >1E24
SCRNTB   3 P   >237E   #2 P >11E0
SEC      1 D   >4044
SINF       U   >0000   #2 P >1A86
SOPY     3 P   >23A4   #2 P >0AF6
SPLY     3 P   >23A4   #2 P >0AFC
STM      3 A   >0032   #1 P >0538   #1 P >0546
STPS     3 P   >230C   #1 P >0338
SVBY     3 P   >23A4   #2 P >0AF8
SWLOCK   1 D   >41E4
SWTAB    3 D   >41F6   #1 P >00C2   #1 P >0294   #1 P >02E0   #1 P >03[illegible]sk list address
                       #1 P >04EC   #9 P >2952
SWTP     1 D   >41EA
TANF       U   >0000   #2 P >1A8E
TICS     1 D   >4188   #4 P >250C                                  2 word timer
TICS2    1 D   >418A   #4 P >2510
TIMC     3 P   >230E   #1 P >0156
TIME     1 D   >41E6
TMBFS    3 D   >4216   #1 P >052C
TMD      3 D   >43D8   #1 P >008E
TMD1     3 D   >43D7   #1 P >0528
TMS      3 D   >43E0   #1 P >0096
TMS2     3 D   >43DE   #1 P >0530
TRCY     3 P   >23A4   #2 P >0AFE
TSKN     1 D   >41DF   #2 P >1AEE   #9 P >2966   #9 P >2970      Current task number
TSKTB    3 P   >22E8   #1 P >00C6
TYPE     1 P   >068E   #3 P >237C
X12PC    3 P   >2380   #1 P >0072                                  XOP 12 definition
X12WS    3 P   >2390   #1 P >0070
XCLS     3 P   >232E   #1 P >0994
XPSC     3 P   >2350   #1 P >0996
XSUII    1 P   >028E   #7 A >0016                                  Interrupt processor
YRS      1 D   >404A
```

(12.11 RUN MODULE EXAMPLE continued)

```
.BURN                                                    The object can be burned directly in EPROM
PROLOG TI9900 OBJECT BURN R2.4
PORT=2
FILE NAME=MODULE
BEGINNING ADDRESS TO LOAD=
PLEASE WAIT.....
0000 048F 0490 0869 086A 086B 086C 086D 086E 0872 0EFF
0F02 0F03 0F06 23B9 23BA 23BB 23BC 23BD 23BE 23BF 23C0
23C1 23C2 23C3 23C4 23C5 23C6 23C7 23C8 23CA 2418 2776
287A 0014 291E 297A 29A7 29AC 2A67 2AAA 01D6 01A2 0968
0124 0856 0118 0066 006A 0D32 04B4 0062 004A 18DA 1902
0056 0042 0052 005E 142A 1740 17E6 18B4 1ECA 1F5C 1FC0
1FD8 1FE8 1FFE 2036 2098 20F8 217E 2194 21E0 21F6 224A
224E 2384 239A 173C 17E2 182E 184C 1C70 1E9C 1EC6 1EDA
1EFA 1F0A 1F34 1F3E 1F58 1FD4 2518 2672 005A 004E 0046
18F0 18FA 04BA 0142 01BE 0014 09D4 096A 096C 096E 0970
1F8A 01A8 0080 0088 0090 0082 008A 0092 0084 008C 0094
0086 008E 0096 2404 2406 12B4 1400 1412 0FBC 0966 0972
29B6 2564 2568 01BA 0272 29CA 0072 0070 0016
FILE NAME=
ENTER STEP SIZE FF
BEGINNING ADDRESS=0000
     LAST ADDRESS=0017
BEGINNING ADDRESS TO BURN=0000
     LAST ADDRESS TO BURN=00FF
0 = BOTH BYTES
1 = LEFT BYTES
2 = RIGHT BYTES
3 = RIGHT NIBBLE
OPTION=1
000
07F
p
420142024202420242044202420442044202420242024202420242
424202420242244224422442244225422642274224421342184218
420423244203420442034 24A27004A52280052542900000000000
000000000000000000000000000000000000000000000000000100
030204FF06403C403C401840FF400C40631F1F1F
LAST BEGINNING ADDRESS=0000
     LAST LAST ADDRESS=00FF
          LAST OPTION=0001
BEGINNING ADDRESS=0000
     LAST ADDRESS=0017
BEGINNING ADDRESS TO BURN=
```

(12.11 RUN MODULE EXAMPLE continued)

```
.LOGO                                                  The application is tested directly
LOGO R2.4                                              on a PDOS system by using the LOGO
*<CR>                                                  utiltily to overlay PDOS.
 0(,adr)          BLWP @adr
 1,<file name>    LOAD FILE
 2,base           SET BUFFER BASE
 3,low,high,adr   SET CHECKSUM
 4,low,high,FILE  OUTPUT OBJECT
 5,low,high       WRITE BINARY TO DISK
 6                EXIT
BUFFER LIMITS ARE:
      LOW=>0000
     HIGH=>77EC
HIGHEST PC=>0000
*1,TEMP
LOADING.....
IDT='R$MA2.4 '
IDT='R$MB2.4 '
IDT='R$MC2.4 '
IDT='R$MF2.4 '
IDT='R$PMEX00'
IDT='R$PMEX01'
IDT='TASK #2 '
IDT='SUBR#0  '
IDT='SUBR#1  '
ENTRY ADDRESS=>0000
*0
GO!!!!!!!!!!!!!!!!!!!
CALL SUBROUTINE #0.  HERE WE GO.....                   Console port
**** SUBROUTINE CALL #0 WORKS! ****
WE ARE BACK!!!!  COM[0]= 1
CALL SUBROUTINE #1.
SUB #1 - CURRENT TASK LISTING
TASK  PAGE  TIME    SB    WS    PC    SR    BM    EM   CRU   PORT

*0     0     3    >42A2 >441C >0654 >D40F >4282 >4A82 >0080 >0001
+1     0     -30  >4AA2 >4A82 >1040 >D00F >4A82 >5282 >0000 >0000
+2     0     -5   >52A2 >5282 >292E >C40F >5282 >54A4 >0180 >0002
ENTER NUMBER:10
 10 FACTORIAL = 3628800
CALL SUBROUTINE #0.  HERE WE GO.....
**** SUBROUTINE CALL #0 WORKS! ****
WE ARE BACK!!!!  COM[0]= 2
CALL SUBROUTINE #1.
SUB #1 - CURRENT TASK LISTING
TASK  PAGE  TIME    SB    WS    PC    SR    BM    EM   CRU   PORT

*0     0     3    >42A2 >441C >0654 >D40F >4282 >4A82 >0080 >0001
+1     0     -30  >4AA2 >4A82 >1040 >D00F >4A82 >5282 >0000 >0000
+2     0     3    >52A2 >541C >0654 >D00F >5282 >54A4 >0180 >0002
```

(12.11 RUN MODULE EXAMPLE continued)

```
 ENTER NUMBER:20
  20 FACTORIAL = 2.432902E18
 CALL SUBROUTINE #0.  HERE WE GO.....
 **** SUBROUTINE CALL #0 WORKS! ****
 WE ARE BACK!!!!  COM[0]= 3
 CALL SUBROUTINE #1.
 SUB #1 - CURRENT TASK LISTING
 TASK  PAGE  TIME    SB    WS    PC    SR    BM    EM   CRU   PORT

 *0     0     3    >42A2 >441C >0654 >D40F >4282 >4A82 >0080 >0001
 +1     0     -30  >4AA2 >4A82 >1040 >D00F >4A82 >5282 >0000 >0000
 +2     0     -5   >52A2 >5282 >292E >C40F >5282 >54A4 >0180 >0002
 ENTER NUMBER:_
```

```
PDOS       PDOS                PDOPDOS             PDOS            Aux port console
              PDOS          PDOS                  PDOS
 PDOS    PDPDOS         PPDOS         PDOS             PDOS
    PDOS                     PDOS               PDOS PDOS
 PDOS               PDOS           PDOS PDOS   PPPDOS  PDOS
PDOS PDOS               PDOS                 PDOPDOS PDOS
   PDOS        PDOS             PDOS          PDOS     PDPDOS
   PDOS  PDOS         PDOS           PDPDOS       PDOS
PDOS      PDOS            PDOS             PDOSPDOS    PDOS
 PDOS        PDOS                  PDOS PDOS          PDOS
PDPDOPDOS       PDOSPDOS              PPDOS            PDOS
```