EY-0060E-TP-0001

Programming RSX-11M in MACRO

A Self-Paced Course

Tests/Exercises

digital

EY-0060E-TP-0001

Programming RSX-11M in MACRO

A Self-Paced Course

Tests/Exercises

Prepared by Educational Services
of
Digital Equipment Corporation

Printed in U.S.A.

9/82-14

CONTENTS

9 FILE I/O

10 FILE CONTROL SERVICES

FIGURES

INTRODUCTION

This book contains tests/exercises for two different courses, Programming RSX-11M in MACRO and Programming RSX-11M in FORTRAN. Most of the questions apply to both courses. If a question begins with "In MACRO" or "In FORTRAN", that question applies only to the specified course. Solutions are provided for all tests/exercises. Where it is appropriate, separate solutions are provided for MACRO and FORTRAN. Solutions which involve programs should also be available on-line.

Check the Student Guide in the Student Workbook for your course for information on how to use the tests/exercises.

Using System Services

TEST/EXERCISE

1. Match the function with the type of system service used to perform it.

Function	Type of System Service
___ a. The tasks send data back and forth to each other	1. System and task information
___ b. The tasks read data from a file on disk	2. Task control
___ c. The tasks get input from an operator at a terminal	3. Task communication/coordination
	4. I/O to peripheral devices
	5. File and record access
	6. Memory use

2. Draw a figure to illustrate a method of providing a system service through the Executive.

TEST/EXERCISE

3. What is the other method for providing a system service?

4. Identify two system libraries you might use in writing programs that use system services.

Using System Services

SOLUTION

1. Match the function with the type of system service used to perform it.

	Function	Type of System Service
3	a. The tasks send data back and forth to each other	1. System and task information
5	b. The tasks read data from a file on disk	2. Task control
4	c. The tasks get input from an operator at a terminal	3. Task communication/coordination
		4. I/O to peripheral devices
		5. File and record access
		6. Memory use

2. Draw a figure to illustrate a method of providing a system service through the Executive.

See Figure 1-1 or 1-2

SOLUTION

3. What is the other method for providing a system service?

   Insert the code into your task.

4. Identify two system libraries you might use in writing programs that use system services.

   Any two of the following:

   SYSLIB.OLB
   RSXMAC.SML
   RMSMAC.MLB
   RMSLIB.OLB
   FOROTS.OLB
   F4POTS.OLB

   Also acceptable:

   FCSREF.TSK
   FORRES.TSK
   F4PRES.TSK
   RMSSEQ.TSK

TEST/EXERCISE

1. In MACRO-11

   a. Modify the task READF to use the $C form of the Read Event Flags directive.

   b. Modify the task READF to use the $S form of the Read Event Flags directive.

2. In FORTRAN, modify the task READF to set all of the odd numbered flags from 1 to 15(10).

3. Modify WFLAG and SFLAG to use a global event flag instead of a group global event flag. Omit any unnecessary code in the tasks. Check with your instructor to find out which event flag to use.

4. Write a task which does some work and periodically checks a group global event flag. Have it display a message and exit when the flag has been set. Write another task, or modify SFLAG to set the flag.

5. Add a requested exit AST routine to WFLAG.

6. In MACRO-11, add an odd address trap SST routine to the task SST. Include an instruction which causes the trap to occur.

SOLUTION

1.a

```
 1          .TITLE  READF
 2          .IDENT  /01/
 3          .ENABL  LC                  ; Enable lower case
 4  ;+
 5  ; File LEX21A.MAC
 6  ;
 7  ; Modified to use the $C form of the Read All Event ;;EX
 8  ; Flags directive
 9  ;
10  ; This task starts up, sets event flag 1, reads the
11  ; event flags, moves them into registers R0-R3 and then
12  ; exits. It uses the $ form of the directive calls.
13  ;
14  ; The flags are returned as follows:
15  ;
16  ;               word 0 = event flags 1-16
17  ;               word 1 = event flags 17-32
18  ;               word 2 = event flags 33-48
19  ;               word 3 = event flags 49-64
20  ;-
21
22          .MCALL  RDAF$C,SETF$,EXIT$S,DIR$ ; System macros
23                                                    ;;EX
24  BUFF:   .BLKW   4                   ; Buffer for event flag
25                                      ;  values
26
27  SETF:   SETF$   1                   ; DPB for Set Event Flag
28                                      ;  directive
29
30  START:  CLR     R4                  ; Clear error counter
31          DIR$    #SETF               ; Set event flag 1
32          BCS     ERR1                ; Branch on dir error
33          RDAF$C  BUFF                ; Read the event flags;;EX
34                                      ;  (1 - 64).
35          BCS     ERR2                ; Branch on dir error
36          MOV     BUFF,R0             ; Move the event flag
37          MOV     BUFF+2,R1           ;  values into the
38          MOV     BUFF+4,R2           ;  registers
39          MOV     BUFF+6,R3
40          IOT                         ; Trap and display
41                                      ;  registers
42
43  ; Come here on directive errors
44  ERR2:   INC     R4                  ; R4=2 for read error
45  ERR1:   INC     R4                  ; R4=1 for set event
46                                      ;  flag error
47          MOV     $DSW,R0             ; Error code into R0
48          IOT                         ; Trap and display the
49                                      ;  registers
50          .END    START
```

SOLUTION

l.b

```
 1          .TITLE  READF
 2          .IDENT  /01/
 3          .ENABL  LC                  ; Enable lower case
 4  ;+
 5  ; File LEX21B.MAC
 6  ;
 7  ; Modified to use $S form of the Read All Event Flags ;;EX
 8  ; directive
 9  ;
10  ; This task starts up, sets event flag 1, reads the
11  ; event flags, moves them into registers R0-R3 and then
12  ; exits. It uses the $ form of the directive calls.
13  ;
14  ; The flags are returned as follows:
15  ;
16  ;               word 0 = event flags 1-16
17  ;               word 1 = event flags 17-32
18  ;               word 2 = event flags 33-48
19  ;               word 3 = event flags 49-64
20  ;-
21
22          .MCALL  RDAF$S,SETF$,EXIT$S,DIR$ ; System macros
23                                                      ;;EX
24  BUFF:   .BLKW   4                   ; Buffer for event flag
25                                      ;  values
26
27  SETF:   SETF$   1                   ; DPB for Set Event Flag
28                                      ;  directive
29
30  START:  CLR     R4                  ; Clear error counter
31          DIR$    #SETF               ; Set event flag 1
32          BCS     ERR1                ; Branch on dir error
33          RDAF$S  #BUFF               ; Read the event flags
34                                      ;  (1 - 64).          ;;EX
35          BCS     ERR2                ; Branch on dir error
36          MOV     BUFF,R0             ; Move the event flag
37          MOV     BUFF+2,R1           ;  values into the
38          MOV     BUFF+4,R2           ;  registers
39          MOV     BUFF+6,R3
40          IOT                         ; Trap and display
41                                      ;  registers
42
43  ; Come here on directive errors
44  ERR2:   INC     R4                  ; R4=2 for read error
45  ERR1:   INC     R4                  ; R4=1 for set event
46                                      ;  flag error
47          MOV     $DSW,R0             ; Error code into R0
48          IOT                         ; Trap and display the
49                                      ;  registers
50          .END    START
```

SOLUTION

2.

```
      C       READEF.FTN
      C
      C File LEX22.FTN
      C
      C Modified for exercises. Set odd numbered flags. !!EX
      C
      C This task sets event flag 1 and then reads
      C flags 1 to 16 and displays them
      C
            INTEGER*2   IEVF(16),IDSW
      C Set odd event flags.                                !!EX
            DO 5 K=1,15,2                                   !!EX
            CALL SETEF (K,IDSW)                             !!EX
      C Branch on directive error
            IF (IDSW .LT. 0) GOTO 1000
      5     CONTINUE                                        !!EX
      C Read the event flags into the array ievf. Note
      C that in FORTAN, we can only read 1 flag at a time
            DO 20 I=1,16
            CALL READEF (I,IDSW)
      C Branch on directive error
            IF (IDSW .LT. 0) GOTO 1100
      C Check IDSW value, 2 means set, 0 means clear
      C Set the ievf value accordingly (1 means set, 0
      C means clear)
            IF (IDSW .EQ. 2) GOTO 10
            IEVF(I)=IDSW
            GOTO 20
      10    IEVF(I)=1
      20    CONTINUE
      C Write out flag settings, starting with flag 16.
            WRITE (5,30)
      30    FORMAT (' EVENT FLAGS 16. TO 1. ARE:')
            WRITE (5,40) (IEVF(J), J=16,1,-1)
      40    FORMAT (' ',16I2)
            CALL EXIT
      C Come here on directive errors
      1000  WRITE (5,1010) IDSW
      1010  FORMAT (' ERROR SETTING FLAG. ERROR CODE = ',I5)
            CALL EXIT
      1100  WRITE (5,1110) IDSW
      1110  FORMAT (' ERROR READING FLAG. ERROR CODE = ',I5)
            CALL EXIT
            END
```

SOLUTION

3.

```
        .TITLE  WFLAG
        .IDENT  /01/
        .ENABL  LC                      ; Enable lower case
;+
; FILE LEX23A.MAC
;
; Modified to use global event flag 35.                 ;;EX
;
; This program creates the group global event flags,
; clears event flag 65. and waits for it to be set. When
; the flag is set it writes a message and exits.
;
; Assemble and task-build instructions:
;
;       >MACRO/LIST/OBJECT:WFLAG LB:[1,1]PROGMACS/LIBR-;;EX
;       ->ARY,dev:[ufd]LEX23A
;       >LINK/MAP WFLAG,LB:[1,1]PROGSUBS/LIBRARY
;
; Install and Run instructions:
;
;       Run WFLAG, then run SFLAG. At least one of the
;       tasks must be installed, or else the RUN command
;       will try to install both tasks under the same
;       name, TTnn.
;-
        .MCALL  EXIT$S,WTSE$C,CLEF$C,CRGF$C ; System
                                            ;  macros
        .MCALL  TYPE                    ; Supplied macro

START:  CLR     R0                      ; R0 used to identify
                                        ;  the error
        TYPE    <CLEAR AND THEN WAIT FOR EF 35. TO BE SET>
                                                ;;EX
        CLEF$C  35.                     ; Clear event flag 35.;;EX
        BCS     ERR2                    ; Branch on directive
                                        ;  error
        WTSE$C  35.                     ; Wait for event flag 35
                                        ;  to be set        ;;EX
        BCS     ERR3                    ; Branch on directive
                                        ;  error
        TYPE    <EF 35. HAS BEEN SET. WFLAG WILL NOW EXIT>
                                                        ;;EX
        EXIT$S
ERR3:   INC     R0                      ; R0 = 3 if error on
                                        ; wait for dir
ERR2:   INC     R0                      ; R0 = 2 if error on
                                        ;  clear flag dir
        MOV     $DSW,R1                 ; Place DSW in R1
        IOT                             ; Trap and dump registers
        .END    START
```

SOLUTION

```
1           PROGRAM WFLAG
2   C
3   C FILE LEX23A.FTN
4   C
5   C Modified to use event flag 35(10)                !!EX
6   C
7   C This task creates the group global event flags, and
8   C then clears event flag 65. and waits for it to be set.
9   C When the flag is set, it writes a message and exits
10  C
11  C Install and run instructions:
12  C
13  C       Run WFLAG, then run SFLAG. At least one of the
14  C       tasks must be installed, or else the RUN command
15  C       will try to install both tasks under the same
16  C       name (TTnn)
17  C
18          WRITE (5,20)
19  20      FORMAT (' CLEAR AND WAIT FOR EF 35. TO BE SET')!!EX
20          CALL CLREF (35,IDSW)                       !!EX
21          IF (IDSW .LT. 0) GOTO 1100
22          CALL WAITFR (35,IDSW)                      !!EX
23          IF (IDSW .LT. 0) GOTO 1200
24          WRITE (5,30)
25  30      FORMAT (' EF 35. HAS BEEN SET. FWAIT WILL NOW EXIT')
26  C                                                  !!EX
27          CALL EXIT
28  C Error processing
29  C
30  1100    WRITE (5,1110) IDSW
31  1110    FORMAT (' DIRECTIVE ERROR CLEARING EVENT FLAG 35.
32          1 DSW = ',I5)                              !!EX
33          CALL EXIT
34  1200    WRITE (5,1210) IDSW
35  1210    FORMAT (' DIRECTIVE ERROR WAITING FOR EVENT FLAG
36          1 35. DSW = ',I5)
37          CALL EXIT
38          END
```

```
1           .TITLE  SFLAG
2           .IDENT  /01/
3           .ENABL  LC              ; Enable lower case
4   ;+
5   ; FILE LEX23B.MAC
6   ;
7   ; Modified to use event flag 35.                  ;;EX
8   ;
9   ; This task sets event flag 65.  It assumes that the
10  ; group global event flags have already been created.
11  ;
12  ; Assemble and task-build instructions:
13  ;
14  ;       MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[ufd]SFLAG
15  ;       LINK/MAP SFLAG,LB:[1,1]PROGSUBS/LIBRARY
```

SOLUTION

```
16    ;
17    ; Install and Run notes:
18    ;
19    ;       First run WFLAG, then run SFLAG. At least one of
20    ;       the tasks must be installed, or else the RUN
21    ;       command will try to install both tasks under
22    ;       the same name, TTnn.
23    ;
24    ;-
25            .MCALL  EXIT$S,SETF$C   ; System macros
26            .MCALL  TYPE            ; Supplied macros
27    ;
28    START:  TYPE    <EF 35. IS BEING SET. THEN SFLAG WILL EXIT.>
29                                                    ;;EX
30            SETF$C  35.             ; Set event flag 35. ;;EX
31            BCS     ERR             ; Branch on dir error
32            EXIT$S                  ; Exit
33    ERR:    MOV     $DSW,R1         ; Save DSW
34            IOT                     ; Trap and dump registers
35            .END    START


 1            PROGRAM SFLAG
 2    C
 3    C FILE LEX23B.FTN
 4    C
 5    C Modified to use event flag 35.                !!EX
 6    C
 7    C This task sets event flag 65. It assumes that the
 8    C group global event flags have already been created.
 9    C
10    C Install and run instructions:
11    C
12    C       Run WFLAG, then run SFLAG. At least one of the
13    C       tasks must be installed, or else the RUN command
14    C       will try to install both tasks under the same
15    C       name (TTnn).
16    C
17            WRITE (5,10)
18    10      FORMAT (' EF 35. IS BEING SET. THEN SFLAG WILL EXIT')
19    C                                               !!EX
20            CALL SETEF (35,IDSW)                    !!EX
21    C The DSW value returned for SETEF is 2 if it was set
22    C and 0 if it was clear. A 1 is NOT returned for success
23            IF (IDSW .LT. 0) GOTO 1000
24            CALL EXIT
25    C Error code
26    1000    WRITE (5,1010)
27    1010    FORMAT (' DIRECTIVE ERROR SETTING EF 35. DSW = '
28            1,I4)                                   !!EX
29            CALL EXIT
30            END
```

SOLUTION

4.

```
        .TITLE  LEX24
        .IDENT  /01/
        .ENABL  LC              ; Enable lower case
;+
; FILE LEX24.MAC
;
; This program creates the group global event flags,
; clears event flag 65., does some work and periodically
; cehcks event flag 65. When the flag is set it writes a
; message and exits.
;
; Assemble and task-build instructions:
;
;       MACRO/LIST/OBJECT:WFLAG LB:[1,1]PROGMACS/LIB-;;EX
;       RARY,dev:[ufd]LEX24                             ;;EX
;       LINK/MAP WFLAG,LB:[1,1]PROGSUBS/LIBRARY
;
; Install and Run instructions:
;
;       Run WFLAG, then run SFLAG. At least one of the
;       tasks must be installed, or else the RUN command
;       will try to install both tasks under the same
;       name, TTnn.
;-
        .MCALL  EXIT$S,WTSE$C,CLEF$C,CRGF$C ; System
                                            ;  macros
        .MCALL  TYPE            ; Supplied macro

START:  CLR     R0              ; R0 used to identify
                                ;  the error
        TYPE    <LEX24 IS CREATING THE GROUP GLOBAL EVENT FLAGS>
        CRGF$C                  ; Create group global
                                ;  event flags
        BCC     OK              ; Branch on directive ok
; If group global event flags already exist,
; just display message and continue
        CMP     $DSW,#IE.RSU    ; Check for efs already
                                ;  in existence
        BNE     ERR1            ; Branch on any other
                                ;  dir error
        TYPE    <GROUP GLOBAL EVENT FLAGS ALREADY EXIST>
OK:     TYPE    <CLEAR EF 65. WORK UNTIL IT IS SET>
        CLEF$C  65.             ; Clear event flag 65.
        BCS     ERR2            ; Branch on directive
                                ;  error
AGAIN:  CLR     R1              ; Clear counter     ;;EX
; Loop 2**16 times, then check flag                 ;;EX
LOOP:   INC     R1              ; Increment counter ;;EX
        BNE     LOOP            ; Not yet cycled, loop;;EX
                                ;  again            ;;EX
```

SOLUTION

```
51              TYPE    <COUNTER HAS CYCLED> ; Display message;;EX
52              CLEF$C  65.                 ; Use Clear to read flag;;EX
53              BCS     ERR3                ; Branch on dir error;;EX
54              CMP     $DSW,#IS.SET        ; IS.SET means flag was;;EX
55                                          ;  set                  ;;EX
56              BNE     AGAIN               ; No, loop again        ;;EX
57              TYPE    <EF 65. HAS BEEN SET. LEX24 WILL NOW EXIT>
58              EXIT$S
59      ERR3:   INC     R0                  ; R0 = 3 if error on ;;EX
60                                          ;  Clear Flag dir while
61                                          ;  waiting
62      ERR2:   INC     R0                  ; R0 = 2 if error on
63                                          ;  clear flag dir
64      ERR1:   INC     R0                  ; R0 = 1 if error on
65                                          ;  create group flags dir
66              MOV     $DSW,R1             ; Place DSW in R1
67              IOT                         ; Trap and dump registers
68              .END    START
```

```
 1              PROGRAM LEX24
 2      C
 3      C FILE LEX24.FTN
 4      C
 5      C This task creates the group global event flags, and
 6      C then clears event flag 65. and does some work while
 7      C waiting for it to be set. When the flag is set, it
 8      C writes a message and exits
 9      C
10      C Install and run instructions:
11      C
12      C       Run LEX24, then run SFLAG. At least one of the
13      C       tasks must be installed, or else the RUN command
14      C       will try to install both tasks under the same
15      C       name (TTnn)
16      C
```

SOLUTION

```
17              WRITE (5,10)
18      10      FORMAT (' LEX24 IS CREATING THE GROUP GLOBAL EVENT FLAGS')
19      C                                                       !!EX
20              CALL CRGF (,IDSW)
21              IF (IDSW .LT. 0) GOTO 900
22      15      WRITE (5,20)
23      20      FORMAT (' CLEAR EF 65. WORK UNTIL IT IS SET')
24              CALL CLREF (65,IDSW)
25              IF (IDSW .LT. 0) GOTO 1100
26      22      DO 25 K=1,65535                                 !!EX
27      25      CONTINUE                                        !!EX
28              WRITE (5,28)                                    !!EX
29      28      FORMAT (' COUNTER HAS CYCLED')                  !!EX
30              CALL READEF (65,IDSW)                           !!EX
31              IF (IDSW .LT. 0) GOTO 1200                      !!EX
32              IF (IDSW .NE. 2) GOTO 22                        !!EX
33              WRITE (5,30)
34      30      FORMAT (' EF 65. HAS BEEN SET. LEX24 WILL NOW EXIT')
35              CALL EXIT
36      C Error processing
37      C
38      C Check for code of -17, meaning flags already exist
39      900     IF (IDSW .NE. -17) GOTO 1000
40      C In that case, just display a message and continue.
41              WRITE (5,910)
42      910     FORMAT (' GROUP GLOBAL EVENT FLAGS ALREADY EXIST')
43              GOTO 15
44      C Here for fatal errors, display message and exit
45      1000    WRITE (5,1010) IDSW
46      1010    FORMAT (' DIRECTIVE ERROR CREATING GROUP GLOBAL
47              1EF''S. DSW = ',I5)
48              CALL EXIT
49      1100    WRITE (5,1110) IDSW
50      1110    FORMAT (' DIRECTIVE ERROR CLEARING EVENT FLAG 65.
51              1 DSW = ',I5)
52              CALL EXIT
53      1200    WRITE (5,1210) IDSW
54      1210    FORMAT (' DIRECTIVE ERROR READING EVENT FLAG
55              1 65. DSW = ',I5)                               !!EX
56              CALL EXIT
57              END
```

SOLUTION

5.

```
        .TITLE  WFLAG
        .IDENT  /01/
        .ENABL  LC              ; Enable lower case
;+
; FILE LEX25.MAC
;
; Modified to include a Requested Exit AST      ;;EX
;
; This program creates the group global event flags,
; clears event flag 65. and waits for it to be set. When
; the flag is set it writes a message and exits.
;
; Assemble and task-build instructions:
;
;       MACRO/LIST/OBJECT:WFLAG LB:[1,1]PROGMACS/LIB-;;EX
;       RARY,dev:[ufd]LEX25                           ;;EX
;       LINK/MAP WFLAG,LB:[1,1]PROGSUBS/LIBRARY
;
; Install and Run instructions:
;
;       Run WFLAG, then run SFLAG. At least one of the
;       tasks must be installed, or else the RUN command
;       will try to install both tasks under the same
;       name, TTnn.
;-
        .MCALL  EXIT$S,WTSE$C,CLEF$C,CRGF$C ; System
                                            ;  macros
        .MCALL  SREA$C,ASTX$S   ; System Macros    ;;EX
        .MCALL  TYPE            ; Supplied macro

START:  CLR     R0              ; R0 used to identify
                                ;  the error
        SREA$C  REXAST          ; Set up Requested Exit
                                ;  AST              ;;EX
        BCS     ERR0            ; Branch on dir error
        TYPE    <WFLAG IS CREATING THE GROUP GLOBAL EVENT FLAGS>
        CRGF$C                  ; Create group global
                                ;  event flags
        BCC     OK              ; Branch on directive ok
; If group global event flags already exist,
; just display message and continue
        CMP     $DSW,#IE.RSU    ; Check for efs already
                                ;  in existence
        BNE     ERR1            ; Branch on any other
                                ;  dir error
        TYPE    <GROUP GLOBAL EVENT FLAGS ALREADY EXIST>
OK:     TYPE    <CLEAR AND THEN WAIT FOR EF 65. TO BE SET>
        CLEF$C  65.             ; Clear event flag 65.
        BCS     ERR2            ; Branch on directive
                                ;  error
```

SOLUTION

```
51          WTSE$C  65.             ; Wait for event flag 65
52                                  ;  to be set
53          BCS     ERR3            ; Branch on directive
54                                  ;  error
55          TYPE    <EF 65. HAS BEEN SET. WFLAG WILL NOW EXIT>
56          EXIT$S
57  ; AST Service routine                                   ;;EX
58  REXAST: TYPE    <WHY ME? NOT THIS TIME!!> ; Type message
59                                                          ;;EX
60          ASTX$S                  ; AST exit to return ;;EX
61  ERR3:   INC     R0              ; R0 = 3 if error on
62                                  ; wait for dir
63  ERR2:   INC     R0              ; R0 = 2 if error on
64                                  ;  clear flag dir
65  ERR1:   INC     R0              ; R0 = 1 if error on
66                                  ;  create group flags dir
67  ERR0:   MOV     $DSW,R1         ; Place DSW in R1, leave
68                                  ;  R0=0 for specify ;;EX
69                                  ;  requested exit AST err
70          IOT                     ; Trap and dump registers
71          .END    START
```

```
 1          PROGRAM WFLAG
 2  C
 3  C FILE LEX25.FTN
 4  C
 5  C Modified to include a Requested Exit AST      !!EX
 6  C
 7  C This task creates the group global event flags, and
 8  C then clears event flag 65. and waits for it to be set.
 9  C When the flag is set, it writes a message and exits
10  C
11  C Install and run instructions:
12  C
13  C       Run WFLAG, then run SFLAG. At least one of the
14  C       tasks must be installed, or else the RUN command
15  C       will try to install both tasks under the same
16  C       name (TTnn)
17  C
18          EXTERNAL REXAST                 !!EX
19  C Set up Requested Exit AST             !!EX
20          CALL SREA (REXAST,IDSW)         !!EX
21          IF (IDSW .LT. 0) GOTO 950       !!EX
22          WRITE (5,10)
23  10      FORMAT (' WFLAG IS CREATING THE GROUP GLOBAL EVENT FLAGS')
24          CALL CRGF (,IDSW)
25          IF (IDSW .LT. 0) GOTO 900
26  15      WRITE (5,20)
27  20      FORMAT (' CLEAR AND WAIT FOR EF 65. TO BE SET')
28          CALL CLREF (65,IDSW)
29          IF (IDSW .LT. 0) GOTO 1100
30          CALL WAITFR (65,IDSW)
31          IF (IDSW .LT. 0) GOTO 1200
32          WRITE (5,30)
```

SOLUTION

```
33    30      FORMAT (' EF 65. HAS BEEN SET. FWAIT WILL NOW EXIT')
34            CALL EXIT
35    C Error processing
36    C
37    C Check for code of -17, meaning flags already exist
38    900     IF (IDSW .NE. -17) GOTO 1000
39    C In that case, just display a message and continue.
40            WRITE (5,910)
41    910     FORMAT (' GROUP GLOBAL EVENT FLAGS ALREADY EXIST')
42            GOTO 15
43    C Here for fatal errors, display message and exit
44    950     WRITE (5,960) IDSW                              !!EX
45    960     FORMAT (' DIRECTIVE ERROR SETTING UP AST ROUTINE.
46            1 DSW = ',I5)                                   !!EX
47            CALL EXIT                                       !!EX
48    1000    WRITE (5,1010) IDSW
49    1010    FORMAT (' DIRECTIVE ERROR CREATING GROUP GLOBAL
50            1EF''S. DSW = ',I5)
51            CALL EXIT
52    1100    WRITE (5,1110) IDSW
53    1110    FORMAT (' DIRECTIVE ERROR CLEARING EVENT FLAG 65.
54            1 DSW = ',I5)
55            CALL EXIT
56    1200    WRITE (5,1210) IDSW
57    1210    FORMAT (' DIRECTIVE ERROR WAITING FOR EVENT FLAG
58            1 65. DSW = ',I5)
59            CALL EXIT
60            END
61    C                                                       !!EX
62            SUBROUTINE REXAST                               !!EX
63    C                                                       !!EX
64    C AST service routine                                   !!EX
65    C                                                       !!EX
66            INTEGER PLIST(6),IOWVB                          !!EX
67            REAL TEXT1(6),TEXT2(7)                          !!EX
68            DATA IOWVB/"11000/                              !!EX
69            DATA TEXT1 /'TRYI','NG T','O AB',               !!EX
70            1'ORT ','ME ','EH? '/                           !!EX
71            DATA TEXT2 /'WE W','ON''T',' LET',              !!EX
72            1' YOU',' THI','S TI','ME! '/                   !!EX
73    C Set up for QIO directive                              !!EX
74            CALL GETADR(PLIST(1),TEXT1(1))                  !!EX
75            PLIST(2) = 23                                   !!EX
76            PLIST(3) = "40                                  !!EX
77    C Use QIO directive to display text                     !!EX
78            CALL WTQIO(IOWVB,5,1,,,PLIST)                   !!EX
79    C Set up for 2nd line of text                           !!EX
80            CALL GETADR(PLIST(1),TEXT2(1))                  !!EX
81            PLIST(2) = 27                                   !!EX
82    C Use QIO directive to display text                     !!EX
83            CALL WTQIO(IOWVB,5,1,,,PLIST)                   !!EX
84            RETURN                                          !!EX
85            END                                             !!EX
```

SOLUTION

6.

```
        .TITLE  SST
        .IDENT  /01/
        .ENABL  LC              ; Enable lower case
;
; FILE LEX26.MAC
;
; Modified to include an odd address trap       ;;EX
;
; This task sets up an SST vector table to handle SST's
; for BPT, IOT, and odd address traps. It then executes
; instructions to cause these traps to occur. In each
; SST routine, a message is displayed and then the task
; continues. Finally, a TRAP instruction is executed.
; Since no user SST routine is specified for TRAP, the
; Executive aborts the task.
;
; Assemble and task-build instructions:
;
;       MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[ufd]LEX26
;       LINK/MAP LEX26,LB:[1,1]PROGSUBS/LIBRARY
;
        .MCALL  SVTK$C,EXIT$S   ; External system macros
        .MCALL  TYPE            ; External supplied macro
;
VTABLE: .WORD   ODDTRP,MPTVIO,BPT,IOT ; SST vector table
;                                                   ;;EX
START:  SVTK$C  VTABLE,4        ; Have Executive set up
                                ; SST table
        BPT                     ; BPT instruction
        TST     1               ; Test location 1,  ;;EX
                                ;  causing an odd  ;;EX
                                ;  addr trap       ;;EX
        CLR     120000          ; Clear location 120000,
                                ;  causing a memory
                                ;  protect violation
        IOT                     ; IOT instruction
        EXIT$S                  ; Exit
NEW:    TRAP                    ; TRAP instruction
;
```

SOLUTION

```
40    ; SST routines
41    ;
42    ODDTRP: TYPE    <ODD ADDRESS TRAP CAUGHT> ; Type    ;;EX
43                                              ;  message ;;EX
44            RTI                       ; Return from trap ;;EX
45    MPTVIO: TYPE    <MEMORY PROTECT VIOLATION CAUGHT> ; Type
46                                      ;  message
47            CMP     (SP)+,(SP)+       ; Clean off three
48            TST     (SP)+             ;  specific stack words
49                                      ;  for memory protect SST
50            RTI                       ; Return from trap
51    BPT:    TYPE    <BPT CAUGHT>      ; Type message
52            RTI                       ; Return from trap
53    IOT:    TYPE    <IOT CAUGHT>      ; Type message
54            MOV     #NEW,(SP)         ; Change PC on stack so
55                                      ;  return from trap
56                                      ;  returns to NEW
57            RTI                       ; Return from trap
58            .END    START
```

Using the QIO Directive

TEST/EXERCISE

1. Modify SYNCHQ or ASYNCQ to write prompting text (e.g., "TYPE SOME TEXT: ") before issuing the read.

2. In MACRO-11, modify NUMER, replacing the error handling code with code which writes out an error message plus the appropriate status code. Refer to SYNQER for sample error messages.

3. Modify NOECHO to use one QIO directive to both write the prompt and read the input. Also, have the read timeout if no key is struck for 20(10) seconds, in which case, display a timeout message and exit.

4. Write a task which prints a message on every terminal in the system. The task should break through any pending I/O at the terminal. (Note: This task must be task-built as a privileged task, using the /PRIVILEGED:0 qualifier in the task-build command; /PR:0 in MCR)

Using the QIO Directive

SOLUTION

1.
```
          .TITLE  SYNCHQ
          .IDENT  /01/
          .ENABL  LC                ; Enable lower case
;+
; FILE LEX31.MAC
;
; Modified to display prompting text                ;;EX
;
; This task reads a line of text from the terminal,
; converts all upper case characters to lower case, and
; prints the converted message back at the terminal. It
; uses synchronous QIO directives.
;-
          .MCALL  QIOW$C,QIOW$S,EXIT$S ; External system
                                       ;  macros

IOSB:     .BLKW   2                 ; I/O Status Block
BUFF:     .BLKB   80.               ; Text buffer
PRMPT:    .ASCII  /TYPE SOME TEXT: / ; Prompt          ;;EX
LPRMPT    =.-PRMPT                  ; Length of prompt  ;;EX
          .EVEN                                         ;;EX

START:    CLR     R5                ; Error Count
          CLR     R4                ; Error indicator - 0
                                    ;  means directive error
                                    ;  (DSW in R3), neg
                                    ;  means I/O error
                                    ;  (I/O status in R3)
          QIOW$C  IO.WVB,5,1,,IOSB,,<PRMPT,LPRMPT,40>
                                    ; Display prompt     ;;EX
          BCS     ERR3              ; Branch on dir error;;EX
          TSTB    IOSB              ; Check for I/O error;;EX
          BLT     ERR3A             ; Branch on I/O error;;EX
          QIOW$C  IO.RVB,5,1,,IOSB,,<BUFF,80.> ; Issue
                                               ;  read
          BCS     ERR1              ; Branch on dir error
          TSTB    IOSB              ; Check for I/O error
          BLT     ERR1A             ; Branch on I/O error
          MOV     IOSB+2,R0         ; Get count of characters
                                    ;  typed in
          CLR     R1                ; Offset into buffer to
                                    ;  character
LOOP:     CMPB    BUFF(R1),#'A      ; Check for upper case
                                    ;  ASCII character
          BLT     NEXT              ; Branch if below range
          CMPB    BUFF(R1),#'Z
          BGT     NEXT              ; Branch if above range
; Here if upper case, move to register R2 and convert
          MOVB    BUFF(R1),R2       ; Move to register
          ADD     #32.,R2           ; Convert to lower case
          MOVB    R2,BUFF(R1)       ; Replace in message
```

Using the QIO Directive

SOLUTION

```
52   NEXT:   INC     R1              ; Increment offset into
53                                   ;  buffer to next char
54           SOB     R0,LOOP         ; Decrement count of
55                                   ;  characters left to check
56           QIOW$S  #IO.WVB,#5,#1,,#IOSB,,<#BUFF,IOSB+2,#40>
57                                   ; Write text
58           BCS     ERR2            ; Branch on dir error
59           TSTB    IOSB            ; Check for I/O error
60           BLT     ERR2A           ; Branch on I/O error
61           EXIT$S                  ; Exit
62   ;
63   ; Error code
64   ;
65   ERR3A:  INC     R5              ; R5=3 means Prompt QIO
66                                   ;  error         ;;EX
67   ERR2A:  INC     R5              ; Up error count - 2nd QIO
68   ERR1A:  INC     R5              ;                - 1st QIO
69           MOVB    IOSB,R3         ; I/O error. I/O status
70                                   ;  to R3.
71           DEC     R4              ; Negative value in R4
72                                   ;  means I/O error
73           IOT                     ; Trap and display
74                                   ;  registers
75   ERR3:   INC     R5              ; R5=3 means Prompt QIO
76                                   ;  error         ;;EX
77   ERR2:   INC     R5              ; Up error count - 2nd QIO
78   ERR1:   INC     R5              ;                - 1st QIO
79           MOV     $DSW,R3         ; Directive error. DSW
80                                   ;  to R3, leave R4=0.
81           IOT                     ; Trap and display
82                                   ;  registers
83           .END START
```

Using the QIO Directive

SOLUTION

```
1           PROGRAM ASYNCQ
2     C
3     C FILE  LEX31.FTN
4     C
5     C Modified to display prompting text                !!EX
6     C
7     C This program reads a line of text from the terminal,
8     C converts any upper case characters to lower case and
9     C prints the converted message back at the terminal.
10    C It uses asynchronous QIOs and an event flag for
11    C synchronization.
12    C
13          BYTE IOSB(4),IBUF(80)
14          DIMENSION IPAR(6),K(10)
15          EQUIVALENCE (NUM,IOSB(3))
16          REAL PRMPT(4)                                 !!EX
17          DATA PRMPT /'TYPE',' SOM','E TE','XT: '/!!EX
18          DATA IOWVB/"11000/
19          DATA IORVB/"10400/
20          DATA IVFC/"40/
21    C Set up values for the QIO
22          IUNIT=5
23    C Set up for QIO to issue prompt                      !!EX
24          CALL GETADR(IPAR(1),PRMPT(1))                 !!EX
25          IPAR(2)=16                                    !!EX
26          IPAR(3)="40                                   !!EX
27    C Issue asynchronous write                          !!EX
28          CALL QIO(IOWVB,IUNIT,5,,IOSB,IPAR,IDS)        !!EX
29          IF (IDS .LT. 0) GOTO 780                      !!EX
30          CALL WAITFR(5,IDS)                            !!EX
31          IF (IDS .LT. 0) GOTO 785                      !!EX
32          IF (IOSB(1) .LT. 0) GOTO 790                  !!EX
33    C Set up for read                                   !!EX
34          IPAR(3)=0                                     !!EX
35          IPAR(2)=80
36    C Get the address of the I/O buffer
37          CALL GETADR(IPAR(1),IBUF(1))
38    C Issue the QIO
39          CALL QIO(IORVB,IUNIT,5,,IOSB,IPAR,IDS)
40    C Check the directive status
41          IF (IDS .LT. 0) GO TO 800
42    C Do some work while I/O operation is being performed
43          DO 50 I=1,10
44          K(I)=64*I
45    50    CONTINUE
46    C     Wait for I/O to complete
47          CALL WAITFR(5,IDS)
48    C     Check directive status
49          IF (IDS .LT. 0) GO TO 805
50    C Check the I/O status
51          IF (IOSB(1) .LT. 0) GO TO 810
```

SOLUTION

```
C Convert to lowercase
      DO 100 I=1,NUM
      IF (IBUF(I) .LT. 'A') GO TO 100
      IF (IBUF(I) .GT. "132) GO TO 100
      IBUF(I)=IBUF(I)+32
100   CONTINUE
C Set up I/O Parameter List for write
      IPAR(2)=NUM
      IPAR(3)=IVFC
C Write the converted line to the terminal
      CALL QIO(IOWVB,IUNIT,5,,IOSB,IPAR,IDS)
C     Check directive status
      IF (IDS .LT. 0) GO TO 820
C Wait for the I/O to complete
      CALL WAITFR(5,IDS)
C Check directive status
      IF (IDS .LT. 0) GO TO 825
C Check the I/O status
      IF (IOSB(1) .LT. 0) GO TO 830
      GO TO 850
780   WRITE(5,880)IDS                                    !!EX
      GO TO 850                                          !!EX
785   WRITE(5,885)IDS                                    !!EX
      GO TO 850                                          !!EX
790   WRITE(5,890)IOSB(1)                                !!EX
      GO TO 850                                          !!EX
800   WRITE(5,900)IDS
      GO TO 850
805   WRITE(5,905)IDS
      GO TO 850
810   WRITE(5,910)IOSB(1)
      GO TO 850
820   WRITE(5,920)IDS
      GO TO 850
825   WRITE(5,925)IDS
      GO TO 850
830   WRITE(5,930)IOSB(1)
850   CALL EXIT
880   FORMAT(' DIRECTIVE ERROR ON WRITE OF PROMPT,       !!EX
     1 CODE = ',I4)                                      !!EX
885   FORMAT('DIRECTIVE ERROR ON WAIT FOR WRITE OF       !!EX
     1PROMPT, CODE = ',I4)                               !!EX
890   FORMAT(' I/O ERROR ON WRITE OF PROMPT, CODE =      !!EX
     1 ',I4)                                             !!EX
900   FORMAT(' DIRECTIVE ERROR ON READ, CODE = ',I4)
905   FORMAT(' DIRECTIVE ERROR ON 1ST WAIT, CODE = ',I4)
910   FORMAT(' I/O ERROR ON READ, CODE = ',I4)
920   FORMAT(' DIRECTIVE ERROR ON WRITE, CODE = ',I4)
925   FORMAT(' DIRECTIVE ERROR ON 2ND WAIT, CODE = ',I4)
930   FORMAT(' I/O ERROR ON WRITE, CODE = ',I4)
      END
```

SOLUTION

2.

```
 1          .TITLE  NUMER
 2          .IDENT  /01/
 3          .ENABL  LC                  ; Enable lower case
 4  ;+
 5  ; FILE LEX32.MAC
 6  ;
 7  ; Modified to include error message code          ;;EX
 8  ;
 9  ; This task does a simple addition and outputs the
10  ; results. It demonstrates the use of $EDMSG for
11  ; formatting messages with numeric data
12  ;-
13          .MCALL  QIOW$,EXIT$S,DIR$ ; System macros
14          .MCALL  QIOW$S              ; System macros ;;EX
15          .NLIST  BEX                 ; Do not list binary
16                                      ;  extensions
17  ; Data
18  A:      .WORD   10                  ; 1st addend and start
19                                      ;  of argument block
20  B:      .WORD   22                  ; 2nd addend
21  C:      .BLKW   1                   ; Location for sum
22  ;
23  OUT:    QIOW$   IO.WVB,5,1,,IOSB,,<BUF,,40> ;QIO for
24                                      ;  output message
25  IOSB:   .BLKW   2                   ; I/O status block
26  ;
27  ; Set up for $EDMSG
28  ;
29  BUF:    .BLKB   80.                 ; Output buffer
30  FMES:   .ASCIZ  /%D. WAS ADDED TO %D., GIVING %D./
31                                      ; Format string
32  ; Set up for error messages using $EDMSG          ;;EX
33          .EVEN                                   ;;EX
34  ARG:    .BLKW   1                   ; Argument block;;EX
35  FMT1D:  .ASCIZ  /DIRECTIVE ERROR ON WRITE, DSW = %D/ ;;EX
36  FMT1I:  .ASCIZ  'I/O ERROR ON WRITE, I/O STATUS = %D';;EX
37          .EVEN                                   ;;EX
38
39          .LIST   BEX                 ; List binary extensions
40          .EVEN                       ; Move to word boundary
41  START:  MOV     A,C                 ; Move 1st addend to sum
42                                      ;  word
43          ADD     B,C                 ; Add 2nd addend to form
44                                      ;  sum
45  ; Set up for call to $EDMSG
46          MOV     #BUF,R0             ; Addr of output buffer
47          MOV     #FMES,R1            ; Addr of format string
48          MOV     #A,R2               ; Addr of argument block
49          CALL    $EDMSG              ; Make call, character
50                                      ; count returned in R1
```

SOLUTION

```
51          MOV     R1,OUT+Q.IOPL+2 ; Place # of characters
52                                  ;  to write into IOPL
53                                  ;  in QIO DPB
54          DIR$    #OUT            ; Write output message
55          BCS     ERR1D           ; Branch on dir error
56          TSTB    IOSB            ; Check for I/O error
57          BLT     ERR1I           ; Branch on I/O error
58          EXIT$S
59  ;
60  ; Error code
61  ;
62  ERR1I:  MOV     #FMT1I,R1       ; Format string for  ;;EX
63                                  ;  1st I/O error message
64          MOVB    IOSB,R0         ; Extend sign on I/O ;;EX
65          MOV     R0,ARG          ;  status byte by moving;;EX
66                                  ;  it through R0 to the ;;EX
67                                  ;  argument block    ;;EX
68          BR      EDAWT           ; Branch to common edit;;EX
69                                  ;  and write code    ;;EX
70  ERR1D:  MOV     #FMT1D,R1       ; Format string for 1st;;EX
71                                  ;  directive error   ;;EX
72          MOV     #$DSW,ARG       ; Move DSW to arg block;;EX
73  ; Finish setting up for $EDMSG                       ;;EX
74  EDAWT:  MOV     #BUF,R0         ; Output buffer address;;EX
75          MOV     #ARG,R2         ; Argument block address;;EX
76          CALL    $EDMSG          ; Edit output string ;;EX
77          QIOW$S  #IO.WVB,#5,#1,,,,<#BUF,R1,#40> ; Write;;EX
78                                  ;  out message       ;;EX
79          EXIT$S                  ; Exit               ;;EX
80          .END START
```

SOLUTION

3.

```
        .TITLE  NOECHO
        .IDENT  /01/
        .ENABL  LC                  ; Enable lower case
;+
; FILE LEX33.MAC
;
; Modified to combine QIOs and include timeout  ;;EX
;
; This task writes a prompt and then issues a QIO to read
; from the terminal without echo.  It then displays the
; word which was entered.
;
; Assemble and task-build instructions:
;
;       MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[uic]LEX33
;       LINK/MAP LEX33,PROGSUBS/LIBRARY
;-
        .MCALL  EXIT$S,QIOW$C,QIOW$S ; System macros
        .MCALL  DIRERR,IOERR        ; Supplied macros
;
; Data
;
        .NLIST  BEX                 ; Don't list of binary
                                    ;  extensions
MES:    .ASCII  /SECRET WORD: /     ; Prompt messege
LEN     =       .-MES               ; Length of prompt
BUFF:   .ASCII  <15>/NO LONGER A SECRET WORD: /
                                    ; Preceding remark
BLEN    =       .-BUFF              ; Length of Remark
BUF:    .BLKB   80.                 ; Input buffer
TMOMS:  .ASCII  /READ TIMED OUT/ ; Timeout message ;;EX
LTMOMS  =.-TMOMS                                   ;;EX
        .EVEN                       ; Word align for IOSB
IOSB:   .WORD   0                   ; IOSB is broken into
LENT:   .WORD   0                   ;  two parts for
                                    ;  convenience.
; Define functions locally to allow us of an assignment
; statement to shorten directive statement
IO.RPR  =004400                     ; Define functions
TF.RNE  =20                         ;
TF.TMO  =200                        ;
IO.FNC  =<IO.RPR!TF.RNE!TF.TMO> ; QIO function code
        .LIST BEX                   ; List binary extensions
;
; Code
;
START:  QIOW$C  IO.FNC,5,1,,IOSB,,<BUF,80.,2,MES,LEN,44>
                                    ; Issue read after ;;EX
                                    ; prompt           ;;EX
        BCS     DERR1               ; Branch on dir error
```

SOLUTION

```
51              TSTB    IOSB                ; Check for I/O error
52              BLT     IERR1               ; Branch on I/O error
53              CMPB    IOSB,#IS.TMO        ; Check for timeout    ;;EX
54              BNE     NOTIMO              ; Branch if no timeout;;EX
55              QIOW$C  IO.WVB,5,1,,IOSB,,<TMOMS,LTMOMS,40>    ;;EX
56              BCC     DIR4OK              ; Branch on dir ok -   ;;EX
57                                          ;  need this, too far ;;EX
58                                          ;  for branch
59              JMP     DERR4               ; Jump on dir error ;;EX
60      DIR4OK: TSTB    IOSB                ; Check for I/O error ;;EX
61              BLT     IERR4               ; Branch on I/O error ;;EX
62              EXIT$S                      ; Exit                ;;EX
63      NOTIMO: MOV     LENT,R0             ; Get length of input ;;EX
64              ADD     #BLEN,R0            ; Add length of remark
65              QIOW$S  #IO.WVB,#5,#1,,#IOSB,,<#BUFF,R0,#40>
66                                          ; Write out text
67              BCS     DERR3               ; Branch on dir error
68              TSTB    IOSB                ; Check for I/O error
69              BLT     IERR3               ; Branch on I/O error
70              EXIT$S                      ; Exit
71      ;
72      ; Errors come here
73      ;
74      IERR1:  IOERR   #IOSB,<Error on READ AFTER PROMPT> ;;EX
75                                          ;  Display message and
76      IERR3:  IOERR   #IOSB,<Error on 2nd WRITE> ;  exit
77      IERR4:  IOERR   #IOSB,<Error writing timeout message>;;EX
78      DERR1:  DIRERR  <Error in QIO on READ AFTER PROMPT> ;;EX
79                                          ; Display dir message and
80      DERR3:  DIRERR  <Error in QIO on 2nd WRITE> ;  exit
81      DERR4:  DIRERR  <Error writing timeout message>     ;;EX
82              .END    START
```

Using the QIO Directive

SOLUTION

```
1           PROGRAM NOECHO
2   C
3   C File LEX33.FTN
4   C
5   C Modified to use read after prompt and to timeout !!EX
6   C
7   C This task prompts for input, reads it without echo and
8   C then skips to the next line and displays the input
9   C text and exits.
10  C
11          BYTE    BUFF(80),IOSB(4),CR(1)
12          INTEGER PARM(6)
13          REAL    PROMPT(4)               ! Prompt   !!EX
14  C
15          DATA    IOFNC   /'4620/         ! QIO      !!EX
16  C                                       !  function!!EX
17  C                                       !  code    !!EX
18          DATA    ISTMO /2/               ! Timeout  !!EX
19  C                                       !  status  !!EX
20          DATA    CR /'15/                ! Carriage return character
21          DATA    PROMPT /'SECR','ET W','ORD:','    '/
22  C                                       ! Text     !!EX
23  C Set up the I/O parameter list
24          CALL GETADR (PARM(1),BUFF(1))   ! buffer address
25          PARM(2) = 80                    ! Buffer length
26          PARM(3) = 2                     ! Timeout = 2 !!EX
27  C                                       !  * 10 sec   !!EX
28          CALL GETADR (PARM(4),PROMPT(1)) ! Prompt addr !!EX
29          PARM(5) = 13                    ! Prompt length!!EX
30          PARM(6) = '44                   ! Vertical    !!EX
31  C                                       !  format contr!!EX
32  C Issue read no echo, read after prompt, with timeout !!EX
33          CALL WTQIO (IOFNC,5,1,,IOSB,PARM,IDS)
34          IF (IDS .LT. 0) GO TO 100       ! Dir error?
35          IF (IOSB(1) .LT. 0) GO TO 110   ! I/O error?
36  C Check for timeout
37          IF (IOSB(1) .NE. ISTMO) GOTO 1  ! Branch if no!!EX
38  C                                       !  timeout    !!EX
39          TYPE *,'READ TIMED OUT'         ! Display     !!EX
40  C                                       !  message    !!EX
41          CALL EXIT                       !  and exit   !!EX
42  1       WRITE (5,2) CR,(BUFF(I),I=1,IOSB(3)) ! Echo input
43  2       FORMAT (' ',A1,'NO LONGER A SECRET WORD: ',80A1)
44          CALL EXIT
45  C
46  C Error conditions
47  C
48  100     TYPE *, 'DIRECTIVE ERROR ON READ. STATUS = ',IDS
49          CALL EXIT
50  110     TYPE *, 'I/O ERROR ON READ. CODE = ',IOSB(1)
51          CALL EXIT
52          END
```

SOLUTION

4.

```
        .TITLE  LEX34
        .IDENT  /01/
        .ENABL  LC                      ; Enable lower case
;+
; FILE LEX34.MAC
;
; Solution to Module 3, Lab Exercise 4
;
; Task does a write breakthrough to all terminals.
;
; Assemble and task-build instructions:
;
;       >MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[ufd]-
;       ->LEX34
;       >LINK/MAP/PRIVILEGED:0 LEX34,LB:[1,1]PROGSUBS/-
;       ->LIBRARY
;-
        .MCALL  ALUN$,QIOW$,DIR$,EXIT$S
        .MCALL  DIRERR,IOERR
BUFF:   .ASCII  /HELLO THERE/
        LEN = .-BUFF
        .EVEN
IOSB:   .BLKW   2                       ; I/O status block for QIO
ALUN:   ALUN$   4,TT,0                  ; DPB to assign to TT0:,
                                        ;  will modify for others
QIO:    QIOW$   IO.WLB!TF.WBT!TF.RCU,4,1,,IOSB,,<BUFF,LEN,40>
;
        .ENABLE LSB
START:  MOV     #ALUN,R0                ; R0 => DPB for ALUN$
        MOV     #QIO,R1                 ; R1 => DPB for QIOW$
BRO:    DIR$    R0                      ; Assign LUN
        BCS     ALFAIL                  ; If ALUN$ failed
        DIR$    R1                      ; Type message at TTn:
        BCC     1$                      ; If I/O was queued OK
        DIRERR  <ERROR ON QIOW$>
1$:     CMPB    #IS.SUC,IOSB            ; Did I/O succeed?
        BEQ     2$                      ; Yes
        IOERR   #IOSB,<ERROR ON QIOW$>
2$:     INC     A.LUNU(R0)              ; Next terminal
        BR      BRO
;
; Error from ALUN
ALFAIL: CMP     #IE.IDU,$DSW            ; Did it fail because of
                                        ;  illegal unit #?
        BNE     3$                      ; No, some other error
        EXIT$S                          ; Yes.  Must have passed
                                        ;  the last terminal
3$:     DIRERR  <ERROR ON ALUN$>        ; Other error
        .END    START
```

Using the QIO Directive

SOLUTION

```
1           PROGRAM LEX34
2     C+
3     C FILE LEX34.FTN
4     C
5     C Solution to Module 3, Lab Exercise 4
6     C
7     C Task does a write breakthrough to all terminals.
8     C
9     C Task-build with /PRIVILEGED:0 qualifier
10    C-
11          INTEGER TTUNIT,DSW
12          DATA TTUNIT/0/              ! First output to TT0:
13          INTEGER PARAM(6),IOSB(2)
14          BYTE SUCCOD(2)              ! I/O success codes
15          EQUIVALENCE (SUCCOD,IOSB) ! First bytes of IOSB
16          INTEGER IEIDU               ! Mnemonic for "Illegal
17          DATA IEIDU/-99/             ! Device or Unit" DSW code
18          INTEGER IOFCOD              ! I/O function code
19    C                                 !  mnemonic
20          DATA IOFCOD/"501/           ! Write logical block,
21    C                                 !  write breakthrough,
22    C                                 !  and restore cursor
23    C
24    C Load parameter list
25          CALL GETADR(PARAM(1),'HELLO THERE')
26          PARAM(2) = 11               ! Length of string
27          PARAM(3) = "40              ! Blank for carr. ctrl.
28    10    CALL ASNLUN(4,'TT',TTUNIT,DSW) ! Assign LUN 4 to
29    C                                       !  TTn:
30          IF (DSW.LT.0) GOTO 900
31          CALL WTQIO(IOFCOD,4,1,,IOSB,PARAM,DSW)
32          IF (DSW.LT.0) GOTO 910  ! Directive error
33          IF (SUCCOD(1).NE.1) GOTO 920    ! I/O error
34          TTUNIT = TTUNIT+1
35          GOTO 10
36    C
37    C Error from ASNLUN.  If ASNLUN failed because of illegal
38    C unit number, must have passed the last terminal.  Exit.
39    900   IF (DSW.EQ.IEIDU) CALL EXIT
40          TYPE 905,DSW                ! Other error
41    905   FORMAT (' ERROR ON ASNLUN. DSW = ',I6)
42          CALL EXIT
43    910   TYPE 915,TTUNIT,DSW
44    915   FORMAT (' DIRECTIVE ERROR ON QIO TO TT',O2,':'/
45          1 ' DSW = ',I6)
46          CALL EXIT
47    920   TYPE 925,TTUNIT,SUCCOD(2),SUCCOD(1),IOSB(2)
48    925   FORMAT (' I/O ERROR ON QIO TO TT',O2,':'/
49          1 ' I/O STATUS BLOCK = ',I4,' ','/',I4,' /',I6)
50          CALL EXIT
51          END
```

TEST/EXERCISE

1. Modify RECV1 and SEND1 to synchronize using Suspend and Resume directives instead of event flags.

2. Modify RECV2 so that the display includes the name of the sending task in addition to the data.

3. Write another sender task to send data to RECV2. Modify the receiver so that it receives data from your task only, not from SEND2.

4. Modify SPAWN so that it spawns CLI..., MCR..., or ...DCL several different times and sends a different MCR or DCL command line each time. Display the exit status after each command executes.

5. Write a parent task and an offspring task. Have the parent spawn the offspring. Have the offspring emit status to the parent every five seconds for 30 seconds and then exit. Have the parent display each status value. Optional: Use an AST routine in the parent for synchronization.

Using Directives for Intertask Communication

SOLUTION

1.

```
          .TITLE  SEND1
          .IDENT  /01/
          .ENABL  LC                ; Enable lower case
;+
; FILE LEX41A.MAC
;
; Modified to use Suspend and Resume directives for;;EX
; synchronization                                  ;;EX
;
; This task prompts at TI: for a line of text and sends
; the data to RECV1 for processing.  Synchronization is
; handled through a common event flag.
;
; Assemble and task-build instructions:
;
;       >MACRO/LIST/OBJECT:SEND1 LB:[1,1]PROGMACS/LI-;;EX
;       ->BRARY,dev:[ufd]LEX41A
;       >LINK/MAP SEND1,LB:[1,1]PROGSUBS/LIBRARY
;
; Install and run instructions: RECV1 must be installed
; and run prior to running SEND1. RECV1 continues to run
; until it receives 3 data packets.
;-
          .MCALL  SDAT$C,EXIT$S,RSUM$C ; System macros;;EX
          .MCALL  TYPE,INPUT,DIRERR ; Supplied macros
;
;
BUFFER: .BLKB   26.               ; Data buffer to be sent
;
          .ENABL  LSB             ; Enable local symbol
                                  ;  blocks
;
START:: TYPE    <TYPE A LINE OF TEXT, 26 CHARACTERS OR LESS>
                                  ; Type prompt
          INPUT   #BUFFER,#26.    ; Get text to send
          SDAT$C  RECV1,BUFFER    ; Send data to RECV1 ;;EX
          BCC     1$              ; Branch on directvie ok
          DIRERR  <UNABLE TO QUEUE DATA TO RECV1> ; Display
                                  ; error message and exit
1$:       RSUM$C  RECV1           ; Resume RECV1       ;;EX
          BCC     5$              ; Branch on directive ok;;EX
          DIRERR  <UNABLE TO RESUME RECV1> ;           ;;EX
5$:       EXIT$S                  ; Exit               ;;EX
          .END    START
```

Using Directives for Intertask Communication

SOLUTION

```
1           PROGRAM SEND1
2   C
3   C FILE LEX41A.FTN
4   C
5   C Modified to use Suspend and Receive directives for !!EX
6   C synchronization                                     !!EX
7   C
8   C This task prompts at TI: for a line of text and sends
9   C the data to RECV1 for processing.  Synchronization is
10  C handled through a common event flag.
11  C
12  C Install and run instructions: LEX41B must be        !!EX
13  C installed under the name RECV1 and run prior to     !!EX
14  C running LEX41A. RECV1 continues to run until it     !!EX
15  C receives 3 data packets.
16  C
17          BYTE BUFFER(26)
18          DATA RTASK/6RRECV1 /     ! Receiver task
19  C Prompt for input
20          TYPE *,'TYPE A LINE OF TEXT, 26 CHARACTERS OR LESS'
21          READ (5,10) BUFFER       ! Read text
22  10      FORMAT (26A1)
23          CALL SEND (RTASK,BUFFER,,IDSW) ! Send data !!EX
24          IF (IDSW .LT. 0) GOTO 900 ! Branch on dir error
25          CALL RESUME (RTASK,IDSW)  ! Resume RECV1   !!EX
26          IF (IDSW .LT. 0) GOTO 950 ! Branch on dir err!!EX
27          CALL EXIT                 ! Exit
28  C Error code
29  900     TYPE *,'UNABLE TO QUEUE DATA TO RECV1. DSW = ',IDSW
30          CALL EXIT
31  950     TYPE *,'UNABLE TO RESUME RECV1. DSW = ',IDSW  !!EX
32          CALL EXIT                                      !!EX
33          END
```

SOLUTION

```
1           .TITLE  RECV1
2           .IDENT  /01/
3           .ENABL  LC                  ; Enable lower case
4   ;+
5   ; FILE LEX41B.MAC
6   ;
7   ; Modified to use Suspend and Resume for synchronization;;EX
8   ;
9   ; This task and receives data from any sender task
10  ; (e.g., SEND1).It prints the data on TI:.  Then it
11  ; waits for another data packet. It does this until it
12  ; has received 3 messages and then exits.
13  ;
14  ; This task synchronizes with its sender through an
15  ; event flag.
16  ;
17  ; Assemble and task-build instructions:
18  ;
19  ;       >MACRO/LIST/OBJECT:RECV1 LB:[1,1]PROGMACS/LIB-;;EX
20  ;       ->RARY,dev:[ufd]RECV1                          ;;EX
21  ;       LINK/MAP RECV1,LB:[1,1]PROGSUBS/LIBRARY
22  ;
23  ; Install and run instructions: RECV1 must be installed
24  ; and run before running SEND1.
25  ;-
26          .MCALL  RCVD$C,EXIT$S,SPND$S; System macros ;;EX
27          .MCALL  TYPE,DIRERR         ; Supplied macros
28  ;
29  ;
30  RBUFF:  .BLKW   15.                 ; Receive buffer
31  ;
32          .ENABL  LSB                 ; Enable local symbol
33                                      ;  blocks
34  ;
35  START:  MOV     #3,R5               ; Initialize message
36                                      ;  counter
37  AGAIN:  SPND$S                      ; Suspend self until;;EX
38                                      ;  message arrives
39          BCC     3$                  ; Branch on directive ok
40          DIRERR  <SUSPEND DIRECTIVE FAILED> ; Display ;;EX
41                                      ; error message and exit
42  ; We get here when resumed by SEND1                  ;;EX
43  3$:     RCVD$C  ,RBUFF              ; Receive from anyone
44          BCC     5$                  ; Branch on directive ok
45          DIRERR  <RECEIVE DIRECTIVE FAILED IN "RECV1">
46                                      ; Display error message
47                                      ;  and exit
48  ; Successful receipt
49  5$:     TYPE    <DATA RECEIVED BY "RECV1":> ; Display
50                                              ;  data
```

SOLUTION

```
51          TYPE    #RBUFF+4,#26.   ; Display data sent by
52                                  ;  sender
53          DEC     R5              ; Decrement message
54                                  ;  counter
55          BNE     AGAIN           ; If not yet 0, get
56                                  ;  another message
57          TYPE    <"RECV1" HAS RECEIVED 3 MESSAGES AND WILL NOW EXIT>
58          EXIT$S                  ; Exit after 3 messages
59          .END    START
```

```
 1          PROGRAM RECV1
 2  C
 3  C FILE LEX41B.FTN                          !!EX
 4  C
 5  C Modified to use Suspend and Receive directives for !!EX
 6  C synchronization                                    !!EX
 7  C
 8  C This task receives data from LEX41A. It prints
 9  C the data on TI:.  Then it waits for another data
10  C packet. It does this until it has received 3 messages
11  C and then exits.
12  C
13  C This task synchronizes with its sender through an
14  C event flag.
15  C
16  C Install and run instructions: LEX41B must be          !!EX
17  C installed under the name RECV1 and run before running!!EX
18  C LEX41A.                                               !!EX
19  C
20          INTEGER RBUFF(15)       ! Receive buffer
21  C
22          DO 100 I=1,3
23  10      CALL SUSPND (IDSW)      ! Suspend until SEND1 !!EX
24  C                               !  sends data and resumes
25          IF (IDSW .EQ. 2) GOTO 20                    !!EX
26          TYPE *,'SUSPEND DIRECTIVE FAILED. DSW = ',IDSW!!EX
27          GOTO 1000
28  20      CALL RECEIV (,RBUFF,,IDSW) ! Receive from anyone
29          IF (IDSW .EQ. 1) GOTO 30
30          TYPE *,'RECEIVE DIRECTIVE FAILED IN "RECV1".
31          1 DSW = ',IDSW
32          GOTO 1000
33  30      TYPE *,'DATA RECEIVED BY "RECV1":'
34          WRITE (5,35) (RBUFF(K),K=3,15)
35  35      FORMAT (' ',13A2)
36  100     CONTINUE
37          TYPE *,'"RECV1" HAS RECEIVED 3 MESSAGES AND WILL
38          1 NOW EXIT'
39  1000    CALL EXIT
40          END
```

SOLUTION

2.

```
        .TITLE  RECV2
        .IDENT  /01/
        .ENABL  LC              ; Enable lower case
;
; FILE LEX42.MAC                        ;;EX
;
; Modified to display the sender task name in addition ;;EX
; to the data                                          ;;EX
;
; This task receives data from another task. It prints
; the data, along with a header, on TI:.  Then it waits
; for another data packet, continuing this until it has
; received 3 messages.
;
; This task synchronizes with its sender using RCST$.
; Because of this synchronization, the tasks can be run
; in any order, with any relative priorities.
;
; Assemble and task build instructions:
;
;       >MACRO/LIST/OBJECT:RECV2 LB:[1,1]PROGMACS/LIB-;;EX
;       ->RARY,dev:[ufd]LEX42A                       ;;EX
;       >LINK/MAP RECV2,LB:[1,1]PROGSUBS/LIBRARY
;
; Install and run instructions: RECV2 must be installed.
;
        .MCALL  RCST$C,RCVD$C,EXIT$S ; System macros
        .MCALL  TYPE,DIRERR     ; Supplied macros
;
RBUFF:  .BLKW   15.             ; Receive buffer
TASKNM: .BLKW   3               ; Buffer for task name;;EX
;
        .ENABL  LSB             ; Enable local symbol
                                ;  blocks
;
START:  MOV     #3,R5           ; Set up message counter
RECEIV: RCST$C  ,RBUFF          ; Receive from anyone
        BCC     5$              ; Branch on directive ok
        DIRERR  <RECEIVE DIRECTIVE FAILED IN "RECV2">
                                ; Display error message
                                ;  and exit
; Successful receipt or unstopped by another task. First
; check for unstopped after being stopped, in which case
; we have to receive the data
5$:     CMP     $DSW,#IS.SET    ; Were we stopped due to
                                ;  no data
        BNE     6$              ; If not, we have a data
                                ;  packet
        RCVD$C  ,RBUFF          ; Now get the packet
        BCC     6$              ; Branch on directive ok
```

SOLUTION

```
51          DIRERR  <RECEIVE DIR FAILED AFTER "RECV2" UNSTOPPED>
52                                  ; Display error message
53                                  ;  and exit
54  ; Convert task name from Radix-50 to ASCII
55  6$:     MOV     #TASKNM,R0      ; Address for converted;;EX
56                                  ;  name                ;;EX
57          MOV     RBUFF,R1        ; Word to be converted;;EX
58          CALL    $C5TA           ; Convert it           ;;EX
59          MOV     RBUFF+2,R1      ; Next word to be      ;;EX
60                                  ;  converted           ;;EX
61          CALL    $C5TA           ; Convert it           ;;EX
62          TYPE    <DATA RECEIVED BY "RECV2":> ; Display text
63          TYPE    #TASKNM,#6      ; Display task name    ;;EX
64          TYPE    #RBUFF+4,#26.   ;  and data sent
65  ; Had to change SOB - too far for branch!              ;;EX
66          DEC     R5              ; Decrement message    ;;EX
67                                  ;  counter             ;;EX
68          BEQ     DONE            ; Branch if done       ;;EX
69          JMP     RECEIV          ; Receive again if not;;EX
70                                  ;  yet 3 messages      ;;EX
71  DONE:   TYPE    <"RECV2" HAS RECEIVED 3 MESSAGES>
72          TYPE    <AND WILL NOW EXIT> ; Type exit message
73          EXIT$S                  ; Exit
74          .END    START
```

```
 1          PROGRAM RECV2
 2  C
 3  C FILE LEX42.FTN                !!EX
 4  C
 5  C Modified to display the sender task name in addition !!EX
 6  C to the data                                          !!EX
 7  C
 8  C This task receives data from another task (e.g. SEND2).
 9  C It prints the data, along with a header, on TI:.  Then
10  C it waits for another data packet, continuing this
11  C until it has received 3 messages.
12  C
13  C This task synchronizes with its sender using RCST.
14  C Because of this synchronization, the tasks can be run
15  C in any order, with any relative priorities.
16  C
17  C Install and run instructions: LEX42 must be installed!!EX
18  C under the name RECV2.                                 !!EX
19  C
```

Using Directives for Intertask Communication

SOLUTION

```
20    C
21            INTEGER RBUFF(15)          ! Receive buffer
22            INTEGER DSW,ISSET
23            INTEGER TASKNM(3)          ! Buffer for ASCII form!!EX
24    C                                  !  of task name        !!EX
25            DATA ISSET/2/              ! DSW code mnemonic
26    C
27    C
28            DO 100, I=1,3
29            CALL RCST(,RBUFF,DSW)      ! Receive from anyone
30            IF (DSW.GE.0) GOTO 50
31            Type *,'RECEIVE DIRECTIVE FAILED IN "RECV2".
32            1 DSW = ',DSW              ! Display error message
33            GOTO    1000               !  and exit
34    C
35    C Successful receipt or unstopped by another task. First
36    C check for unstopped after being stopped, in which case
37    C we have to receive the data
38    50      IF (DSW.NE.ISSET) GOTO 60 ! Were we stopped due
39    C                                 !  to no data? If not
40    C                                 !  (NE), we have a
41    C                                 !  data packet
42    C Stopped due to no data:
43            CALL RECEIV(,RBUFF,,DSW)   ! Now get the packet
44            IF (DSW.EQ.1) GOTO 60
45            TYPE *,'RECEIVE DIRECTIVE FAILED AFTER "RECV2"
46            1UNSTOPPED. DSW = ',DSW    ! Display error
47            GOTO 1000                  !  message and exit
48    C Display data
49    60      CALL R50ASC (6,RBUFF,TASKNM)          !!EX
50            TYPE 75,TASKNM,(RBUFF(J),J=3,15)      !!EX
51    75      FORMAT (' DATA RECEIVED BY "RECV2":'/1X,3 !!EX
52            1A2,1X,13A2)                          !!EX
53    100     CONTINUE
54    C Have received 3 messages
55            TYPE *,'"RECV2" HAS RECEIVED 3 MESSAGES AND WILL
56            1 NOW EXIT'
57    1000    CALL EXIT                  ! Exit
58            END
```

SOLUTION

3.

```
        .TITLE  LEX43A
        .IDENT  /01/
        .ENABL LC                   ; Enable lower case
;
; FILE LEX43A.MAC                           ;;EX
;
; A second sender to RECV2
;
; This task prompts at TI: a line of text and sends the
; data to task RECV2 for processing.  Synchronization is
; handled through RECV2's stop bit. RECV2 will continue
; to run until it receives 3 messages. RECV2 and LEX43A
; may be run in any order.
;
; Assemble and task build instructions:
;
;       MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[ufd]LEX43A
;       LINK/MAP LEX43A,LB:[1,1]PROGSUBS/LIBRARY
;
; Install and run instructions: LEX43B must be installed
; under the name RECV2
;
        .MCALL  SDAT$C,USTP$C,EXIT$S ; System macros
        .MCALL  TYPE,INPUT,DIRERR ; Supplied macros
;
BUFFER: .BLKB   26.                 ; Send buffer
;
        .ENABL  LSB                 ; Enable local symbol
                                    ;  blocks
;
START:: TYPE    <TYPE A LINE OF TEXT, 26 CHARACTERS OR LESS>
                                    ; Display prompt
        INPUT   #BUFFER,#26.        ; Issue read
        SDAT$C  RECV2,BUFFER        ; Queue data to RECV2
        BCC     1$                  ; Branch on directive ok
        DIRERR  <UNABLE TO QUEUE DATA TO "RECV2">
                                    ; Display error message
                                    ;  and exit
1$:     USTP$C  RECV2               ; Unstop RECV2
        BCC     2$                  ; Branch on directive ok
        CMP     $DSW,#IE.ITS        ; Isn't he stopped?
        BEQ     2$                  ; That's ok, he'll pick
                                    ;  up data when he
                                    ;  executes RCDS$
        CMP     $DSW,#IE.ACT        ; Is he not active?
        BEQ     2$                  ; If not, he'll pick up
                                    ;  data when activated
        DIRERR  <UNABLE TO UNSTOP "RECV2">  ; Any other
                                    ;  error is bad
2$:     EXIT$S                      ; Exit
        .END    START
```

Using Directives for Intertask Communication

SOLUTION

```
1           PROGRAM LEX43A
2   C
3   C FILE LEX43A.FTN                              !!EX
4   C
5   C A second sender task to send data to RECV2     !!EX
6   C
7   C This task prompts at TI: for a line of text and sends
8   C the data to RECV2 for processing.  The receiver will
9   C continue to run until it receives 3 messages.
10  C Synchronization is handled through RECV2's stop bit.
11  C RECV2 and LEX43A may be run in any order.
12  C
13  C Install and run instructions: LEX43B must be   !!EX
14  C installed under the name RECV2.                !!EX
15  C
16          BYTE BUFFER(26)          ! Send buffer
17          INTEGER DSW
18          REAL RECV2
19          DATA RECV2/5HRECV2/      ! Receiving task name
20          INTEGER IEITS,IEACT      ! Error mnemonics
21          DATA IEITS,IEACT/-8,-7/
22  C
23          TYPE *,'TYPE A LINE OF TEXT, 26 CHARACTERS OR LESS'
24          READ (5,5) BUFFER
25  5       FORMAT (26A1)
26          CALL SEND(RECV2,BUFFER,,DSW) ! Send data to RECV2
27          IF (DSW.EQ.1) GOTO 10
28          TYPE *,'UNABLE TO QUEUE DATA TO "RECV2". DSW = '
29          1,DSW
30  10      CALL USTP(RECV2,DSW)     ! Unstop RECV2
31          IF (DSW.EQ.1) GOTO 20    ! Branch on directive ok
32          IF (DSW.EQ.IEITS) GOTO 20 ! Isn't he stopped?
33  C                                !  That's ok, he'll pick
34  C                                !  up data when he
35  C                                !  executes RCDS$
36          IF (DSW.EQ.IEACT) GOTO 20 ! Is he not active? If
37  C                                !  not, he'll pick up
38  C                                !  data when activated
39          TYPE *,'UNABLE TO UNSTOP "RECV2". DSW = ',DSW
40                                   ! Any other error is bad
41  20      CALL EXIT                ! Exit
42          END
```

SOLUTION

```
 1            .TITLE  RECV2
 2            .IDENT  /01/
 3            .ENABL  LC                  ; Enable lower case
 4    ;
 5    ; FILE LEX43B.MAC                   ;;EX
 6    ;
 7    ; Modified to recieve only from LEX43A  ;;EX
 8    ; NOTE: THE TASK WILL EXIT WITH A NO DATA QUEUED ERROR;;EX
 9    ; IF SEND2 SENDS DATA AND UNSTOPS THE TASK. MORE      ;;EX
10    ; COMPLICATED CODING IS NEEDED TO HAVE THIS TASK      ;;EX
11    ; DISTINGUISH BETWEEN TASKS WHICH SEND DATA AND UNSTOP;;EX
12    ; IT                                                  ;;EX
13    ;
14    ;
15    ; This task receives data from another task. It prints
16    ; the data, along with a header, on TI:.  Then it waits
17    ; for another data packet, continuing this until it has
18    ; received 3 messages.
19    ;
20    ; This task synchronizes with its sender using RCST$.
21    ; Because of this synchronization, the tasks can be run
22    ; in any order, with any relative priorities.
23    ;
24    ; Assemble and task-build instructions:
25    ;
26    ;       >MACRO/LIST/OBJECT:RECV2 LB:[1,1]PROGMACS/LIB-;;EX
27    ;       ->RARY,dev:[ufd]LEX43B                      ;;EX
28    ;       >LINK/MAP RECV2,LB:[1,1]PROGSUBS/LIBRARY
29    ;
30    ; Install and run instructions: RECV2 (LEX43B) must be;;EX
31    ; installed under the name RECV2.                     ;;EX
32    ;
33           .MCALL  RCST$C,RCVD$C,EXIT$S ; System macros
34           .MCALL  TYPE,DIRERR     ; Supplied macros
35    ;
36    RBUFF: .BLKW   15.             ; Receive buffer
37    ;
38           .ENABL  LSB             ; Enable local symbol blocks
39    ;
40    START: MOV     #3,R5           ; Set up message counter
41    RECEIV: RCST$C  LEX43A,RBUFF   ; Receive from just ;;EX
42                                   ;  LEX43A          ;;EX
43           BCC     5$              ; Branch on directive ok
44           DIRERR  <RECEIVE DIRECTIVE FAILED IN "RECV2">
45                                   ; Display error message
46                                   ;  and exit
```

SOLUTION

```
47   ; Successful receipt or unstopped by another task. First
48   ; check for unstopped after being stopped, in which case
49   ; we have to receive the data
50   5$:     CMP     $DSW,#IS.SET    ; Were we stopped due to
51                                   ;  no data
52           BNE     6$              ; If not, we have a data
53                                   ;   packet
54           RCVD$C  LEX43A,RBUFF    ; Now get the packet
55           BCC     6$              ; Branch on directive ok
56           DIRERR  <RECEIVE DIR FAILED AFTER "RECV2" UNSTOPPED>
57                                   ; Display error message
58                                   ;  and exit
59   6$:     TYPE    <DATA RECEIVED BY "RECV2":> ; Display
60                                               ;  text and
61           TYPE    #RBUFF+4,#26.               ;  data sent
62   ;       SOB     R5,RECEIV       ; Decrement message
63                                   ;  counter. Receive again
64                                   ;  if haven't received 3
65                                   ;  yet
66           DEC     R5                              ;;EX
67           BEQ     DONE                            ;;EX
68           JMP     RECEIV                          ;;EX
69   DONE:   TYPE    <"RECV2" HAS RECEIVED 3 MESSAGES AND WILL NOW EXIT>
70                                                   ; Type exit
71                                                   ;  message
72           EXIT$S                  ; Exit
73           .END    START
```

```
 1           PROGRAM RECV2
 2   C
 3   C FILE LEX43B.FTN                !!EX
 4   C
 5   C Modified to receive only from LEX43A          !!EX
 6   C NOTE: TASK WILL EXIT WITH A NO DATA QUEUED ERROR IF!!EX
 7   C SEND2 SENDS DATA. MORE COMPLICATED CODE IS NEEDED  !!EX
 8   C TO CHECK FOR SEND2 SENDING DATA AND UNSTOPPING RECV2!!EX
 9   C
10   C This task receives data from another task (e.g. SEND2).
11   C It prints the data, along with a header, on TI:.  Then
12   C it waits for another data packet, continuing this
13   C until it has received 3 messages.
14   C
15   C This task synchronizes with its sender using RCST.
16   C Because of this synchronization, the tasks can be run
17   C in any order, with any relative priorities.
18   C
19   C Install and run instructions: LEX43B must be       !!EX
20   C installed under the name RECV2.                    !!EX
21   C
```

SOLUTION

```
22    C
23            INTEGER RBUFF(15)          ! Receive buffer
24            INTEGER DSW,ISSET
25            REAL TASKNM                ! Task name array  !!EX
26            DATA TASKNM /6RLEX43A/     ! Task name in Radix-50!!EX
27            DATA ISSET/2/              ! DSW code mnemonic
28    C
29    C
30            DO 100, I=1,3
31            CALL RCST(TASKNM,RBUFF,DSW) ! Receive from LEX43A
32            IF (DSW.GE.0) GOTO 50
33            Type *,'RECEIVE DIRECTIVE FAILED IN "RECV2",
34            1 DSW = ',DSW              ! Display error message
35            GOTO    1000               !  and exit
36    C
37    C Successful receipt or unstopped by another task. First
38    C check for unstopped after being stopped, in which case
39    C we have to receive the data
40    50      IF (DSW.NE.ISSET) GOTO 60 ! Were we stopped due
41    C                                 !  to no data? If not
42    C                                 !  (NE), we have a
43    C                                 !  data packet
44    C Stopped due to no data:
45            CALL RECEIV(TASKNM,RBUFF,,DSW)  ! Now get the !!EX
46    C                                 ! packet
47            IF (DSW.EQ.1) GOTO 60
48            TYPE *,'RECEIVE DIRECTIVE FAILED AFTER "RECV2"
49            1UNSTOPPED. DSW = ',DSW    ! Display error
50            GOTO 1000                  !  message and exit
51    C Display data
52    60      TYPE 75,(RBUFF(J),J=3,15)
53    75      FORMAT (' DATA RECEIVED BY "RECV2":'/1X,13A2)
54    100     CONTINUE
55    C Have received 3 messages
56            TYPE *,'"RECV2" HAS RECEIVED 3 MESSAGES AND WILL
57            1 NOW EXIT'
58    1000    CALL EXIT                  ! Exit
59            END
```

SOLUTION

4.

```
        .TITLE  SPAWN
        .IDENT  /02/
        .ENABL  LC              ; Enable lower case
;
; File LEX44.MAC                                        ;;EX
;
; This program spawns MCR..., passes it a series of    ;;EX
; command lines, waits for each to exit, and           ;;EX
; displays each command's exit status.                 ;;EX
;
; Assemble and task-build instructions:
;
;       MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[ufd]LEX44
;       LINK/MAP LEX44,LB:[1,1]PROGSUBS/LIBRARY
;
        .MCALL  SPWN$,EXIT$S,WTSE$C,QIOW$S,QIOW$C
                                ; System macros
        .MCALL  DIRERR,IOERR    ; Supplied macros
        .NLIST  BEX             ; Inhibit listing of
                                ;  binary extensions

CMD1:   .ASCII "PIP *.MAC/LI"   ; Command line          ;;EX
LEN1    =.-CMD1                 ; Length of command     ;;EX
CMD2:   .ASCII  /ACT/                                   ;;EX
LEN2 = .-CMD2                                           ;;EX
CMD3:   .ASCII  /TIM/                                   ;;EX
LEN3 = .-CMD3                                           ;;EX

SMES:   .ASCII  /SPAWN IS STARTING AND WILL SPAWN/      ;;EX
        .ASCII  / MCR COMMANDS/ ; Startup message       ;;EX
LSMES   =.-SMES                 ; Length of message
        .EVEN
IOSB:   .BLKW   2               ; I/O status block
EXSTAT: .BLKW   8.              ; Exit status block

CMDTBL: .WORD   CMD1,LEN1       ; Table indexing        ;;EX
        .WORD   CMD2,LEN2       ; MCR commands          ;;EX
        .WORD   CMD3,LEN3                               ;;EX
        .WORD   0               ; End of table          ;;EX

SPAWN:  SPWN$   MCR...,,,,,,1,,EXSTAT                   ;;EX

BUFF:   .BLKB   80.             ; Output message buffer
; Format string:
FMT:    .ASCII  /%NSPAWN REPORTING: COMMAND/            ;;EX
        .ASCIZ  / COMPLETED. EXIT STATUS WAS %D.%N/      ;;EX
        .EVEN
START:  QIOW$C  IO.WVB,5,1,,IOSB,,<SMES,LSMES,40>
        BCS     ERR1D           ; Branch on dir error
        TSTB    IOSB            ; Check for I/O error
        BLT     ERR1I           ; Branch on I/O error
```

SOLUTION

```
52          MOV     #CMDTBL,R3          ; R3 => command table ;;EX
53          MOV     #SPAWN,R4           ; R4 => SPAWN DPB     ;;EX
54  GETCMD: MOV     (R3)+,S.PWCA(R4)    ; Set command address ;;EX
55          BEQ     DONE                ; If 0, end of list   ;;EX
56          MOV     (R3)+,S.PWCL(R4)    ; Command length      ;;EX
57          DIR$    R4                  ; Spawn MCR...        ;;EX
58          BCS     ERR2                ; Branch on dir error
59          WTSE$C  1                   ; Wait for task to exit
60          BCS     ERR3                ; Branch on dir error
61          BIC     #177400,EXSTAT      ; Clear high order byte
62                                      ;  of exit status
63          MOV     #BUFF,R0            ; Set up for $EDMSG
64          MOV     #FMT,R1             ;
65          MOV     #EXSTAT,R2          ;
66          CALL    $EDMSG              ; Edit status message
67          QIOW$S  #IO.WVB,#5,#1,,#IOSB,,<#BUFF,R1,#40>
68                                      ; Display exit status
69          BCS     ERR4D               ; Branch on dir error
70          TSTB    IOSB                ; Check for I/O error
71          BLT     ERR4I               ; Branch on I/O error
72          BR      GETCMD              ; Get next command    ;;EX
73  DONE:   EXIT$S                      ; Exit                ;;EX
74  ; Error handling code - ; Display error message and exit
75  ERR1D:  DIRERR  <ERROR WRITING STARTUP MESSAGE>
76  ERR1I:  IOERR   #IOSB,<ERROR WRITING STARTUP TEXT>
77  ERR2:   DIRERR  <ERROR SPAWNING MCR>                      ;;EX
78  ERR3:   DIRERR  <ERROR WAITING FOR EVENT FLAG>
79  ERR4D:  DIRERR  <ERROR WRITING EXIT STATUS>               ;;EX
80  ERR4I:  IOERR   #IOSB,<ERROR WRITING EXIT STATUS>         ;;EX
81          .END    START
```

SOLUTION

```
1           PROGRAM SPWN
2   C
3   C File LEX44.FTN
4   C
5   C This program spawns ...DCL, passes it a series of !!EX
6   C command lines, waits for each to exit, and        !!EX
7   C displays each command's exit status.              !!EX
8   C
9   C Data
10          INTEGER EXSTAT(8),PLIST(6),DSW
11          BYTE BUFF(80)
12  C Commands to be spawned:                           !!EX
13  C
14  C       DIR *.MAC                                   !!EX
15  C       SHOW TASKS/ACTIVE                           !!EX
16  C       SHOW TIME                                   !!EX
17  C
18          REAL CMD(5,3)                               !!EX
19          DATA CMD/'DIR ','*.MA','C'   , 0    , 0 ,
20          1        'SHOW',' TAS','KS/A','CTIV','E',
21          2        'SHOW',' TIM','E'    , 0    , 0/    !!EX
22          INTEGER LEN(3)
23          DATA LEN/9,17,9/
24  C
25          REAL DCL
26          DATA DCL/6R...DCL/
27  C
28  C Code
29          WRITE (5,15)              ! Write message
30  15      FORMAT (' SPAWN IS STARTING AND WILL SPAWN ',
31          1 'DCL COMMANDS')                           !!EX
32          DO 30,I=1,3
33          CALL SPAWN(DCL,,,1,,EXSTAT,,CMD(1,I),LEN(I)
34          1 ,,,DSW)                                   !!EX
35                                    ! Spawn DCL
36          IF (DSW.LT.0) GOTO 900    ! Branch on dir error
37          CALL WAITFR(1,DSW)        ! Wait for task to exit
38          IF (DSW.LT.0) GOTO 910    ! Branch on dir error
39          WRITE (5,25) EXSTAT(1).AND.'377 ! Display low
40                                    ! byte of exit status
41  25      FORMAT (' SPAWN REPORTING: COMMAND COMPLETED.',
42          1 ' EXIT STATUS WAS ',I1,'.')
43  30      CONTINUE
44          CALL EXIT                 ! Exit
45  C Error handling code
46  900     TYPE *,'ERROR SPAWNING DCL. DSW = ',DSW
47          GOTO 1000
48  910     TYPE *,'ERROR WAITING FOR EVENT FLAG. DSW = ',DSW
49  1000    CALL EXIT
50          END
```

SOLUTION

5.

```
1           .TITLE  LEX45A
2           .IDENT  /01/
3           .ENABL  LC                  ; Enable lower case
4   ;+
5   ; File LEX45A.MAC
6   ;
7   ; Solution to Module 4, Lab Exercise 5 - Part A, parent
8   ; task
9   ;
10  ; Task spawns LEX45B and reports status of that task.
11  ; Synchronization is through an AST routine.
12  ;-
13          .GLOBL  $EDMSG
14          .MCALL  CLEF$,WTSE$C,SPWN$C,EXIT$S,QIOW$,DIR$
15          .MCALL  SETF$C,CNCT$C,ASTX$S,QIOW$C,ABRT$C
16          .MCALL  DIRERR
17  ;
18  QIO:    QIOW$   IO.WVB,5,2,,,,<OUTBUF,0,40>  ; Set mss
19                                  ;  length later
20  CLEF:   CLEF$   1
21  ; $EDMSG argument block:
22  EDMARG: .WORD   OFEMST              ; => OFEMST or OFEXIT
23  STATUS: .BLKW   8.                  ; Offspring status block
24  ;
25  MSG:    .ASCIZ  /OFFSPRING %I. STATUS = %D%N/
26  OFEMST: .ASCIZ  /EMITTED STATUS/
27  OFEXIT: .ASCIZ  /EXITED/
28  OUTBUF: .BLKB   200.
29          .EVEN
30  ;
31          .ENABLE LSB
32  START:  MOV     #QIO,R4             ; R4 => QIOW$ DPB
33          DIR$    #CLEF               ; CLEF 1, used to synch
34                                      ;  with AST routine
35          BCS     ERR1
36          SPWN$C  LEX45B,,,,,,,ASTRTN,STATUS  ; Spawn LEX45B
37          BCS     ERR2
38  1$:     WTSE$C  1                   ; Wait until AST occurs
39                                      ;  and AST routine sets
40                                      ;  flag
41          BCS     ERR3
42          MOV     #OUTBUF,R0          ; R0 => $EDMSG output
43          MOV     #MSG,R1             ; R1 => $EDMSG input
44          MOV     #EDMARG,R2          ; R2 => $EDMSG arguments
45          MOVB    STATUS,R5           ; Extend sign on status
46          MOV     R5,STATUS           ;  byte, also keep in R5
47          BMI     2$                  ; Minus values mean EMST
48          MOV     #OFEXIT,EDMARG      ; >= 0 means EXIT
49  2$:     CALL    $EDMSG
50          MOV     R1,Q.IOPL+2(R4) ; Load message length
```

SOLUTION

```
51            DIR$    R4                  ; QIOW$ to TI:
52            BCS     ERR4
53            TST     R5                  ; Did offsprins exit?
54            BGE     3$                  ; Yes
55            DIR$    #CLEF               ; No. Clear EF 1 again
56            BCS     ERR5
57            BR      1$                  ; Wait
58    3$:     EXIT$S                      ; Once offsprins exits,
59                                        ;  so should parent
60    ;
61    ERR1:   DIRERR  <ERROR ON INITIAL CLEF$>
62    ERR2:   DIRERR  <ERROR SPAWNING LEX45B>
63    ERR3:   DIRERR  <ERROR ON WTSE$C>
64    ERR4:   DIRERR  <ERROR ON QIOW$>
65    ERR5:   DIRERR  <ERROR ON CLEF$>
66    ;
67    ; AST routine, entered when offsprins emits status
68    ; (negative status value) or exits (positive status
69    ; value)
70    ;
71    ASTRTN: SETF$C  1                   ; Awaken main code
72            BCS     ERR6
73            CMP     $DSW,#IS.SET        ; If set, main code is
74                                        ;  not ready yet
75            BEQ     OVRRUN              ; We've been overrun
76            TST     STATUS              ; Has offsprins exited?
77            BGE     4$                  ; If so, don't try to
78                                        ;  reconnect
79            CNCT$C  LEX45B,,ASTRTN,STATUS
80            BCS     ERR7
81    4$:     TST     (SP)+               ; Clean up stack from AST
82            ASTX$S                      ; Let main code run
83    ;
84    ; If a new status comes in before we're done with the old
85    ; one, somethins is wrons.  Stop everythins.
86    ;
87    OVRNMS: .ASCII  /STATUS RECEIVED BEFORE READY. /
88            .ASCII  /  ABORTING BOTH TASKS./
89    OVRNML = .-OVRNMS
90            .EVEN
91    ;
92    OVRRUN: QIOW$C  IO.WVB,5,3,,,,<OVRNMS,OVRNML,40>
93            ABRT$C  LEX45B              ; Abort offsprins
94            BCS     ERR8
95            EXIT$S                      ; Exit this task
96    ;
97    ERR6:   DIRERR  <ERROR FROM SETF$ IN AST ROUTINE>
98    ERR7:   DIRERR  <ERROR CONNECTING TO OFFSPRING>
99    ERR8:   DIRERR  <ERROR ABORTING OFFSPRING>
100           .END    START
```

Using Directives for Intertask Communication

SOLUTION

```
1           PROGRAM LEX45A
2     C+
3     C File LEX45A.FTN
4     C
5     C Solution to Module 4, Lab Exercise 5 - Part A, parent
6     C task
7     C
8     C Task spawns LEX45B and reports status of that task.
9     C Negative status values are used when emitting status,
10    C positive values when exiting.
11    C
12    C Synchronization is through an event flag.
13    C-
14    C
15          REAL LEX45B
16          DATA LEX45B/6RLEX45B/
17          INTEGER STATUS(8),DSW
18    C
19    C Spawn LEX45B:
20          CALL SPAWN (LEX45B,,,1,,STATUS,,,,,,DSW)
21          IF (DSW.LT.0) GOTO 900
22    10    CALL WAITFR(1,DSW)          ! Wait until EXIT or
23    C                                 !  EMIT STATUS occurs
24          IF (DSW.LT.0) GOTO 910
25          IF (STATUS(1).GE.0) GOTO 20  ! Offspring exited
26    C                                   ! Emitted status:
27          TYPE 15,STATUS(1).OR."177400 ! Display status,
28    C                                   !  neg sign extended
29    C                                   !  to get neg value
30    15    FORMAT (' OFFSPRING EMITTED STATUS. STATUS = ',
31          1I4/)
32          CALL CNCT (LEX45B,1,,STATUS,,DSW) ! Reconnect
33          IF (DSW.LT.0) GOTO 920
34          GOTO 10                     ! Wait for next status
35    C
36    C Offspring exited:
37    C
38    20    TYPE 25,STATUS(1).AND."377
39    25    FORMAT (' OFFSPRING EXITED. STATUS = ',I4/)
40          CALL EXIT                   ! Once offspring exits,
41    C                                 !  so should parent
42    C
43    900   TYPE *,'ERROR SPAWNING LEX45B. DSW = ',DSW
44          GOTO 1000
45    910   TYPE *,'ERROR ON WAITFR. DSW = ',DSW
46          GOTO 1000
47    920   TYPE *,'ERROR CONNECTING TO OFFSPRING. DSW = ',
48          1DSW
49    1000  CALL EXIT
50          END
```

SOLUTION

```
1           .TITLE  LEX45B
2           .IDENT  /01/
3           .ENABL  LC                  ; Enable lower case
4   ;+
5   ; File LEX45B.MAC
6   ;
7   ; Solution to Module 4, Lab Exercise 5 - Part B,
8   ; offspring task
9   ;
10  ; This task is spawned by LEX45A.  It emits a negative
11  ; status every 5 seconds, then exits after 30 seconds
12  ; (6 emits, then an exit).
13  ;
14  ; If an emit status fails because this task was not
15  ; connected to the parent, another emit status will be
16  ; tried 5 seconds later.  Two consecutive failures cause
17  ; this task to exit with an error message.
18  ;
19  ; This task must be installed under task name LEX45B.
20  ;-
21          .MCALL  EMST$S,QIOW$C,WTSE$C,MRKT$C,EXIT$S
22          .MCALL  DIRERR
23  ;
24  NCNCT:  .ASCII  /LEX45B NOT CONNECTED TO ANY PARENT/
25          .BYTE   15,12
26          .ASCII  /WILL TRY AGAIN IN 5 SECONDS/
27  NCNCTL = .-NCNCT
28          .EVEN
29  ;
30  START:  CLR     R0                  ; R0 = exit status
31          CLR     R1                  ; R1 =  0 means last
32                                      ;  attempt to emit status
33                                      ;  suceeded. R0 < 0 means
34                                      ;  it failed because we
35                                      ;  were not connected
36          MOV     #6,R3               ; R3 = number of emits
37                                      ;  yet to be issued
38  EMST:   DEC     R3                  ; Set timer (again)?
39          BMI     EXIT                ; No, just exit
40          MRKT$C  1,5,2               ; Set timer for 5 seconds
41          BCS     ERR1
42          DEC     R0                  ; Use status < 0 when
43                                      ;  emitting
44          EMST$S  ,R0                 ; Emit to parent
45          BCS     1$                  ; Failed.  Why?
46          CLR     R1                  ; Note success
47          BR      WAIT                ; Wait for 5 secs to pass
48  1$:     CMP     $DSW,#IE.ITS        ; Failed because not
49                                      ;  connected?
50          BNE     ERR2                ; Any other reason, quit
```

SOLUTION

```
51            TST     R1                  ; Failed last time too?
52            BMI     ERR2                ; Then give up
53            DEC     R1                  ; Else note we failed this
54                                        ;  time
55                                        ; And announce the
56                                        ;  problem:
57            QIOW$C  IO.WVB,5,2,,,,<NCNCT,NCNCTL,40>
58            BCS     ERR3
59                                        ; And try again in 5 secs
60    WAIT:   WTSE$C  1                   ; Wait for 5 secs to pass
61            BCS     ERR4
62            BR      EMST
63    EXIT:   EXIT$S                      ; Exit (with success)
64    ;
65    ; Directive errors
66    ;
67    ERR1:   DIRERR  <ERROR ON MRKT$C>
68    ERR2:   DIRERR  <ERROR EMITTING TO PARENT>
69    ERR3:   DIRERR  <ERROR ON QIOW$C>
70    ERR4:   DIRERR  <ERROR ON WTSE$C>
71            .END    START
```

```
 1            PROGRAM LEX45B
 2    C+
 3    C File LEX45B.FTN
 4    C
 5    C Solution to Module 4, Lab Exercise 5 - Part B,
 6    C offspring task
 7    C
 8    C This task is spawned by LEX45A.  It emits a negative
 9    C status every 5 seconds, then exits after 30 seconds
10    C (6 emits, then an exit).
11    C
12    C If an emit status fails because this task was not
13    C connected to the parent, another emit status will be
14    C tried 5 seconds later.  Two consecutive failures cause
15    C this task to exit with an error message.
16    C
17    C This task must be installed under task name LEX45B.
18    C-
```

Using Directives for Intertask Communication

SOLUTION

```
19    C
20            INTEGER DSW,IEITS
21            DATA IEITS/-8/                ! Error mnemonic
22            LOGICAL*1 ERLAST              ! Flag if last EMST
23    C                                     !  failed because we were
24    C                                     !  not connected
25            DATA ERLAST/.FALSE./
26    C
27            DO 50,I=1,6                   ! Issue 6 EMSTs
28            CALL MARK (1,5,2,DSW)         ! Set timer for 5 seconds
29            IF (DSW.LT.0) GOTO 900
30            CALL EMST(,(-I),DSW)          ! Emit to parent
31            IF (DSW.LT.0) GOTO 20         ! Failed.  Why?
32            ERLAST = .FALSE.              ! Note success
33            GOTO 30                       ! Wait for 5 secs to pass
34    20      IF (DSW.NE.IEITS) GOTO 910 ! Failed for reason
35    C                                        !  other than not
36    C                                        !  connected
37            IF (ERLAST) GOTO 910          ! Failed last time too?
38    C                                     !  Then give up.
39            ERLAST = .TRUE.               ! Else note we failed
40                                          !  this time
41    C                                     ! And announce the
42    C                                     !  problem:
43            TYPE 25
44    25      FORMAT ('LEX45B NOT CONNECTED TO ANY PARENT'/
45            1 'WILL TRY AGAIN IN 5 SECONDS')
46    C                                     ! And try again in 5 secs
47    30      CALL WAITFR(1,DSW)            ! Wait for 5 secs to pass
48            IF (DSW.LT.0) GOTO 920
49    50      CONTINUE
50            CALL EXIT                     ! Exit (with success)
51    C
52    C Directive errors
53    C
54    900     TYPE *,'ERROR ON MRKT. DSW = ',DSW
55            GOTO 1000
56    910     TYPE *,'ERROR EMITTING TO PARENT. DSW = ',DSW
57            GOTO 1000
58    920     TYPE *,'ERROR ON WAITFR. DSW = ',DSW
59    1000    CALL EXIT
60            END
```

Memory Management Concepts

TEST/EXERCISE

1. Write 'M' if the statement applies to mapped systems, 'U' if it applies to unmapped systems, or 'M,U' if it applies to both.

   ___ a. Physical addresses up to 32K words accessible with 16-bit addressing.

   ___ b. Physical addresses up to 128K words accessible with 18-bit addressing.

   ___ c. Program relocation possible without having to program or task-build again.

   ___ d. Detection of memory protection violations.

   ___ e. Program executes only at physical addresses that match the virtual addresses created by the task builder.

   ___ f. Virtual address limit of 32K words.

2. Fill in the headings and the missing values in Figure 1.

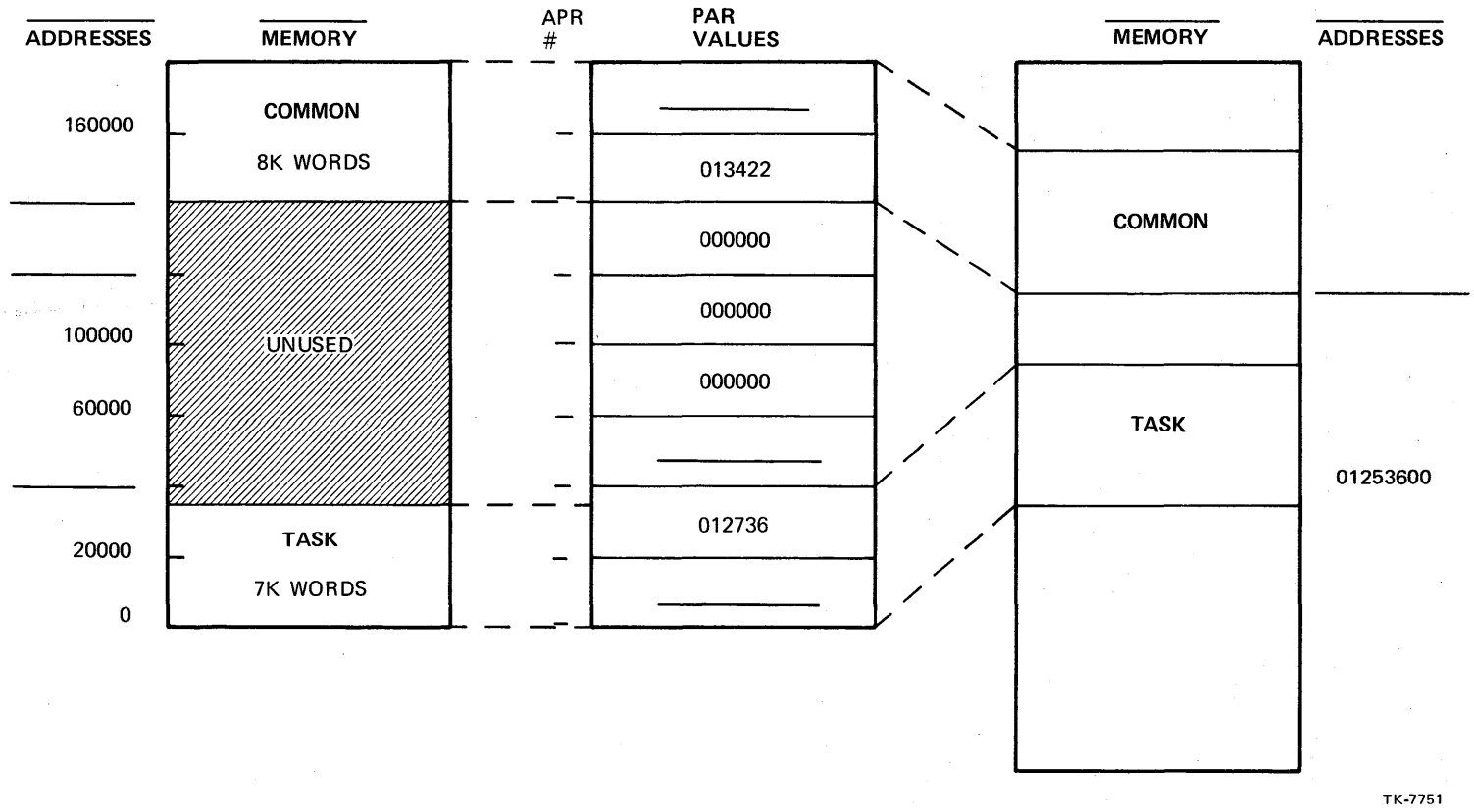


Figure 1 Virtual Addresses, APRs and Physical Addresses in a Mapped System

SOLUTION

1. Write 'M' if the statement applies to mapped systems, 'U' if it applies to unmapped systems, or 'M,U' if it applies to both.

   U a. Physical addresses up to 32K words accessible with 16-bit addressing. (M is also acceptable since 32K words is the limit of 16-bit addressing even on a mapped system.)

   M b. Physical addresses up to 128K words accessible with 18-bit addressing.

   M c. Program relocation possible without having to program or task-build again.

   M d. Detection of memory protection violations.

   U e. Program executes only at physical addresses that match the virtual addresses created by the task builder.

   M,U f. Virtual address limit of 32K words.

2. Fill in the headings and the missing values in Figure 1.

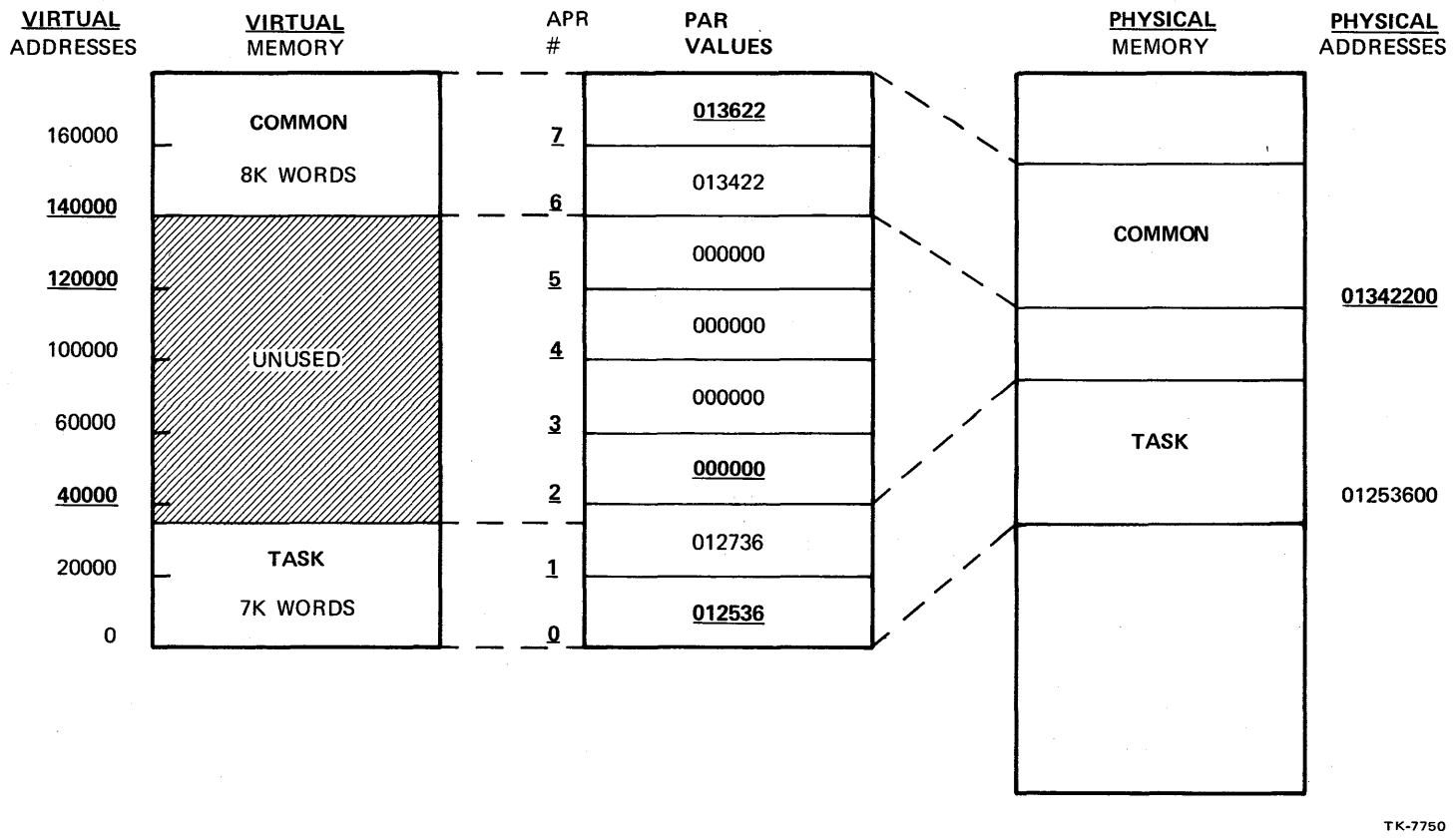


Figure 1 Virtual Addresses, APRs and Physical Addresses in a Mapped System

Overlaying Techniques

TEST/EXERCISE

The following is an output display from a task.

```
MAIN CALLING SUBROUTINE G
G CALLING SUBROUTINE G1
G1 RUNNING
MAIN CALLING SUBROUTINE H1
H1 RUNNING
MAIN CALLING SUBROUTINE H
H CALLING SUBROUTINE H1
H1 RUNNING
H CALLING SUBROUTINE H2
H2 RUNNING
MAIN EXITING
```

The calling sequence parallels the output display.

1. Draw an overlay tree diagram or a memory allocation diagram for a possible overlay structure for the task.

2. Write the modules MAIN, G, G1, H, H1, and H2. Assemble or compile each one.

3. Task-build and run the task without overlays. Obtain a map.

4. Task-build and run the task with all disk-resident overlays. Obtain a map.

5. Task-build and run the task with all memory-resident overlays. Obtain a map.

TEST/EXERCISE

6. Task-build and run the task with G, G1 and H in memory-resident overlays, H1 and H2 in disk-resident overlays. Obtain a map.

7. Use the map to fill in the following table:

Type of Overlay	Starting Virtual Address of G	Starting Virtual Address of H1
No Overlays		
All Disk-Resident Overlays		
All Memory-Resident Overlays		
Disk-Resident and Memory-Resident Overlays		

8. (Optional) Task-build Example 6-5 so that the module TOTAL is in an overlay segment.

9. (Optional) Modify Exercise 8. Add a subroutine RTOTAL which displays the running total after each job (e.g., THE TOTAL SO FAR IS xx).

NOTE 1

For debugging, place RTOTAL in the root segment and place all calls to RTOTAL in the module MAIN.

NOTE 2

Once RTOTAL is debugged, build the task with RTOTAL in an existing overlay segment. Place RTOTAL so that the task executes the fastest. (Still use autoload, but place RTOTAL to minimize loading of overlay segments.)

SOLUTION

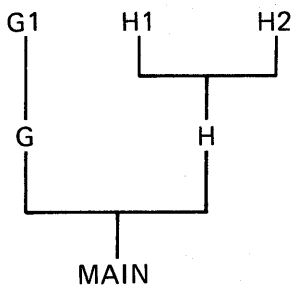
The following is an output display from a task.

```
MAIN CALLING SUBROUTINE G
G CALLING SUBROUTINE G1
G1 RUNNING
MAIN CALLING SUBROUTINE H1
H1 RUNNING
MAIN CALLING SUBROUTINE H
H CALLING SUBROUTINE H1
H1 RUNNING
H CALLING SUBROUTINE H2
H2 RUNNING
MAIN EXITING
```

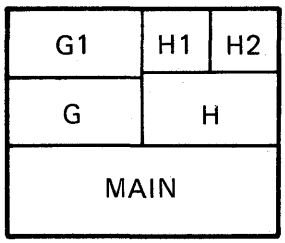
The calling sequence parallels the output display.

1. Draw an overlay tree diagram or a memory allocation diagram for a possible overlay structure for the task.

OVERLAY TREE



MEMORY ALLOCATION DIAGRAM



TK-7744

SOLUTION

2.

```
        .TITLE  MAIN
        .IDENT  /01/
        .ENABLE LC                      ; Enable lower case
;
; File LEX6A.MAC
;
; Mainline routine for Module 6, Lab Exercises 1-6.
; Illustrate different overlays and their effects.
;
        .GLOBL  G,H1,H                  ; Subroutines called
        .MCALL  QIOW$C,EXIT$S
        .MCALL  DIRERR
;
; Messages
;
CGMS:   .ASCII  /MAIN CALLING SUBROUTINE G/
CGML = .-CGMS
CH1MS:  .ASCII  /MAIN CALLING SUBROUTINE H1/
CH1ML = .-CH1MS
CHMS:   .ASCII  /MAIN CALLING SUBROUTINE H/
CHML = .-CHMS
EXMS:   .ASCII  /MAIN EXITING/
EXML = .-EXMS
        .EVEN
;
; For each routine, type message then call routine
;
START:  QIOW$C  IO.WVB,5,1,,,,<CGMS,CGML,40>
        BCS     IOFAIL
        CALL    G
        QIOW$C  IO.WVB,5,1,,,,<CH1MS,CH1ML,40>
        BCS     IOFAIL
        CALL    H1
        QIOW$C  IO.WVB,5,1,,,,<CHMS,CHML,40>
        BCS     IOFAIL
        CALL    H
        QIOW$C  IO.WVB,5,1,,,,<EXMS,EXML,40>
        EXIT$S
IOFAIL:: DIRERR <ERROR ON QIO TO TERMINAL>
        .END    START
```

Overlaying Techniques

SOLUTION

```
1           PROGRAM MAIN
2   C
3   C File LEX6A.FTN
4   C
5   C Mainline routine for Module 6, Lab Exercises 1-6.
6   C Illustrate different overlays and their effects.
7   C
8   C For each routine, type message then call routine
9   C
10          TYPE *,'MAIN CALLING SUBROUTINE G'
11          CALL    G
12          TYPE *,'MAIN CALLING SUBROUTINE H1'
13          CALL    H1
14          TYPE *,'MAIN CALLING SUBROUTINE H'
15          CALL    H
16          TYPE *,'MAIN EXITING'
17          CALL EXIT
18          END
```

```
1           .TITLE  G
2           .IDENT  /01/
3           .ENABL  LC                  ; Enable lower case
4   ;
5   ; File LEX6B.MAC
6   ;
7   ; Subroutine for Module 6, Lab Exercises 1-6.
8   ; Illustrate different overlays and their effects.
9   ;
10          .GLOBL  G1                  ; Subroutine called
11          .GLOBL  IOFAIL              ; Error routine
12          .MCALL  QIOW$C
13  ;
14  ; Messages
15  ;
16  CG1MS:  .ASCII  /G CALLING SUBROUTINE G1/
17  CG1ML = .-CG1MS
18          .EVEN
19  ;
20  ; Type message then call routine
21  ;
22  G::     QIOW$C  IO.WVB,5,1,,,,<CG1MS,CG1ML,40>
23          BCS     ERROR
24          CALL    G1
25          RETURN
26  ERROR:  JMP     IOFAIL
27          .END
```

Overlaying Techniques

SOLUTION

```
1            SUBROUTINE G
2    C
3    C File LEX6B.FTN
4    C
5    C Subroutine for Module 6, Lab Exercises 1-6.
6    C Illustrate different overlays and their effects.
7    C
8    C Type message then call routine
9    C
10           TYPE *,'G CALLING SUBROUTINE G1'
11           CALL     G1
12           RETURN
13           END
```

```
1            .TITLE   G1
2            .IDENT   /01/
3            .ENABL   LC                ; Enable lower case
4    ;
5    ; File LEX6C.MAC
6    ;
7    ; Subroutine for Module 6, Lab Exercises 1-6.
8    ; Illustrate different overlays and their effects.
9    ;
10           .GLOBL   IOFAIL            ; Error routine
11           .MCALL   QIOW$C
12   ;
13   ; Messages
14   ;
15   G1RUN:   .ASCII   /G1 RUNNING/
16   G1RUNL = .-G1RUN
17           .EVEN
18   ;
19   ; Type message then return
20   ;
21   G1::     QIOW$C   IO.WVB,5,1,,,,<G1RUN,G1RUNL,40>
22            BCS      ERROR
23            RETURN
24   ERROR:   JMP      IOFAIL
25           .END
```

Overlaying Techniques

SOLUTION

```
1           SUBROUTINE G1
2   C
3   C File LEX6C.FTN
4   C
5   C Subroutine for Module 6, Lab Exercises 1-6.
6   C Illustrate different overlays and their effects.
7   C
8   C Type message then return
9   C
10          TYPE *,'G1 RUNNING'
11          RETURN
12          END
```

```
1           .TITLE  H
2           .IDENT  /01/
3           .ENABL  LC                  ; Enable lower case
4   ;
5   ; File LEX6D.MAC
6   ;
7   ; Subroutine for Module 6, Lab Exercises 1-6.
8   ; Illustrate different overlays and their effects.
9   ;
10          .GLOBL  H1,H2               ; Subroutines called
11          .GLOBL  IOFAIL              ; Error routine
12          .MCALL  QIOW$C
13  ;
14  ; Messages
15  ;
16  CH1MS:  .ASCII  /H CALLING SUBROUTINE H1/
17  CH1ML = .-CH1MS
18  CH2MS:  .ASCII  /H CALLING SUBROUTINE H2/
19  CH2ML = .-CH2MS
20          .EVEN
21  ;
22  ; Type message then call routine
23  ;
24  H::     QIOW$C  IO.WVB,5,1,,,,<CH1MS,CH1ML,40>
25          BCS     ERROR
26          CALL    H1
27          QIOW$C  IO.WVB,5,1,,,,<CH2MS,CH2ML,40>
28          BCS     ERROR
29          CALL    H2
30          RETURN
31  ERROR:  JMP     IOFAIL
32          .END
```

Overlaying Techniques

SOLUTION

```
1           SUBROUTINE H
2   C
3   C File LEX6D.FTN
4   C
5   C Subroutine for Module 6, Lab Exercises 1-6.
6   C Illustrate different overlays and their effects.
7   C
8   C Type message then call routine
9   C
10          TYPE *,'H CALLING SUBROUTINE H1'
11          CALL H1
12          TYPE *,'H CALLING SUBROUTINE H2'
13          CALL H2
14          RETURN
15          END
```

```
1           .TITLE  H1
2           .IDENT  /01/
3           .ENABL  LC                  ; Enable lower case
4   ;
5   ; File LEX6E.MAC
6   ;
7   ; Subroutine for Module 6, Lab Exercises 1-6.
8   ; Illustrate different overlays and their effects.
9   ;
10          .GLOBL  IOFAIL              ; Error routine
11          .MCALL  QIOW$C
12  ;
13  ; Messages
14  ;
15  H1RUN:  .ASCII  /H1 RUNNING/
16  H1RUNL = .-H1RUN
17          .EVEN
18  ;
19  ; Type message then return
20  ;
21  H1::    QIOW$C  IO.WVB,5,1,,,,<H1RUN,H1RUNL,40>
22          BCS     ERROR
23          RETURN
24  ERROR:  JMP     IOFAIL
25          .END
```

SOLUTION

```
 1            SUBROUTINE H1
 2    C
 3    C File LEX6E.FTN
 4    C
 5    C Subroutine for Module 6, Lab Exercises 1-6.
 6    C Illustrate different overlays and their effects.
 7    C
 8    C Type message then return
 9    C
10            TYPE *,'H1 RUNNING'
11            RETURN
12            END
```

```
 1            .TITLE  H2
 2            .IDENT  /01/
 3            .ENABL  LC                ; Enable lower case
 4    ;
 5    ; File LEX6F.MAC
 6    ;
 7    ; Subroutine for Module 6, Lab Exercises 1-6.
 8    ; Illustrate different overlays and their effects.
 9    ;
10            .GLOBL  IOFAIL            ; Error routine
11            .MCALL  QIOW$C
12    ;
13    ; Messages
14    ;
15    H2RUN:  .ASCII  /H2 RUNNING/
16    H2RUNL = .-H2RUN
17            .EVEN
18    ;
19    ; Type message then return
20    ;
21    H2::    QIOW$C  IO.WVB,5,1,,,,,<H2RUN,H2RUNL,40>
22            BCS     ERROR
23            RETURN
24    ERROR:  JMP     IOFAIL
25            .END
```

```
 1            SUBROUTINE H2
 2    C
 3    C File LEX6F.FTN
 4    C
 5    C Subroutine for Module 6, Lab Exercises 1-6.
 6    C Illustrate different overlays and their effects.
 7    C
 8    C Type message then return
 9    C
10            TYPE *,'H2 RUNNING'
11            RETURN
12            END
```

SOLUTION

3.
```
; Module 6, Lab Exercise 3
;
; Task-build command to build MACRO-11 without overlays
;
        >LINK/MAP LEX6A,LEX6B,LEX6C,LEX6D,LEX6E,LEX6F,-
        ->LB:[1,1]PROGSUBS/LIBRARY
;
; LEX6A = MAIN
; LEX6B = G
; LEX6C = G1
; LEX6D = H
; LEX6E = H1
; LEX6F = H2



; Module 6, Lab Exercise 3
;
; Task-build command to build FORTRAN with no overlays
;
        >LINK/MAP LEX6A,LEX6B,LEX6C,LEX6D,LEX6E,LEX6F,-
        ->LB:[1,1]F4POTS/LIBRARY
;
; LEX6A = MAIN
; LEX6B = G
; LEX6C = G1
; LEX6D = H
; LEX6E = H1
; LEX6F = H2
```

SOLUTION

4.

```
; Module 6, Lab Exercise 4
;
; .ODL file for building MACRO-11 with all disk resident
; overlays
        .ROOT   LEX6A-PROGSUBS/LB-*(LEX6B-LEX6C,OVRH)
OVRH:   .FCTR   LEX6D-(LEX6E,LEX6F)
;
; LEX6A = MAIN
; LEX6B = G
; LEX6C = G1
; LEX6D = H
; LEX6E = H1
; LEX6F = H2
;
        .END
```

```
; Module 6, Lab Exercise 4
;
; .ODL file for building FORTRAN with all disk-resident
; overlays
        .ROOT   LEX6A-FLIB-*(LEX6B-LEX6C-FLIB,HSEGS)
HSEGS:  .FCTR   LEX6D-FLIB-(LEX6E-FLIB,LEX6F-FLIB)
FLIB:   .FCTR   LB:[1,1]F4POTS/LB
;
; LEX6A = MAIN
; LEX6B = G
; LEX6C = G1
; LEX6D = H
; LEX6E = H1
; LEX6F = H2
        .END
```

5.

```
; Module 6, Lab Exercise 5
;
; .ODL file for MACRO-11 with all memory-resident
; overlays
;
        .ROOT   LEX6A-PROGSUBS/LB-*!(LEX6B-LEX6C,OVRH)
OVRH:   .FCTR   LEX6D-!(LEX6E,LEX6F)
;
; LEX6A = MAIN
; LEX6B = G
; LEX6C = G1
; LEX6D = H
; LEX6E = H1
; LEX6F = H2
;
        .END
```

SOLUTION

```
; Module 6, Lab Exercise 5
;
; .ODL file for FORTRAN with all memory-resident overlays
;
        .ROOT   LEX6A-FLIB-*!(LEX6B-LEX6C-FLIB,HSEGS)
HSEGS:  .FCTR   LEX6D-FLIB-!(LEX6E-FLIB,LEX6F-FLIB)
FLIB:   .FCTR   LB:[1,1]F4POTS/LB
;
; LEX6A = MAIN
; LEX6B = G
; LEX6C = G1
; LEX6D = H
; LEX6E = H1
; LEX6F =H2
;
        .END
```

6.

```
; Module 6, Lab Exercise 6
;
; .ODL file for MACRO-11 with some memory-resident, some
; disk-resident overlays
        .ROOT   LEX6A-PROGSUBS/LB-*!(LEX6B-LEX6C,OVRH)
OVRH:   .FCTR   LEX6D-(LEX6E,LEX6F)
;
; LEX6A =MAIN
; LEX6B = G
; LEX6C = G1
; LEX6D = H
; LEX6E = H1
; LEX6F = H2
;
        .END
```

```
; Module 6, Lab Exercise 6
;
; .ODL file for FORTRAN with some disk-resident, some
; memory-residetn overlays
;
        .ROOT   LEX6A-FLIB-*!(LEX6B-LEX6C-FLIB,HSEGS)
HSEGS:  .FCTR   LEX6D-FLIB-(LEX6E-FLIB,LEX6F-FLIB)
FLIB:   .FCTR   LB:[1,1]F4POTS/LB
;
; LEX6A = MAIN
; LEX6B = G
; LEX6C = G1
; LEX6D = H
; LEX6E = H1
; LEX6F = H2
;
        .END
```

SOLUTION

7. Use the map to fill in the following table:

Type of Overlay	Starting Virtual Address of G	Starting Virtual Address of H1
No Overlays		
All Disk-Resident Overlays	Answers will vary depending on students' particular solution.	
All Memory-Resident Overlays		
Disk-Resident and Memory-Resident Overlays		

8.
```
; Module 6, Lab Exercise 8
;
; .ODL file in MACRO-11 to place TOTAL in an overlay
; segment.
; All overlays are disk-resident
        .ROOT   MAIN-*(A-(JOB1,JOBXX),B,TOTAL)
        .END
```

```
; Module 6, Lab Exercise 8
;
; .ODL file in FORTRAN to place TOTAL in an overlay
; segment.
; All overlays are disk-resident
        .ROOT   MAIN-FLIB-*(OVRA,B-FLIB,TOTAL-FLIB)
OVRA:   .FCTR   A-FLIB-(JOB1-FLIB,JOBXX-FLIB)
FLIB:   .FCTR   LB:[1,1]F4POTS/LB
        .END
```

SOLUTION

9.

```
        .TITLE  MAIN
        .IDENT  /01/
        .ENABL  LC                      ; Enable lower case
;+
; FILE LEX69A.MAC                       ;;EX
;
; Modified to call RTOTAL to display the running ;;EX
; total after each call to A                     ;;EX
;
; This program prints a message and then calls
; subroutine A. Subroutine A asks whether to perform Job
; 1 or Job 2. It then calls either subroutine JOB1 or
; JOB2 which performs the Job and displays the results.
; MAIN then calls subroutine B. Subroutine B displays a
; message and exits. MAIN then calls subroutine A 3
; more times, keeping a grand total of the operations.
; Finally, it displays the grand total and exits.
;
; Task-build instructions: Use LEX69A.ODL as the input;;EX
; file.
;-
        .MCALL QIOW$C,EXIT$S,QIOW$S ; Supplied macros
        .NLIST  BEX                     ; Do not list binary
                                        ;  extensions
        .BLKW   1024.*4                 ; Leave space to make
                                        ;  segment larger
MES1:   .ASCII  /THE MAIN SEGMENT IS RUNNING AND WILL/
        .ASCII  / CALL A/
        LMES1=.-MES1
MES2:   .ASCII  /THE MAIN SEGMENT WILL NOW CALL B/
        LMES2 =.-MES2
MES3:   .ASCII  /THE MAIN SEGMENT WILL NOW CALL A/
        LMES3 =.-MES3
MES4:   .ASCII  /THE MAIN SEGMENT WILL NOW CALL TOTAL/
        LMES4 =.-MES4
MES5:   .ASCII  /THE MAIN SEGMENT WILL NOW EXIT/
        LMES5 =.-MES5
        .PSECT  OTHER   D,GBL,OVR,REL,RW ; PSECT for data
OP1:    .WORD   5                       ; 1st operand
        .WORD   OP                      ; address of operation
                                        ;  in ASCII
OP2:    .WORD   2                       ; 2nd operand
ANS:    .BLKW   1                       ; Answer to operation

        .PSECT                          ; Back to blank PSECT
        .EVEN                           ; Move to word boundary
TOT::   .WORD   0                       ; Total
OP::    .BLKB   1                       ; Operand in ASCII
        .EVEN                           ; Move to word boundary

```

SOLUTION

```
51   START:  QIOW$C  IO.WVB,5,1,,,,<MES1,LMES1,40> ;Write MES1
52           CALL    A                  ; Call subroutine A
53           CALL    RTOTAL             ; Call routine to     ;;EX
54                                      ;  display runnins    ;;EX
55                                      ;  total              ;;EX
56           QIOW$C  IO.WVB,5,1,,,,<MES2,LMES2,40> ;Write MES2
57           CALL    B                  ; Call subroutine B
58   ; Set up for loop
59           MOV     #3,R4              ; Counter
60   LOOP:   QIOW$C  IO.WVB,5,1,,,,<MES3,LMES3,40> ; Write MES3
61           CLR     ANS                ; Clear answer in case
62                                      ;  of no operation
63           CALL    A                  ; Call subroutine A
64           CALL    RTOTAL             ; Call routine to     ;;EX
65                                      ;  display runnins    ;;EX
66                                      ;  total              ;;EX
67           SOB     R4,LOOP            ; Decrement counter and
68                                      ;  loop back until done
69           QIOW$C  IO.WVB,5,1,,,,<MES4,LMES4,40> ; Write MES4
70           CALL    TOTAL              ; Call routine to
71                                      ;  display srand total
72           QIOW$C  IO.WVB,5,1,,,,<MES5,LMES5,40> ; Write MES5
73           EXIT$S                     ; Exit
74           .END START
```

```
 1           PROGRAM MAIN
 2   C
 3   C FILE LEX69A.FTN                  !!EX
 4   C
 5   C Modified to call RTOTAL to display the runnins    !!EX
 6   C after each call to A                              !!EX
 7   C
 8   C This program prints a message and then calls subroutine
 9   C A. Subroutine A asks whether to perform Job 1 or Job 2.P
10   C It then calls either subroutine JOB1 or JOB2 which
11   C performs the operation and displays the results. MAIN
12   C then calls subroutine B which displays a message. MAIN
13   C then calls subroutine A 3 more times, keepins a srand
14   C total of the operations. Finally, it displays the
15   C srand total and exits.
16   C
17   C Task-build instructions: Use LEX69A.ODL as the input!!EX
18   C file for RTOTAL in the root. Use LEX69B.ODL as the  !!EX
19   C input file for RTOTAL in the best overlay segment   !!EX
20   C
```

Overlaying Techniques

SOLUTION

```
21          COMPLEX DUMMY(1024)      ! Leave space to make
22    C                              !  segment larger
23          COMMON /OTHER/OP1,OP,OP2,ANS
24          INTEGER OP1,OP,OP2,ANS
25          DATA OP1,OP2/5,2/
26    C
27          COMMON /TOTCOM/TOT
28          INTEGER TOT              ! Total
29    C
30          TYPE *,'THE MAIN SEGMENT IS RUNNING AND WILL
31          1CALL A'
32          CALL A                   ! Call subroutine A
33          CALL RTOTAL              ! Call subroutine  !!EX
34    C                              !  RTOTAL to display!!EX
35    C                              !  running total   !!EX
36          TYPE *,'THE MAIN SEGMENT WILL NOW CALL B'
37          CALL B                   ! Call subroutine B
38          DO 10, I=1,3
39          TYPE *,'THE MAIN SEGMENT WILL NOW CALL A'
40          ANS = 0                  ! Clear answer in case
41    C                              !  of no operation
42    10    CALL A                   ! Call subroutine A
43          CALL RTOTAL              ! Call subroutine  !!EX
44    C                              !  RTOTAL to display!!EX
45    C                              !  the running total!!EX
46          TYPE *,'THE MAIN SEGMENT WILL CALL TOTAL'
47          CALL TOTAL(TOT)          ! Call routine to
48    C                              !  display grand total
49          TYPE *,'THE MAIN SEGMENT WILL NOW EXIT'
50          CALL EXIT                ! EXIT
51          END
```

Overlaying Techniques

SOLUTION

```
 1            .TITLE  RTOTAL
 2            .IDENT  /01/
 3            .ENABL  LC                  ; Enable lower case
 4    ;
 5    ; FILE LEX69B.MAC
 6    ;
 7    ; Subroutine to print the running total
 8    ;
 9            .MCALL  QIOW$S              ; External system macros
10            .NLIST  BEX                 ; Do not list binary
11                                        ;  extensions
12    RTOFMT: .ASCIZ  /THE TOTAL SO FAR IS %D./ ;Format string
13    RTOTBF: .BLKB   100.                ; Output buffer
14            .EVEN
15            .NLIST  BEX                 ; List binary extensions
16
17    RTOTAL::MOV     #RTOTBF,R0          ; Set up for $EDMSG
18            MOV     #RTOFMT,R1          ;
19            MOV     #TOT,R2             ;
20            CALL    $EDMSG              ; Edit message
21            QIOW$S  #IO.WVB,#5,#1,,,,<#RTOTBF,R1,#40>
22                                        ; Print it
23            RETURN
24            .END
```

```
 1            SUBROUTINE RTOTAL
 2    C
 3    C FILE LEX69B.FTN
 4    C
 5    C Subroutine to print the running total
 6    C
 7            COMMON /TOTCOM/TOT
 8            INTEGER TOT
 9            TYPE 5,TOT
10    5       FORMAT(' THE TOTAL SO FAR IS', I4,'.')
11            RETURN
12            END
```

```
; Module 6, Lab Exercise 9
;
; .ODL file in MACRO-11, placing RTOTAL in the root
; segment for testing
; All overlays are memory-resident
        .ROOT   LEX69A-LEX69B-*!(A-!(JOB1,JOBXX),B,TOTAL)
; LEX69A = MAIN modified to call RTOTAL
; LEX69B = RTOTAL
        .END
```

SOLUTION

```
; Module 6, Lab Exercise 9
;
; .ODL file in FORTRAN, placing RTOTAL in the root
; segment for testing
; All overlays are memory-resident
        .ROOT   LEX69A-LEX69B-FLIB-*!(OVRA,OVRB,TOTAL-FLIB)
OVRA:   .FCTR   A-FLIB-!(JOB1-FLIB,JOBXX-FLIB)
OVRB:   .FCTR   B-FLIB
FLIB:   .FCTR   LB:[1,1]F4POTS/LB
;
; LEX69A = MAIN modified to call RTOTAL
; LEX69B = RTOTAL
;
        .END
```

```
; Module 6, Lab Exercise 9
;
; .ODL file in MACRO-11, placing RTOTAL in the best
; overlay segment
; All overlays are memory-resident
        .ROOT   LEX69A-*!(A-LEX69B-!(JOB1,JOBXX),B,TOTAL)
; LEX69A = MAIN modified to call RTOTAL
; LEX69B = RTOTAL
        .END
```

```
; Module 6, Lab Exercise 9
;
; .ODL file in FORTRAN, placing RTOTAL in the best
; overlay segment
; All overlays are memory-resident
        .ROOT   LEX69A-FLIB-*!(OVRA,OVRB,OVRC)
OVRA:   .FCTR   A-LEX69B-FLIB-!(JOB1-FLIB,JOBXX-FLIB)
OVRB:   .FCTR   B-FLIB
OVRC:   .FCTR   TOTAL-FLIB
FLIB:   .FCTR   LB:[1,1]F4POTS/LB
;
; LEX69A = MAIN modified to call RTOTAL
; LEX69B = RTOTAL
;
        .END
```

Static Regions

TEST/EXERCISE

1. Create an initialized resident common (size: 32(1Ø) blocks = 1Ø24(1Ø) words, contents: 25(1Ø) in each word). Check with your course administrator to find out where to place the common type partition. Write two tasks, one that modifies all values in the common, and one that reads the values and displays them.

2. Create a resident library using the supplied FORTRAN callable subroutines AADD, SUBB, MULL and DIVV (all in LIB.MAC). Write a task that calls one or more of the routines. For example, write a task that asks for four numbers (A, B, C, and D) and then computes and displays (A * B) + (C * D) = answer.

Static Regions

SOLUTION

1.

```
        .TITLE LEX71A
        .IDENT  /01/
        .ENABL  LC              ; Enable lower case
;+
; File LEX71A.MAC
;
; Program which creates and initializes a common region
; which will be referenced using overlaid Psects.
;
; Size 1024. words, contents all 25's
;
; Task-build instructions: Must include /SHAREABLE:COMMON
; and /NOHEADER switches; STACK=0 and PAR=COMWP options.
; Must create .STB file. May be /CODE:PIC or absolute
; (default).
;
; The code is placed in a Psect named MYDATA
;-
        .PSECT  MYDATA D,GBL,OVR ; Defaults REL,RW
        .REPT   1024.           ; Repeat count
        .WORD   25.             ; Word of 25(10)
        .ENDR                   ; End repeat range
        .END
```

```
        BLOCK DATA LEX71A
C
C File LEX71A.FTN
C
C Program to create and initialize a resident common
C
C Size is 1024 words, initialized with all 25's
C
C Task-build instructions: Must include /SHAREABLE:COMMON
C and /NOHEADER switches; STACK=0 and PAR=COMWP options.
C Must create .STB file. May be /CODE:PIC or absolute
C (the default). OTS library NOT required.
C
        COMMON /MYDATA/ I(1024)
        DATA I /1024*25/
        END
```

Static Regions

SOLUTION

```
1           .TITLE  LEX71B
2           .IDENT /01/
3           .ENABL  LC                  ; Enable lower case
4   ;+
5   ; FILE LEX71B.MAC
6   ;
7   ; This task decremtents the values in the static common
8   ; region LEX71A. It uses the technique of overlaid Psects
9   ; to reference the region.
10  ;
11  ; Task-build instructions:
12  ;
13  ;       >LINK/MAP/OPTION LEX71B
14  ;       Option? RESCOM=LEX71A/RW
15  ;       Option? <RET>
16  ;-
17          .MCALL  QIOW$S,EXIT$S       ; System macros
18          .PSECT  MYDATA D,GBL,OVR    ; Psect used in COMWP
19  M=.                                 ; local symbol for start
20                                      ;  of region
21          .PSECT                      ; Back to blank Psect
22  IOSB:   .BLKW   2                   ; I/O status block
23  ARG:    .BLKW   1                   ; Argument block for
24                                      ;  error code
25  BUFF:   .BLKB   100.                ; Output buffer
26  FERR1:  .ASCIZ  /DIR ERROR ON QIO. CODE = %D/ ; Directive
27                                      ;  error message
28  FERR2:  .ASCIZ  !I/O ERROR ON QIO. CODE = %D! ; I/O error
29                                      ;  message
30  DONE:   .ASCII  /LEX71B HAS MODIFIED THE VALUES/ ; Done
31          .ASCII  / IN THE COMMON LEX71A/          ;  message
32  LDONE   =.-DONE
33  W       =1024.                      ; Word count in region
34          .EVEN
35  ;
36  START:  MOV     #M,R2               ; Starting addr of data
37                                      ;  in the region
38          MOV     #W,R5               ; Loop count
39  LOOP:   DEC     (R2)+               ; Decrement value
40          SOB     R5,LOOP             ; Loop back if not done
41          QIOW$S  #IO.WVB,#5,#1,,#IOSB,,<#DONE,#LDONE,#40>
42          BCS     ERROR               ; Check for dir error
43          TSTB    IOSB                ; Check for I/O error
44          BLT     ERROR1              ; Branch on I/O error
45          EXIT$S                      ; Exit
46  ; Error code
47  ERROR:  MOV     $DSW,ARG            ; Move DSW to arg block
48          MOV     #FERR1,R1           ; Addr of format string
49          BR      SETUP               ; Branch to $EDMSG code
```

SOLUTION

```
50   ERROR1: MOVB    IOSB,R0          ; Extend sign on I/O
51           MOV     R0,ARG           ;  status and place in
52                                    ;  arg block
53           MOV     #FERR2,R1        ; Addr of format strings
54   SETUP:  MOV     #BUFF,R0         ; Addr of output buffer
55           MOV     #ARG,R2          ; Addr of argument block
56           CALL    $EDMSG           ; Edit message
57           QIOW$S  #IO.WVB,#5,#1,,,,<#BUFF,R1,#40> ; Write
58                                    ;  message
59           EXIT$S                   ; Exit
60           .END    START
```

```
 1           PROGRAM LEX71B
 2   C
 3   C File LEX71B.FTN
 4   C
 5   C Task to decrement each word in the static common
 6   C region LEX71A. It uses a COMMON to reference
 7   C the data.
 8   C
 9   C Task-build instructions:
10   C
11   C       LINK/MAP/OPTION LEX71B,LB:[1,1]FOROTS/LIBRARY
12   C       Option? RESCOM=LEX71A/RW
13   C       Option? <RET>
14   C
15           COMMON /MYDATA/ L(1024)! Common to reference
16   C                              !        shared region
17   C Decrement values
18           DO 5 K=1,1024
19           L(K)=L(K)-1
20   5       CONTINUE
21           WRITE (5,10)           ! Display done message
22   10      FORMAT (' LEX71B HAS MODIFIED THE VALUES IN THE
23           1 COMMON LEX71A')
24           CALL EXIT
25           END
```

Static Regions

SOLUTION

```
1           .TITLE  LEX71C
2           .IDENT /01/
3           .ENABL  LC                  ; Enable lower case
4   ;+
5   ; FILE LEX71C.MAC
6   ;
7   ; This task sets the values from the static common
8   ; region LEX71A. It uses the technique of overlaid Psects
9   ; to reference the region.
10  ;
11  ; Task-build instructions:
12  ;
13  ;       >LINK/MAP/OPTION LEX71C
14  ;       Option? RESCOM=LEX71A/RO
15  ;       Option? <RET>
16  ;-
17          .MCALL  QIOW$S,EXIT$S       ; System macros
18          .PSECT  MYDATA D,GBL,OVR ; Psect used in COMWP
19  M=.                                 ; local symbol for start
20                                      ;  of region
21          .PSECT                      ; Back to blank Psect
22  IOSB:   .BLKW   2                   ; I/O status block
23  ARG:    .BLKW   1                   ; Argument block for
24                                      ;  error code
25  BUFF:   .BLKB   100.                ; Output buffer
26  FMT:    .ASCIZ  /%8D/               ; Format string for
27                                      ;  output of data
28  FERR1:  .ASCIZ  /DIR ERROR ON QIO. DSW = %D/ ; Directive
29                                      ;  error message
30  FERR2:  .ASCIZ  !I/O ERROR ON QIO. CODE = %D! ; I/O error
31                                      ;       message
32
33          N=128.                      ; Loop count = 128. lines,
34                                      ;  8 #s per line
35          .EVEN
36  ;
37  START:  MOV     #M,R2               ; Starting addr of data
38                                      ;  in the region
39          MOV     #N,R5               ; Loop count
40  LOOP:   MOV     #BUFF,R0            ; Output buffer
41          MOV     #FMT,R1             ; Format string
42          CALL    $EDMSG              ; Edit message
43          QIOW$S  #IO.WVB,#5,#1,,#IOSB,,<#BUFF,R1,#40>
44          BCS     ERROR               ; Check for dir error
45          TSTB    IOSB                ; Check for I/O error
46          BLT     ERROR1              ; Branch on I/O error
47  ; Stay here for good write
48          SOB     R5,LOOP             ; Decrement counter, loop
49                                      ;  back if not yet done
50          EXIT$S                      ; Exit
```

SOLUTION

```
51    ; Error code
52    ERROR:  MOV     $DSW,ARG          ; Move DSW to arg block
53            MOV     #ERR1,R1          ; Addr of format string
54            BR      SETUP             ; Branch to $EDMSG code
55    ERROR1: MOVB    IOSB,R0           ; Extend sign on I/O
56            MOV     R0,ARG            ;  status and place in
57                                      ;  arg block
58            MOV     #ERR2,R1          ; Addr of format string
59    SETUP:  MOV     #BUFF,R0          ; Addr of output buffer
60            MOV     #ARG,R2           ; Addr of argument block
61            CALL    $EDMSG            ; Edit message
62            QIOW$S  #IO.WVB,#5,#1,,,,<#BUFF,R1,#40> ; Write
63                                      ;  message
64            EXIT$S                    ; Exit
65            .END    START
```

```
 1            PROGRAM LEX71C
 2    C
 3    C File LEX71C.FTN
 4    C
 5    C Task to read data from the static common region LEX71A
 6    C and print it out at TI:. It uses a COMMON to reference
 7    C the data.
 8    C
 9    C Task-build instructions:
10    C
11    C       LINK/MAP/OPTION LEX71C,LB:[1,1]FOROTS/LIBRARY
12    C       Option? RESCOM=LEX71A/RO
13    C       Option? <RET>
14    C
15            COMMON /MYDATA/ L(1024)! Common to reference
16    C                               !        shared region
17    C Loop through to display region, 8 numbers on a line
18            DO 50 J = 1,1024,8
19            WRITE (5,10) (L(K),K=J,J+7) ! Write values
20    10      FORMAT (' ',I2,7I8)
21    50      CONTINUE
22            CALL EXIT
23            END
```

SOLUTION

2.

```
 1          .TITLE  LEX72
 2          .IDENT  /01/
 3          .ENABL  LC                  ; Enable lower case
 4  ;+
 5  ; File LEX72.MAC
 6  ;
 7  ; Solution to Module 7, Lab Exercise 2
 8  ;
 9  ; Task computes sum of products using resident library
10  ; routines.
11  ;
12  ; Assembly and task build instructions:
13  ;
14  ;       MACRO/LIST LB:[1,1]PROGMACS/LIB,dev:[ufd]LEX72
15  ;       LINK/MAP/OPTIONS LEX72,LB:[1,1]PROGSUBS/LIB
16  ;       Option? RESLIB=LIB/RO
17  ;-
18          .MCALL  QIOW$,QIOW$S,QIOW$C,DIR$,EXIT$S
19          .MCALL  DIRERR,IOERR
20          .GLOBL  $CDTB,$EDMSG        ; Routines in SYSLIB
21          .GLOBL  MULL,AADD           ; Routines in library LIB
22
23  ; Messages
24  HDRMS:  .ASCII  /TASK WILL COMPUTE (A*B)+(C*D)/<15><12>
25          .ASCII  /ENTER NUMBERS IN DECIMAL./
26  HDRML = .-HDRMS
27  APRMT:  .ASCII  /ENTER A: /
28  PLEN = .-APRMT                      ; Length of prompt
29                                      ;  (assumed to be all the
30                                      ;  same length)
31  BPRMT:  .ASCII  /ENTER B: /
32  CPRMT:  .ASCII  /ENTER C: /
33  DPRMT:  .ASCII  /ENTER D: /
34
35  ; ASCII buffers
36  ASCA:   .BLKB   7                   ; ASCII for A's value
37  ASCB:   .BLKB   7                   ; Same for B
38  ASCC:   .BLKB   7                   ; C
39  ASCD:   .BLKB   7                   ; D
40  OUTBUF: .BLKB   80.
41
42  ; $EDMSG format string
43  EDMFMT: .ASCIZ  /%N(%VA * %VA) + (%VA * %VA) = %D/
44          .EVEN
45
46  ; FORTRAN-compatible argument blocks:
47  MULARG: .WORD   3                   ; For MUL
48          .WORD   M1
49          .WORD   M2
50          .WORD   MULRES
```

SOLUTION

```
51    ADDARG: .WORD   3                   ; For ADD
52            .WORD   MURES1              ; First MUL result
53            .WORD   MULRES              ; Second result
54            .WORD   GRTOT               ; Grand total
55
56    ; ASCII buffer table.  Initially each entry in this table
57    ; consists of the address of a prompt string followed by
58    ; the address of the buffer to store the input.  After a
59    ; string is input, however, the prompt string address is
60    ; replaced by the length of the input string.  This
61    ; table, with the addition of the final value GRTOT, then
62    ; serves as the $EDMSG argument block.
63    EDMARG:
64    ABTBL:  .WORD   APRMT,ASCA
65            .WORD   BPRMT,ASCB
66    CDTBL:  .WORD   CPRMT,ASCC
67            .WORD   DPRMT,ASCD
68    GRTOT:  .WORD                       ; Grand total (numeric
69                                        ;  value is inserted
70                                        ;  directly into $EDMSG
71                                        ;  block)
72    ;
73    ; Other numeric values
74    M1:     .WORD                       ; First MUL argument
75    M2:     .WORD                       ; Second MUL argument
76    MURES1: .WORD                       ; First MUL result
77    MULRES: .WORD                       ; MUL result
78
79    RDPRMT: QIOW$   IO.RPR,5,1,,IOSB,,<,7,,,PLEN,'$>
80    IOSB:   .BLKW   2
81
82    ;
83    ; Code
84    ;
85    START:  QIOW$C  IO.WVB,5,1,,,,<HDRMS,HDRML,40> ; Identify
86            MOV     #M1,R5              ; R5 => location to store
87                                        ;  binary input values
88            MOV     #RDPRMT,R4          ; R4 => "read with
89                                        ;  prompt" DPB
90            MOV     #ABTBL,R3           ; R3 => ASCII buffer table
91            CALL    GETINP              ; Get A
92            CALL    GETINP              ; Get B
93            MOV     #MULARG,R5          ; R5 => MUL arg block
94            CALL    MULL                ; Do first multiply
95            MOV     MULRES,MURES1       ; Save result
96            MOV     #M1,R5              ; Reset registers
97            MOV     #RDPRMT,R4          ;  (FORTRAN calling
98            MOV     #CDTBL,R3           ;  convention does not
99                                        ;  guarantee they are
100                                       ;  preserved.)
```

SOLUTION

```
101             CALL    GETINP          ; Get C
102             CALL    GETINP          ; Get D
103             MOV     #MULARG,R5
104             CALL    MULL            ; Do second multiply
105             MOV     #ADDARG,R5
106             CALL    AADD            ; Add multiplication
107                                     ;  results
108             MOV     #OUTBUF,R0      ; Prepare
109             MOV     #EDMFMT,R1      ;  for
110             MOV     #EDMARG,R2      ;  $EDMSG
111             CALL    $EDMSG
112             QIOW$S  #IO.WVB,#5,#1,,,,<#OUTBUF,R1,#40>
113             BCS     IODER
114             EXIT$S
115
116     ; Subroutine GETINP to get input values.
117     ;
118     ; Input:        R5 =>   location to store binary result
119     ;               R4 =>   QIO DPB
120     ;               R3 =>   Address of prompt string,
121     ;                       followed by address to store
122     ;                       ASCII input
123     ;
124     ; Output:       R5 =    input value +2
125     ;               R4      Unchanged
126     ;               R3 =    input value +4. Location formerly
127     ;                       containing address of prompt now
128     ;                       contains length of input
129     ;
130     GETINP: MOV     (R3)+,Q.IOPL+6(R4)  ; Load prompt address
131             MOV     (R3)+,R0        ; R0 => input buffer
132             MOV     R0,Q.IOPL(R4)   ; Copy to QIO DPB
133             DIR$    R4              ; Get input
134             BCS     IODER           ; Directive error
135             CMPB    IOSB,#IS.SUC    ; I/O successful?
136             BNE     IOIOER          ; No
137             MOV     IOSB+2,-4(R3)   ; Save input length
138             CALL    $CDTB           ; Convert to binary
139             MOV     R1,(R5)+        ; Store binary
140             RETURN
141
142     ; Error messages:
143     IODER:  DIRERR  <ERROR ON QIOW$>
144     IOIOER: IOERR   #IOSB,<ERROR ON QIOW$>
145             .END    START
```

Static Regions

SOLUTION

```
1           PROGRAM LEX72
2     C+
3     C File LEX72.FTN
4     C
5     C Solution to Module 7, Lab Exercise 2
6     C
7     C Task computes sum of products using resident library
8     C routines.
9     C
10    C Task build instructions:
11    C
12    C       LINK/MAP/OPTIONS LEX72,LB:[1,1]F4POTS/LIB
13    C       Option? RESLIB=LIB/RO
14    C-
15          INTEGER A,B,C,D,MURES1,MURES2,GRTOT
16    C ASCII bytes to make prompting code cleaner
17          BYTE ASCA,ASCB,ASCC,ASCD
18          DATA ASCA,ASCB,ASCC,ASCD/'A','B','C','D'/
19    C
20          TYPE 5
21    5     FORMAT (' TASK WILL COMPUTE (A*B)+(C*D)'/
22          1 ' ENTER NUMBERS IN DECIMAL.')
23    C FORMAT statements used repeatedly below:
24    15    FORMAT ('$ENTER ',A1,': ')
25    25    FORMAT (I6)
26          TYPE 15,ASCA              ! Prompt for
27          ACCEPT 25,A               !  and input A
28          TYPE 15,ASCB              ! Prompt for
29          ACCEPT 25,B               !  and input B
30          CALL MULL(A,B,MURES1)     ! MURES1 = A*B
31          TYPE 15,ASCC              ! Prompt for
32          ACCEPT 25,C               !  and input C
33          TYPE 15,ASCD              ! Prompt for
34          ACCEPT 25,D               !  and input D
35          CALL MULL(C,D,MURES2)     ! MURES2 = C*D
36          CALL AADD(MURES1,MURES2,GRTOT)  ! GRTOT = sum
37          TYPE 35, A,B,C,D,GRTOT
38    35    FORMAT (' (',I6,' * ',I6,') + (',I6,' * ',I6,') = ',I6)
39          CALL EXIT
40          END
```

Dynamic Regions

TEST/EXERCISE

1. Referring to Exercise 1 of Module 7 (Static Regions), modify the tasks that reference the common so that they both map to the common dynamically using the memory management directives.

2. Write a task that creates a dynamic region two blocks long, fills it with a character typed in at the terminal, and leaves it in existence on exit. Write a second task that modifies one value in the region, then displays all the values in the region at the terminal, and finally deletes the region.

3. Modify SNDREF so that it sends the region by reference to a second receiver task, in addition to RCVREF. Write the second receiver task, which should modify values in the region and then display the values in the region at the terminal.

Dynamic Regions

SOLUTION

1.

```
          .TITLE  LEX81B
          .IDENT  /01/
          .ENABL  LC                ; Enable lower case
;+
; File LEX81B.MAC
;
; LEX71B modified to use memory management directives
;
; Program to attach to the existing region LEX71A, create
; a virtual address window (mapped on creation), decrement
; all values in the region by 1, detach from the region
; and exit.
;
; Assemble and task-build instructions:
;
;       >MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[ufd]LEX81B
;       >LINK/MAP/OPTION LEX81B,LB:[1,1]PROGSUBS/LIBRARY
;       >Option? WNDWS=1
;       >Option? <RET>
;-
          .MCALL  EXIT$S,RDBBK$,WDBBK$,ATRG$C ; System
          .MCALL  CRAW$,DTRG$S,DIR$,QIOW$S    ;  macros
          .MCALL  DIRERR,IOERR      ; Supplied macros
RDB:      RDBBK$  32.,LEX71A,LEX71A,<RS.WRT!RS.RED>
;         Define region with:
;                 Size              = 32. (32. word blocks)
;                 Name              = LEX71A
;                 Partition         = LEX71A
;                 Attach with read and write access
;
WIN:      CRAW$   WDB       ;DPB for create address window
WDB:      WDBBK$  7,32.,0,0,32.,<WS.MAP!WS.RED!WS.WRT>
;         Define window with:
;                 APR                = 7
;                 Size               = 32. (32. word blocks)
;                 Offset in region = 0 (32. word blocks)
;                 Length in region = 32. (32. word blocks)
;                 Map on create with read and write access
;
IOSB:     .BLKW   2                 ; I/O status block
W         =1024.                    ; # of words in region
DONE:     .ASCII  /LEX81B HAS MODIFIED THE VALUES/ ; Done
          .ASCII  / IN LEX71A/                    ;  message
LDONE     =.-DONE
START:    ATRG$C  RDB               ; Attach to region
          BCS     ERR1              ; Check for error
          MOV     RDB+R.GID,WDB+W.NRID ; Move region ID
                                       ; into WDB
          DIR$    #WIN              ; Create window
          BCS     ERR2              ; Check for error
```

SOLUTION

```
51          MOV     #160000,R2          ; Set base addr in region
52          MOV     #W,R5               ; Get word count
53  LOOP:   DEC     (R2)+               ; Decrement value
54          SOB     R5,LOOP             ; Loop until done
55          QIOW$S  #IO.WVB,#5,#1,,#IOSB,,<#DONE,#LDONE,#40>
56                                      ; Write done message
57          BCS     ERR3D               ; Check for dir error
58          TSTB    IOSB                ; Check for I/O error
59          BLT     ERR3I               ; Branch on error
60          DTRG$S  #RDB                ; Detach from region
61          BCS     ERR4                ; Check for error
62          EXIT$S
63  ; Error handling code
64  ERR1:   DIRERR  <ERROR ATTACHING TO REGION>
65  ERR2:   DIRERR  <ERROR CREATING WINDOW AND MAPPING>
66  ERR3D:  DIRERR  <ERROR WRITING DONE MESSAGE>
67  ERR3I:  IOERR   #IOSB,<ERROR WRITING DONE MESSAGE>
68  ERR4:   DIRERR  <ERROR DETACHING FROM REGION>
69          .END    START
```

```
 1          PROGRAM LEX81B
 2  C
 3  C File LEX81B.FTN
 4  C
 5  C LEX71B modified to use memory management directives
 6  C
 7  C Program to attach region LEX71A in partition LEX71A
 8  C create a window and map it to the region upon creation,
 9  C decrement each value in the region by 1, and detach
10  C from it
11  C
12  C Task-build with these options:
13  C               VSECT=DATA:160000:20000
14  C               WNDWS=1
15  C
16          INTEGER RDB(8),WDB(8)
17  C This common block will align with the address window
18          COMMON /DATA/IDATA(1024)
19  C RDB = Region definition block with the following
20  C properties:
21  C       Size            32 (10) (32.-word blocks)
22  C       Name            LEX71A
23  C       Partition       LEX71A
24  C       Protection      WO:none,SY:RWED,OW:RWED,GR:RWED
25  C       Attach with read and write access
26  C Initialize the RDB
27          DATA RDB /0,32,3RLEX,3R71A,3RLEX,3R71A,"3,
28          1"170000/
```

Dynamic Regions

SOLUTION

```
29    C WDB = Window definition block with the following properties:
30    C       APR                 7
31    C       Size                32 (10) (32.-word blocks)
32    C       Offset in region    0 (32.-word blocks)
33    C       Length of window    32 (10) (32.-word blocks)
34    C       Map on create with read and write access
35    C Initialize the WDB
36            DATA WDB /"3400,0,32,0,0,32,"203,0/
37    C
38    C Attach region
39            CALL ATRG (RDB,IDS)
40    C Check for error on attach
41            IF (IDS .LT. 0) GOTO 100
42    C Move region id to WDB
43            WDB(4)=RDB(1)
44    C Create and map window
45            CALL CRAW (WDB,IDS)
46    C Check for error
47            IF (IDS .LT. 0) GOTO 200
48    C Decrement values
49            DO 50 K=1,1024
50            IDATA(K)=IDATA(K)-1
51    50      CONTINUE
52    C Detach from region and delete it
53            CALL DTRG (RDB,IDS)
54    C Check for error
55            IF (IDS .LT. 0) GOTO 300
56    C And Jump to exit
57            WRITE (5,60)
58    60      FORMAT (' LEX81B HAS MODIFIED THE VALUES IN
59            1 THE COMMON LEX71A')
60            GOTO 500
61    C
62    C       Error messages
63    100     WRITE (5,101) IDS
64    101     FORMAT (' ERROR ATTACHING TO REGION, DSW =',I4)
65            GOTO 500
66    200     WRITE (5,201) IDS
67    201     FORMAT (' ERROR IN CREATING WINDOW, DSW =',I4)
68            GOTO 500
69    300     WRITE (5,301) IDS
70    301     FORMAT (' ERROR DETACHING FROM REGION, DSW =',I4)
71    C
72    500     CALL EXIT
73            END
```

Dynamic Regions

SOLUTION

```
1           .TITLE  LEX81C
2           .IDENT  /01/
3           .ENABL  LC                  ; Enable lower case
4   ;+
5   ; File LEX81C.MAC
6   ;
7   ; LEX71C modified to use memory management directives
8   ;
9   ; Program to attach to an existing region, create a
10  ; virtual address window (mapped on creation), read
11  ; ASCII data from the region, detach from the region
12  ; and exit.
13  ;
14  ; Assemble and task-build instructions:
15  ;
16  ;       >MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[ufd]LEX81C
17  ;       >LINK/MAP/OPTION LEX81C,LB:[1,1]PROGSUBS/LIBRARY
18  ;       >Option? WNDWS=1
19  ;       >Option? <RET>
20  ;-
21          .MCALL  EXIT$S,RDBBK$,WDBBK$,ATRG$C ; System
22          .MCALL  CRAW$,DTRG$S,DIR$,QIOW$S    ;  macros
23          .MCALL  DIRERR,IOERR    ; Supplied macros
24  RDB:    RDBBK$  32.,LEX71A,LEX71A,RS.RED
25  ;       Define region with:
26  ;               Size            = 32. (32. word blocks)
27  ;               Name            = LEX71A
28  ;               Partition       = LEX71A
29  ;               Attach with read access
30  ;
31  WIN:    CRAW$   WDB     ;DPB for create address window
32  WDB:    WDBBK$  7,32.,0,0,32.,,<WS.MAP!WS.RED>
33  ;       Define window with:
34  ;               APR              = 7
35  ;               Size             = 32. (32. word blocks)
36  ;               Offset in region = 0 (32. word blocks)
37  ;               Length in region = 32. (32. word blocks)
38  ;               Map on create with read access
39  ;
40  IOSB:   .BLKW   2               ; I/O status block
41  ARG:    .BLKW   1               ; Argument block for
42                                  ;  error code
43  BUFF:   .BLKB   100.            ; Output buffer
44  FMT:    .ASCIZ  /%8D/           ; Format string
45          N=128.                  ; Loop count
46          .EVEN
47  START:  ATRG$C  RDB             ; Attach to region
48          BCS     ERR1            ; Check for error
49          MOV     RDB+R.GID,WDB+W.NRID ; Move region ID
50                                       ; into WDB
51          DIR$    #WIN            ; Create window
```

SOLUTION

```
52              BCS     ERR2                    ; Check for error
53              MOV     #160000,R2              ; Set base addr in region
54              MOV     #N,R5                   ; Loop count
55      LOOP:   MOV     #BUFF,R0                ; Set up for $EDMSG
56              MOV     #FMT,R1
57              CALL    $EDMSG                  ; Edit data
58              QIOW$S  #IO.WVB,#5,#1,,#IOSB,,<#BUFF,R1,#40>
59                                              ; Write data
60              BCS     ERR3D                   ; Check for dir error
61              TSTB    IOSB                    ; Check for I/O error
62              BLT     ERR3I                   ; Branch on error
63              SOB     R5,LOOP                 ; Print the line
64      DONE:   DTRG$S  #RDB                    ; Detach from region
65              BCS     ERR4                    ; Check for error
66              EXIT$S
67      ; Error handling code
68      ERR1:   DIRERR  <ERROR ATTACHING TO REGION>
69      ERR2:   DIRERR  <ERROR CREATING WINDOW AND MAPPING>
70      ERR3D:  DIRERR  <ERROR WRITING DATA>
71      ERR3I:  IOERR   #IOSB,<ERROR WRITING DATA>
72      ERR4:   DIRERR  <ERROR DETACHING FROM REGION>
73              .END    START
```

```
 1              PROGRAM LEX81C
 2      C
 3      C File LEX81C.FTN
 4      C
 5      C LEX71C modified to use memory management directives
 6      C
 7      C Program to attach region LEX71A in partition LEX71A
 8      C create a window and map it to the region upon creation,
 9      C read data out of the region, and detach from it
10      C
11      C Task-build with these options:
12      C                   VSECT=DATA:160000:20000
13      C                   WNDWS=1
14      C
15              INTEGER RDB(8),WDB(8)
16      C This common block will align with the address window
17              COMMON /DATA/IDATA(1024)
18      C RDB = Region definition block with the following
19      C properties:
20      C       Size                32 (10) (32.-word blocks)
21      C       Name                LEX71A
22      C       Partition           LEX71A
23      C       Protection          WO:none,SY:RWED,OW:RWED,GR:RWED
24      C       Attach with read access
25      C Initialize the RDB
26              DATA RDB /0,32,3RLEX,3R71A,3RLEX,3R71A,"000001,
27              1"170000/
```

Dynamic Regions

SOLUTION

```
28    C WDB = Window definition block with the following properties:
29    C       APR                7
30    C       Size               32 (10) (32.-word blocks)
31    C       Offset in region   0 (32.-word blocks)
32    C       Length of window   32 (10) (32.-word blocks)
33    C       Map on create with read access
34    C Initialize the WDB
35            DATA WDB /"3400,0,32,0,0,32,"201,0/
36    C
37    C Attach region
38            CALL ATRG (RDB,IDS)
39    C Check for error on attach
40            IF (IDS .LT. 0) GOTO 100
41    C Move region id to WDB
42            WDB(4)=RDB(1)
43    C Create and map window
44            CALL CRAW (WDB,IDS)
45    C Check for error
46            IF (IDS .LT. 0) GOTO 200
47    C Print contents of region
48            DO 50 J=1,1024,8
49            WRITE (5,11) (IDATA(K),K=J,J+7)
50    11      FORMAT (' ',I2,7I8)
51    50      CONTINUE
52    C Detach from region and delete it
53            CALL DTRG (RDB,IDS)
54    C Check for error
55            IF (IDS .LT. 0) GOTO 300
56    C And Jump to exit
57            GOTO 500
58
59    C       Error messages
60    100     WRITE (5,101) IDS
61    101     FORMAT (' ERROR ATTACHING TO REGION, DSW =',I4)
62            GOTO 500
63    200     WRITE (5,201) IDS
64    201     FORMAT (' ERROR IN CREATING WINDOW, DSW =',I4)
65            GOTO 500
66    300     WRITE (5,301) IDS
67    301     FORMAT (' ERROR DETACHING FROM REGION, DSW =',I4)
68    C
69    500     CALL EXIT
70            END
```

Dynamic Regions

SOLUTION

2.

```
        .TITLE  LEX82A
        .IDENT  /01/
        .ENABL  LC              ; Enable lower case
;
; File LEX82A.MAC
;
; Program to create an named region (attached on
; creation), create a virtual address window (mapped on
; creation), place ASCII data in to region, detach from
; the region and exit, leaving the region in existence.
;
; Task-build instructions:
;
;       Include WNDWS=1 option
;
        .MCALL  EXIT$S,RDBBK$,WDBBK$,CRRG$,CRAW$
        .MCALL  DTRG$,DIR$,QIOW$S,QIOW$C

REG:    CRRG$   RDB     ;DPB for create region
;       Define region with:
;               Size            = 2 (32. word blocks)
;               Name            = MYREG
;               Partition       = GEN
;               Protection      = WO:None,SY:RWED,
;                                 OW:RWED,GR:RWED
;               Do not mark for delete on last detach
;               Attach with write and delete access
RDB:    RDBBK$  2,MYREG,GEN,<RS.NDL!RS.DEL!RS.WRT!RS.ATT>,170000
;
WIN:    CRAW$   WDB     ; DPB for create address window
;       Define window with:
;               APR             = 7
;               Size            = 2 (32. word blocks)
;               Offset in region = 0 (32. word blocks)
;               Length in region = 2 (32. word blocks)
;               Map on create with write access
WDB:    WDBBK$  7,2,0,0,2,<WS.MAP!WS.WRT>
;
DET:    DTRG$   RDB     ; DPB for detaching region
IOSB:   .BLKW   2       ; I/O status block
BUFF:   .BLKB   80.     ; Input/Output buffer
MES:    .ASCII  /ENTER ASCII CHARACTER: /
LEN     =       .-MES
DNMES:  .ASCII  <15>/LEX82A HAS CREATED AND INITIALIZED/
        .ASCII  / THE REGION/
LDNMES  =.-DNMES
; Error format strings
FCRRER: .ASCIZ  /ERROR CREATING REGION. DSW = %D./
FCRWER: .ASCIZ  /ERROR CREATING WINDOW. DSW = %D./
FDETER: .ASCIZ  /ERROR DETACHING FROM REGION. DSW = %D./
```

SOLUTION

```
51    FQI1DE: .ASCII  /DIRECTIVE ERROR ON READ AFTER PROMPT/
52            .ASCIZ  / QIO. DSW = %D./
53    FQI1IE: .ASCII  !I/O ERROR ON READ AFTER PROMPT QIO.!
54            .ASCIZ  / CODE = %D./
55    FQI2DE: .ASCIZ  /DIRECTIVE ERROR ON WRITE QIO. DSW = %D./
56    FQI2IE: .ASCIZ  !I/O ERROR ON WRITE QIO. CODE = %D.!
57
58            .EVEN
59
60    START:  DIR$    #REG    ; Create region
61            BCS     ERR1    ; Check for error
62            MOV     RDB+R.GID,WDB+W.NRID ; Move region ID
63                                         ;  into WDB
64            DIR$    #WIN    ; Create window
65            BCS     ERR2    ; Check for error
66            MOV     #160000,R5 ; Set base address in region
67            QIOW$S  #IO.RPR,#5,#1,,#IOSB,,<#BUFF,#1,,#MES,#LEN,#'$>
68                            ; Prompt and read data
69            BCS     ERR3D   ; Check for directive error
70            TSTB    IOSB    ; Check for I/O error
71            BLT     ERR3I   ; Branch on I/O error
72            MOV     #128.,R0 ; Region size in bytes
73            MOV     #BUFF,R4        ; R4 => character
74    LOOP:   MOVB    (R4),(R5)+ ; Move character to region
75            SOB     R0,LOOP ; Decrement counter and loop
76                            ;  until done
77            QIOW$C  IO.WVB,5,1,,IOSB,,<DNMES,LDNMES,40>
78                            ; Write region created message
79            BCS     ERR4D   ; Branch on dir error
80            TSTB    IOSB    ; Check for I/O error
81            BLT     ERR4I   ; Branch on I/O error
82            DIR$    #DET    ; Detach from region
83            BCS     ERR5    ; Check for error
84            EXIT$S
85    ; Error code
86    ERR1:   MOV     #FCRRER,R1      ; Create region error
87                                    ;  message
88            BR      SHOERR          ; Branch to common code
89    ERR2:   MOV     #FCRWER,R1      ; Create window message
90            BR      SHOERR          ; Branch to common code
91    ERR3D:  MOV     #FQI1DE,R1      ; QIO directive message
92            BR      SHOERR          ; Branch to common code
93    ERR3I:  MOV     IOSB,R0         ; Extend sign on status
94            MOV     R0,$DSW         ;  and move to ars block
95            MOV     #FQI1IE,R1      ; QIO I/O error
96            BR      SHOERR          ; Branch to common code
97    ERR4D:  MOV     #FQI2DE,R1      ; QIO write dir error
98            BR      SHOERR          ; Branch to common code
99    ERR4I:  MOV     IOSB,R0         ; Extend sign on status
100           MOV     R0,$DSW         ;  and move to ars block
```

SOLUTION

```
101             MOV     #FQI2IE,R1      ; QIO write err message
102             BR      SHOERR          ; Branch to common code
103     ERR5:   MOV     #FDETER,R1      ; Detach region message
104
105     SHOERR: MOV     #BUFF,R0        ; Set up for $EDMSG
106             MOV     #$DSW,R2        ;
107             CALL    $EDMSG          ; Edit message
108             QIOW$S  #IO.WVB,#5,#1,,,,<#BUFF,R1,#40>
109                                     ; Display message
110             EXIT$S                  ; Exit
111
112             .END    START
```

```
  1             PROGRAM LEX82A
  2     C
  3     C File LEX82A.FTN
  4     C
  5     C LEX82A creates a named region (attached on creation),
  6     C creates a virtual address window (mapped on creation),
  7     C places an ASCII character input at TI: at all locations
  8     C in the region, detaches from the region and exits,
  9     C leaving the region in existence.
 10     C
 11     C Task-build instructions:
 12     C
 13     C       >LINK/MAP/OPTIONS/CODE:FPP LEX82A,LB:[1,1]FOROTS-
 14     C       ->/LIBRARY
 15     C       Option? VSECT=DATA:160000:20000
 16     C       Option? WNDWS=1
 17     C       Option? <RET>
 18     C
 19     C RDB = Region Definition Block for region with the
 20     C following properties:
 21     C       Size            = 2 (32. word blocks)
 22     C       Name            = MYREG
 23     C       Partition       = GEN
 24     C       Protection      = WO:None,SY:RWED
 25     C                         OW:RWED,GR:RWED
 26     C       Do not mark for delete on last detach
 27     C       Attach with write and delete access
 28     C
 29     C WDB = Window Definition Block for window with the
 30     C following properties:
 31     C       APR             = 7
 32     C       Size            = 2 (32. word blocks)
 33     C       Offset in region = 0 (32. word blocks)
 34     C       Length in region = 2 (32. word blocks)
 35     C       Map on create with write access
 36     C
```

Dynamic Regions

SOLUTION

```
37            INTEGER RDB(8),WDB(8)
38    C Array for dynamic region, variable for ASCII character
39            BYTE ARRAY(128),CHAR
40            COMMON /DATA/ ARRAY
41    C
42    C Initialize the RDB
43            DATA RDB/0,2,3RMYR,3REG ,3RGEN,3R   ,"000152,"170000/
44    C Initialize the WDB
45            DATA WDB/"3400,0,2,0,0,2,"202,0/
46    C Call routine to create and attach region
47            CALL CRRG(RDB,IDS)
48    C Check for error
49            IF(IDS.LT.0)GOTO 800
50    C Create address window and map to region
51            WDB(4)=RDB(1)
52            CALL CRAW(WDB,IDS)
53    C Check for error
54            IF(IDS.LT.0)GOTO 810
55            WRITE (5,50)
56    C Get ASCII character
57    50      FORMAT ('$ENTER ASCII CHARACTER: ')
58            READ (5,60)CHAR
59    60      FORMAT (A1)
60    C Place data in region
61            DO 80 J=1,128
62            ARRAY(J)=CHAR
63    80      CONTINUE
64    C Detach from region
65            CALL DTRG(RDB,IDS)
66    C Check for error
67            IF(IDS.LT.0)GOTO 820
68    C Write message
69            TYPE *,'LEX82A HAS CREATED AND INITIALIZED THE
70            1REGION'
71    C Branch to common exit
72            GOTO 1000
73    C Write create error message
74    800     WRITE(5,805)IDS
75    805     FORMAT(' ERROR IN CREATING REGION, DSW = ',I4)
76    C Go to common exit
77            GO TO 1000
78    C Write attach error message
79    810     WRITE(5,815)IDS
80    815     FORMAT(' ERROR IN CREATING WINDOW AND MAPPING,
81            1DSW = ',I4)
82            GOTO 1000
83    C Write detach error message
84    820     WRITE(5,825)IDS
85    825     FORMAT(' ERROR IN DETACHING FROM REGION, DSW = '
86            1,I4)
87    C Common exit
88    1000    CALL EXIT
89            END
```

Dynamic Regions

SOLUTION

```
 1            .TITLE  LEX82B
 2            .IDENT  /01/
 3            .ENABL  LC                  ; Enable lower case
 4    ;+
 5    ; File LEX82B.MAC
 6    ;
 7    ; Program to attach to an existing region, create a
 8    ; virtual address window (mapped on creation), modify
 9    ; the first byte of the region, read ASCII data from the
10    ; region, detach from the region and mark it for delete,
11    ; and finally exit.  The region will be deleted on last
12    ; detach.
13    ;
14    ; Assemble and task-build instructions:
15    ;
16    ;       >MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[ufd]-
17    ;       ->LEX82B
18    ;       >LINK/MAP/OPTION LEX82B,LB:[1,1]PROGSUBS/LIBRARY
19    ;       >Option? WNDWS=1
20    ;       >Option? <RET>
21    ;-
22           .MCALL  EXIT$S,RDBBK$,WDBBK$,ATRG$C ; System
23           .MCALL  CRAW$,DTRG$S,DIR$,QIOW$S    ;  macros
24           .MCALL  DIRERR,IOERR    ; Supplied macros
25    RDB:   RDBBK$  0,MYREG,GEN,<RS.WRT!RS.RED!RS.MDL!RS.DEL>
26    ;      Define region with:
27    ;              Size            = 0 (32. word blocks)
28    ;                                returned after attach
29    ;              Name            = MYREG
30    ;              Partition       = GEN
31    ;              Mark for delete on last detach
32    ;              Attach with read, write and delete access
33    ;
34    WIN:   CRAW$   WDB     ;DPB for create address window
35    WDB:   WDBBK$  7,200,0,0,0,<WS.MAP!WS.RED!WS.WRT>
36    ;      Define window with:
37    ;              APR              = 7
38    ;              Size             = 200 (32. word blocks)
39    ;              Offset in region = 0 (32. word blocks)
40    ;              Length in region = 0 (32. word blocks)
41    ;                                 returned when mapped
42    ;              Map on create with read and write access
43    ;
44    IOSB:  .BLKW   2               ; I/O status block
45    RSIZ   =128.                   ; Region size in bytes
46    START: ATRG$C  RDB             ; Attach to region
47           BCS     ERR1            ; Check for error
48           MOV     RDB+R.GID,WDB+W.NRID ; Move region ID
49                                   ; into WDB
50           DIR$    #WIN            ; Create window
```

SOLUTION

```
51            BCS     ERR2              ; Check for error
52            MOV     #160000,R5        ; Set base addr in region
53            MOVB    #'Z,(R5)          ; Place Z in 1st byte
54            MOV     #RSIZ,R4          ; Size of region in bytes
55            MOV     #64.,R3           ; Chars per line
56    LOOP:   QIOW$S  #IO.WVB,#5,#1,,#IOSB,,<R5,R3,#40>
57                                      ; Write data
58            BCS     ERR3D             ; Check for dir error
59            TSTB    IOSB              ; Check for I/O error
60            BLT     ERR3I             ; Branch on error
61            SUB     R3,R4             ; Compute chars left
62            BLE     DONE              ; Branch if done
63            ADD     R3,R5             ; Point to next char
64            CMP     R3,R4             ; Check for < 64. chars
65                                      ;  left to print
66            BLE     LOOP              ; > or =, print next line
67            MOV     R4,R3             ; <, print only that many
68                                      ;  chars
69            BR      LOOP              ; Print the line
70    DONE:   DTRG$S  #RDB              ; Detach from region
71            BCS     ERR4              ; Check for error
72            EXIT$S
73    ; Error handling code
74    ERR1:   DIRERR  <ERROR ATTACHING TO REGION>
75    ERR2:   DIRERR  <ERROR CREATING WINDOW AND MAPPING>
76    ERR3D:  DIRERR  <ERROR WRITING DATA>
77    ERR3I:  IOERR   #IOSB,<ERROR WRITING DATA>
78    ERR4:   DIRERR  <ERROR DETACHING FROM REGION>
79            .END    START
```

```
 1            PROGRAM LEX82B
 2    C
 3    C File LEX82B.FTN
 4    C
 5    C FORTAN program to attach region MYREG in partition GEN
 6    C (which was created by LEX82A), create a window and map
 7    C it to the region upon creation, place a Z in the first
 8    C byte, then read data out of the region, and detach
 9    C from it, deleting it in the process.
10    C
11    C Task-build with these options:
12    C                 VSECT=DATA:160000:20000
13    C                 WNDWS=1
14    C
```

Dynamic Regions

SOLUTION

```
15    C
16            INTEGER RDB(8),WDB(8)
17            BYTE IDATA(128)
18    C This common block will alisn with the address window
19            COMMON /DATA/IDATA
20    C RDB = Resion definition block with the followins
21    C properties:
22    C       Size              0 (32.-word blocks)
23    C                           filled in when attached
24    C       Name              MYREG
25    C       Partition         GEN
26    C       Protection        WO:none,SY:RWED,OW:RWED,GR:RWED
27    C       Mark for delete on last detach
28    C       Attach with delete, write and read access
29    C Initialize the RDB
30            DATA RDB /0,0,3RMYR,3REG ,3RGEN,3R    ,"000213,
31            1"170000/
32    C
33    C WDB = Window definition block with the followins
34    C properties:
35    C       APR                 7
36    C       Size                200(8) (32.-word blocks)
37    C       Offset in resion    0 (32.-word blocks)
38    C       Lensth of window    0 (32.-word blocks)
39    C                             filled in when mapped
40    C       Map on create with read access
41    C Initialize the WDB
42            DATA WDB /"3400,0,"200,0,0,0,"203,0/
43    C
44    C Attach resion
45            CALL ATRG (RDB,IDS)
46    C Check for error on attach
47            IF (IDS .LT. 0) GOTO 100
48    C Move resion id to WDB
49            WDB(4)=RDB(1)
50    C Create and map window
51            CALL CRAW (WDB,IDS)
52    C Check for error
53            IF (IDS .LT. 0) GOTO 200
54    C Place ASCII Z in first byte
55            IDATA(1)='Z'
56    C Print contents of resion
57    10      WRITE (5,11) IDATA
58    11      FORMAT (' ',64A1)
59    C Detach from resion and delete it
60            CALL DTRG (RDB,IDS)
61    C Check for error
62            IF (IDS .LT. 0) GOTO 300
63    C And Jump to exit
64            GOTO 500
```

SOLUTION

```
65    C
66    C Error messages
67    100     WRITE (5,101) IDS
68    101     FORMAT (' ERROR ATTACHING TO REGION, DSW =',I4)
69            GOTO 500
70    200     WRITE (5,201) IDS
71    201     FORMAT (' ERROR IN CREATING WINDOW, DSW =',I4)
72            GOTO 500
73    300     WRITE (5,301) IDS
74    301     FORMAT (' ERROR DETACHING FROM REGION, DSW =',I4)
75    C
76    500     CALL EXIT
77            END
```

SOLUTION

3.

```
        .TITLE  SNDREF
        .IDENT  /01/
        .ENABL  LC                      ; Enable lower case
;+
; File LEX83A.MAC                       ;;EX
;
; Modified to send to a 2nd receiver RCVRF2 in  ;;EX
; addition to RCVREF                            ;;EX
;
; LEX83A creates a 64-word (2 block) unnamed region and
; fills it with ASCII characters. It then sends the
; region to RCVREF, and then waits for RCVREF to receive
; the region. (This is signalled by event flag #1.) It
; then prints a message and exits. Since the area is
; unnamed, it is automatically deleted when the last
; attached task exits.
;
; Assemble and task-build instructions:
;
;       >MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[ufd]-;;EX
;       ->LEX83A
;       >LINK/MAP/OPTION LEX83A,LB:[1,1]PROGSUBS/LIBRARY
;       Option? WNDWS=1
;
; Install and run instructions: RCVREF must be installed.
; LEX83B must be installed as RCVRF2. Run LEX83A first,
; then run RCVREF and RCVRF2 (either one first)
;-
        .MCALL  QIOW$C,QIOW$S,RQST$C ; System macros
        .MCALL  WTSE$C,EXIT$S,RDBBK$,WDBBK$
        .MCALL  CRRG$S,CRAW$S,SREF$C
        .MCALL  DIRERR              ; Supplied macro
        .NLIST  BEX                 ; SUPPRESS DATA

; Define region with:
;       Size            = 2     32-WORD BLOCKS
;       Name            = none
;       Partition       = GEN
;       Protection      = WO:none,GR:RWED
;                         OW:RWED,SY:none
;       Attach on create
;       Read and write access desired on attach
RPRO    = 170017
RSTAT   = RS.ATT!RS.RED!RS.WRT

RDB:    RDBBK$  2,,GEN,RSTAT,RPRO
```

Dynamic Regions

SOLUTION

```
48    ; Define window with:
49    ;       APR                 = 7
50    ;       Size                = 2   32-word blocks
51    ;       Offset in region    = 0   32-word blocks
52    ;       Length to map       = 0   32-word blocks (defaults
53    ;                                 to smaller of region
54    ;                                 size and window length)
55    ;       Map on create with read and write access
56    WSTAT   = WS.MAP!WS.WRT
57
58    WDB:    WDBBK$  7,2,0,0,,WSTAT
59    ;
60    MES1:   .ASCII  / LEX83A HAS CREATED THE REGION AND HAS/
61            .ASCII  / SENT IT TO RCVREF AND RCVRF2./  ;;EX
62    LMES1   =.-MES1
63    MES2:   .ASCII  / RCVREF AND RCVRF2 HAVE RECEIVED IT./;;EX
64            .ASCII  / LEX83A IS NOW EXITING./           ;;EX
65    LMES2 = . - MES2
66            .LIST   BEX             ; Show binary extensions
67            .EVEN
68            .ENABL  LSB             ; Enable local symbol
69                                    ;  blocks
70    START:  CRRG$S  #RDB            ; Create and attach to
71                                    ;  region
72            BCS     1$              ; Branch on dir error
73            MOV     RDB+R.GID,WDB+W.NRID ; Copy region ID
74                                         ;  into WDB
75            CRAW$S  #WDB            ; Create and map window
76            BCS     2$              ; Branch on dir error
77            MOV     WDB+W.NBAS,R0   ; base V.A. of region
78    ; Fill region with all M's
79            MOV     #64.,R3         ; count of words to move
80    20$:    MOV     #"MM,(R0)+      ; Move in an ASCII M
81            SOB     R3,20$          ; Loop throush region
82    ; Send the region to RCVREF. EF 1 will be set when
83    ; RCVREF recieves it
84            SREF$C  RCVREF,WDB,1    ; Send by reference to
85                                    ;  RCVREF
86            BCS     3$              ; Branch on dir error
87            SREF$C  RCVRF2,WDB,3    ; Send by reference ;;EX
88                                    ;  to RCVRF2        ;;EX
89            BCS     7$              ; Branch on dir error;;EX
90            QIOW$C  IO.WVB,5,2,,,,<MES1,LMES1,40> ; Display
91                                                  ;  message
92            BCS     4$              ; Branch on dir error
93            WTSE$C  1               ; Wait for RCVREF to set
94                                    ; the region
95            BCS     5$              ; Branch on dir error
96            WTSE$C  3               ; Wait for RCVRF2 to ;;EX
97                                    ;  set the region    ;;EX
98            BCS     8$              ; Branch on dir error;;EX
```

SOLUTION

```
 99          QIOW$C  IO.WVB,5,2,,,,<MES2,LMES2,40> ; Display
100                                                ;  message
101          BCS     6$              ; Branch on dir error
102          EXIT$S                  ; Exit
103    ; Error code
104    1$:   DIRERR  <ERROR ON CREATE OR ATTACH REGION>
105    2$:   DIRERR  <ERROR ON CREATE OR MAP WINDOW>
106    3$:   DIRERR  <ERROR ON SEND BY REFERENCE>
107    4$:   DIRERR  <ERROR ON 1ST WRITE>
108    5$:   DIRERR  <ERROR ON WAIT FOR>
109    6$:   DIRERR  <ERROR ON 2ND WRITE>
110    7$:   DIRERR  <ERROR ON 2ND SEND BY REFERENCE>     ;;EX
111    8$:   DIRERR  <ERROR ON 2ND WAIT FOR>              ;;EX
112          .END    START
```

```
  1          PROGRAM SNDREF
  2    C
  3    C File LEX83A.FTN
  4    C
  5    C Modified to send the region by reference to RCVRF2 !!EX
  6    C in addition to RCVREF                               !!EX
  7    C
  8    C This program creates a 64-word unnamed region and
  9    C fills it with ASCII characters.  It then sends it by
 10    C reference to task RCVREF, and waits for RCVREF to
 11    C receive the region.(This is signalled by event flag
 12    C #1.)  SNDREF then prints a message and exits.  Since
 13    C the area is unnamed, it is automatically deleted when
 14    C the last attached task exits.
 15    C
 16    C Task-build instructions:
 17    C
 18    C       >LINK/MAP/CODE:FPP/OPTIONS LEX83A,LB:[1,1]F0-!!EX
 19    C       ->ROTS/LIBRARY                               !!EX
 20    C       Option? WNDWS=1
 21    C       Option? VSECT=DATA:160000:200
 22    C       Option? <RET>
 23    C
 24    C Install and run instructions: RCVREF must be installed.
 25    C LEX83B must be installed under the name RCVRF2.   !!EX
 26    C Run LEX83A first, then run RCVREF and RCVRF2 (in  !!EX
 27    C either order)
 28    C
 29    C       RDB = Region definition block with the following
 30    C       properties:
 31    C               Size            2 32-word blocks
 32    C               Name            none
 33    C               Partition       GEN
 34    C               Protection      WO:none,SY:RWED,OW:RWED,
 35    C                               GR:none
 36    C               Attach on creation
 37    C               Read and write access desired on attach
 38    C
```

SOLUTION

```
39   C       WDB = Window definition block with the followins
40   C       properties:
41   C               APR               7
42   C               Size              2 32-word blocks
43   C               Offset in region  0 32-word blocks
44   C               Length of region  2 32-word blocks
45   C               Map on create with write access
46   C
47           INTEGER RDB(8),WDB(8),RCV(2),RCV2(2)          !!EX
48   C This common block will align with the address window
49           COMMON /DATA/IDATA(64)
50   C Initialize the RDB
51           DATA RDB/0,"2,0,0,3RGEN,3R   ,"43,"170017/
52   C Initialize the WDB
53           DATA WDB/"3400,0,"2,0,0,"2,"202,0/
54   C Name of receiver task
55           DATA RCV/3RRCV,3RREF/
56           DATA RCV2/3RRCV,3RRF2/                        !!EX
57   C Code
58           CALL CRRG(RDB,IDS)       ! Create region
59           IF (IDS .LT. 0) GOTO 100 ! Check for error
60           WDB(4)=RDB(1)            ! Move region id to WDB
61           CALL CRAW(WDB,IDS)       ! Create window
62           IF (IDS .LT. 0) GOTO 200 ! Check for error
63   C Fill region with data
64           DO 10 I=1,64
65   10      IDATA(I)='MM'
66   C Send-by-reference to receiver task, set event flag 1
67   C when received
68           CALL SREF(RCV,1,WDB,,IDS)
69           IF (IDS .LT. 0) GOTO 400 ! Check for error
70   C Send-by-reference to 2nd receiver, RCVRF2, use event!!EX
71   C flag 2                                               !!EX
72           CALL SREF(RCV2,2,WDB,,IDS)                    !!EX
73           IF (IDS .LT. 0) GOTO 450 ! Check for error    !!EX
74           TYPE *,' LEX83A HAS CREATED THE REGION AND HAS
75           1 SENT IT TO RCVREF AND RCVRF2.' ! Display    !!EX
76   C                                        !  message   !!EX
77           CALL WAITFR(1,IDS)       ! Now wait for reception
78           IF (IDS .LT. 0) GOTO 500 ! Check for error
79           CALL WAITFR(2,IDS)       ! Wait for RCVRF2 to !!EX
80   C                                !  receive           !!EX
81           IF (IDS .LT. 0) GOTO 550 ! Check for error    !!EX
82           TYPE *,' RCVREF AND RCVRF2 HAVE RECEIVED IT.
83           1 LEX83A IS NOW EXITING.' ! Write message     !!EX
84           GOTO 600                 ! And go exit
```

SOLUTION

```
C Error handling code
100     WRITE (5,110)IDS
110     FORMAT (' ERROR CREATING REGION, DSW = ',I4)
        GOTO 600
200     WRITE (5,210)IDS
210     FORMAT (' ERROR CREATING WINDOW, DSW = ',I4)
        GOTO 600
400     WRITE (5,410)IDS
410     FORMAT (' ERROR IN SEND-BY-REFERENCE, DSW = ',I4)
        GOTO 600
450     WRITE (5,460)IDS
460     FORMAT (' ERROR IN 2ND SEND-BY-REFERENCE, DSW
        1 = ',I4)                                          !!EX
        GOTO 600                                           !!EX
500     WRITE (5,510)IDS
510     FORMAT (' ERROR ON WAIT, DSW = ',I4)
        GOTO 600                                           !!EX
550     WRITE (5,560)IDS                                   !!EX
560     FORMAT (' ERROR ON 2ND WAIT, DSW = ',I4)           !!EX
C
600     CALL EXIT
        END
```

```
        .TITLE  LEX83B
        .IDENT  /01/
        .ENABL  LC              ; Enable lower case
;
; File LEX83B.MAC
;
; Second reciever for SNDREF (modifed to LEX83A).
; Program to receive-by-reference (mapped on creation),
; modify the first data byte in the region,
; read ASCII data from the region, detach from the
; region and exit. The region will be deleted on last
; detach.
;
; The first word in the region contains the count of the
; number of bytes of data in the region.
;
; Assemble and task build instructions:
;
;       >MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[ufd]
;       ->LEX83B
;       LINK/MAP/OPTIONS LEX83B,LB:[1,1]PROGSUBS/LIBRARY
;       option? WNDWS=1
```

SOLUTION

```
23    ;
24    ; Install and run instructions: RCVREF must be installed.
25    ; LEX83B must be installed as RCVRF2. Run LEX83A first
26    ; and then run RCVREF and RCVRF2 (in either order).
27    ;
28            .MCALL  EXIT$S,WDBBK$,RREF$ ; External system
29            .MCALL  QIOW$S,CRAW$,DIR$    ;  macros
30            .MCALL  DIRERR,IOERR     ; External supplied
31                                     ;  macros
32    ; Define window with:
33    ;               APR               = 7
34    ;               Size              = 200(8) (32. word blocks)
35    ;       These are filled in on receive as set by sender:
36    ;               Offset in region = 0 (32. word blocks)
37    ;               Length in region = 0 (32. word blocks)
38    ;                                    reset after mapping
39    ;               Access            = 0
40    ; Note: Must map separately (or as part of receive)
41    WDB:    WDBBK$  7,2
42    ;
43    REC:    RREF$   WDB              ; Set up DPB for RREF$
44    WIN:    CRAW$   WDB              ; Set up DPB for CRAW$
45    IOSB:   .BLKW   2                ; I/O status block
46
47    START:  DIR$    #WIN             ; Create virtual address
48                                     ;  window
49            BCS     ERR1             ; Branch on error
50            BIS     #WS.MAP,WDB+W.NSTS ; Set WDB to map on
51                                       ; receive
52            DIR$    #REC             ; Receive by reference
53                                     ;  and map
54            BCS     ERR2             ; Branch on error
55            MOV     #160000,R5       ; Set base address in
56                                     ;  region
57            MOV     WDB+W.NLEN,R3    ; Size of region to R3
58            MUL     #64.,R3          ; Convert blocks to bytes
59            MOV     #'9,(R5)         ; Modify first data byte
60    ;
61            QIOW$S  #IO.WVB,#5,#1,,#IOSB,,<R5,R3,#40> ;Write
62                                                      ;  data
63            BCS     ERR3             ; Branch on directive
64                                     ;  error
65            TSTB    IOSB             ; Check for I/O error
66            BLT     ERR4             ; Branch on error
67            EXIT$S
68    ; Error code
69    ERR1:   DIRERR  <ERROR CREATING VIRTUAL ADDRESS WINDOW>
70    ERR2:   DIRERR  <ERROR ON RECEIVE AND MAP>
71    ERR3:   DIRERR  <ERROR ON WRITE QIO>
72    ERR4:   IOERR   #IOSB,<ERROR ON WRITE QIO>
73            .END    START
```

Dynamic Regions

SOLUTION

```
1           PROGRAM LEX83B
2   C
3   C File LEX83B.FTN
4   C
5   C LEX83B receives by reference a region from the task
6   C LEX83A.  It maps to the region, modifies the first
7   C byte, prints out the contents, and exits. The region
8   C is deleted on last detach.
9   C
10  C Task-build instructions: Include these options
11  C                 WNDWS=1
12  C                 VSECT=DATA:160000:20000
13  C
14  C Install and run instructions: LEX83B must be installed.
15  C as RCVRF2. RCVREF must be installed. Run LEX83A first,
16  C then run LEX83B and RCVREF (in either order).
17  C
18  C WDB = Window definition block with:
19  C       APR                 7
20  C       Size                200(8) 32-word blocks
21  C                                  Allow for full APR
22  C       Offset in region    0 32-word blocks
23  C       Length of region    0 32-word blocks (to be filled
24  C                             in on receive)
25  C       Read and write access
26          INTEGER WDB(8)
27          DATA WDB/"3400,0,"2,0,0,"0,"3,0/
28          BYTE DATA(128)
29  C This common block will align with the address window
30          COMMON /DATA/DATA
31  C
32  C Create address window--do not map at this time
33          CALL CRAW(WDB,IDS)
34  C Check for error on create
35          IF (IDS .LT. 0) GOTO 200
36  C Now set WDB status for mapping--will be done by
37  C receive-by-reference
38          WDB(7)=WDB(7)+"200
39  C Receive data and map
40          CALL RREF(WDB,,IDS)
41  C Check for error
42          IF (IDS .LT. 0) GOTO 100
43  C Modify first value
44          DATA(1)='9'
45  C Calculate number of bytes of data - length in blocks
46  C returned at WDB(6)
47          NCHAR = 64*WDB(6)
48          WRITE(5,10) (DATA(I),I=1,NCHAR)
49  10      FORMAT (' ',64A1)
50  C Go exit
51          GOTO 300
```

SOLUTION

```
52    C       Error messages
53    100     WRITE(5,110)IDS
54    110     FORMAT (' ERROR ON RECEIVE-BY-REFERENCE, DSW =',I4)
55            GOTO 300
56    200     WRITE(5,210)IDS
57    210     FORMAT (' ERROR CREATING WINDOW, DSW =',I4)
58    300     CALL EXIT
59            END
```

TEST/EXERCISE

1. Next to each activity, write O for open, I for I/O operation, or C for close, to identify which step of file I/O is involved.

   ___ a. Records are read from the file.

   ___ b. Access rights to the file are checked.

   ___ c. Existing file is located on disk.

   ___ d. Internal buffers are placed in a pool for re-use.

   ___ e. Records are written to a file.

2. Describe three functions performed by the Files-11 ancillary control processor (F11ACP) when a task creates a new file containing seven blocks.

TEST/EXERCISE

3. For each of the following, tell whether FCS only, RMS only, or both can be used for file I/O. If both can be used, identify which you would prefer and why.

   a. A teacher has a file with one record for each student. The students are identified by student number (1 - 1ØØ). Each record contains the student's test scores (space is reserved for 1Ø scores) and his average. The instructor adds new test scores and updates the averages as he gives tests. In addition, he wants to access any student's test scores and test average using the student number.

TEST/EXERCISE

b. A company has a file of customer records. Each record contains the company name, the address, the contact person, and the equipment bought. At different times, the records are accessed using company name, city, or contact person.

c. A company uses COBOL for its applications. It has a payroll file which is processed in order every two weeks.

SOLUTION

1. Next to each activity, write O for open, I for I/O operation, or C for close, to identify which step of file I/O is involved.

   I a. Records are read from the file.

   O b. Access rights to the file are checked.

   O c. Existing file is located on disk.

   C d. Internal buffers are placed in a pool for re-use.

   I e. Records are written to a file.

2. Describe three functions performed by the Files-11 ancillary control processor (F11ACP) when a task creates a new file containing seven blocks.

   Any three of the following:

   Allocate a file header

   Initialize the file header

   Set up file retrieval pointers

   Create a directory entry

   Allocate blocks to the file

   Connect a task's LUN to the file

SOLUTION

3. For each of the following, tell whether FCS only, RMS only, or both can be used for file I/O. If both can be used, identify which you would prefer and why.

   a. A teacher has a file with one record for each student. The students are identified by student number (1 - 10Ø). Each record contains the student's test scores (space is reserved for 1Ø scores) and his average. The instructor adds new test scores and updates the averages as he gives tests. In addition, he wants to access any student's test scores and test average using the student number.

      Either FCS or RMS may be preferred.

      FCS

      - Easier to program (in MACRO-11)
      - Less overhead (although very close if using a relative file rather than an indexed file)
      - File must have fixed length records with record numbers corresponding to student numbers

      RMS

      - If a relative file is used, can automatically skip over deleted records (if student leaves or drops the course)
      - In FORTRAN, no harder to program
      - Not much overhead for a relative file
      - File must be a relative file with fixed length records, with cell numbers corresponding to student numbers

SOLUTION

b. A company has a file of customer records. Each record contains the company name, the address, the contact person, and the equipment bought. At different times, the records are accessed using company name, city, or contact person.

   Best answer is RMS only since an indexed file with multiple keys is needed for fastest access. FCS can be used, but access by key value is impossible. You would have to step through the file, checking all records, to locate the one you want.

c. A company uses COBOL for its applications. It has a payroll file which is processed in order every two weeks.

   RMS only; COBOL is supported under RMS, but not under FCS.

TEST/EXERCISE

1. Modify CRESEQ so that each record in the file contains the text input from the terminal preceded by "AAAA".

2. Write a task that appends records to a file you have created (using one of the FCS example programs or the editor).

3. In MACRO-11, modify the task CREFXA so that input from the terminal uses FCS routines instead of QIO directives.

4. Write a task that requests input from a terminal of the form:

   n, text

   Use the input to update the nth record of FIXED.ASC, which has fixed length records. Use random access and do not truncate the file.

5. In MACRO-11, modify the task BLOCK1 or BLOCK2 so that it writes or displays two virtual blocks at a time.

6. (Optional) In MACRO-11, modify the task CSI so that the subroutines DISPLY and DELETE actually display and delete the file. Caution: DELET$ delete the highest version of a file if no version number is specified. (See Chapter 6 of the IAS/RSX I/O Operations Reference Manual for information about the routines GCML and CSI.)

SOLUTION

1.

```
        .TITLE  CRESEQ
        .IDENT  /01/
        .ENABL  LC
;+
; File LEX101.MAC
;
; Modified to preced each record with AAAA
;
; CRESEQ creates a file VARI.ASC. It reads
; records from TI:, and places them in the file.
; A ^Z terminates input and closes the file.
;
; Assemble and task-build instructions:
;
;       MACRO/LIST LB:[1,1]PROGMACS/LIBRARY,dev:[ufd]-
;       ->CRESEQ
;       LINK/MAP CRESEQ,LB:[1,1]PROGSUBS/LIBRARY
;-

        .MCALL  EXST$C,QIOW$C,QIOW$,DIR$ ; System macros
        .MCALL  FSRSZ$,FDBDF$,FDAT$A,FDRC$A,FDOP$A ;
        .MCALL  NMBLK$,OPEN$W,PUT$,CLOSE$ ;
        .MCALL  DIRERR,IOERR,FCSERR ; Supplied macros

        FSRSZ$  1               ; 1 file for record I/O

; Define file descriptor block for VARI.ASC

FDB:    FDBDF$                  ; Allocate the FDB
        FDAT$A  R.VAR,FD.CR     ; Variable length records,
                                ;  Listing - implied
                                ;  carriage return, line
                                ;  feed
        FDRC$A  ,BUFF           ; Sequential access and
                                ;   record I/O by
                                ;   default, BUFF is
                                ;   user record buffer
        FDOP$A  1,,FNAME        ; Use LUN 1, file spec
                                ;   at FNAME
FNAME:  NMBLK$  VARI,ASC        ; "VARI.ASC"

; Local Data
BUFF:   .ASCII  /AAAA/          ; USER RECORD BUFFER  ;;EX
INBUF:  .BLKB   80.             ;                     ;;EX
IOST:   .BLKW   2               ; I/O STATUS BLOCK

        .LIST   BEX
        .EVEN

        .ENABL  LSB
```

SOLUTION

```
51
52   START:
53
54   ; Open file for write, call ERR1 if open fails
55            OPEN$W  #FDB,,,,,,ERR1
56   ; Get record from terminal, put to file.
57   10$:     QIOW$C  IO.RVB,5,1,,IOST,,<INBUF,80.> ;        ;;EX
58            BCS     ERR2D               ; Branch on directive
59                                        ;   error
60            TSTB    IOST                ; Check for I/O error
61            BLT     ERR2I               ; Branch on I/O error
62            MOV     IOST+2,R1           ; Number of bytes input
63            ADD     #4,R1               ; Add 4 for 4 A's      ;;EX
64            PUT$    #FDB,,R1            ; Put record to file
65            BCS     ERR3
66            BR      10$                 ; GET NEXT RECORD
67
68   EXIT:    CLOSE$  #FDB,ERR4           ; Close file
69            EXST$C  EX$SUC              ; EXIT STATUS IS 1
70
71            .SBTTL  ERROR HANDLER
72
73   ; Error code - Close file if necessary, display error
74   ; message and exit
75
76   ERR1:    FCSERR  #FDB,<ERROR OPENING FILE>
77   ERR2D:   DIRERR  <DIRECTIVE ERROR ON READ>
78   ERR2I:   CMPB    #IE.EOF,IOST        ; Is it ^Z?
79            BEQ     EXIT                ; If equal, close file
80                                        ;  and exit
81            IOERR   #IOST,<ERROR ON READ> ; Display error
82                                        ;  message and exit
83   ERR3:    CLOSE$  #FDB,ERR4           ; Close file
84            FCSERR  #FDB,<ERROR WRITING RECORD>
85   ERR4:    FCSERR  #FDB,<ERROR CLOSING FILE>
86            .END    START
```

File Control Services

SOLUTION

```
1           PROGRAM CRESEQ  !CREATE FILE SEQUENTIALLY
2   C
3   C FILE LEX101.FTN
4   C
5   C Modified to precede each record with AAAA      !!EX
6   C
7   C This task creates a file of VARI.ASC of
8   C variable-length records using sequential record access.
9   C The records are input from the terminal and copied to
10  C the file.  The process stops when the operator types
11  C CTRL/Z at the terminal.
12  C
13          BYTE BUFF(84),INBUF(80)                   !!EX
14          EQUIVALENCE (BUFF(5),INBUF(1))            !!EX
15          INTEGER LEN
16          DATA BUFF(1),BUFF(2),BUFF(3),BUFF(4)
17          1 /'A','A','A','A'/
18  C
19  C
20  C OPEN FILE
21  C
22  C Default access is sequential
23  C Default is formatted I/O for sequential files
24  C
25          OPEN    (UNIT=1,NAME='VARI.ASC',TYPE='NEW',
26          1       CARRIAGECONTROL='LIST')
27  C
28          TYPE    *,'TYPE IN TEXT, TERMINATE EACH RECORD
29          1 WITH A CARRIAGE RETURN'
30          TYPE    *,'TERMINATE INPUT WITH A CTRL/Z'
31  C Loop
32  10      READ (5,11,END=100) LEN,INBUF    ! Read record!!EX
33  11      FORMAT (Q,80A1)
34  C
35          LEN = LEN+4                      ! Add 4 for A's
36  C                                                  !!EX
37          WRITE (1,12) (BUFF(I),I=1,LEN)   ! Write record
38  12      FORMAT (80A1)                    !  to file
39          GO TO 10
40  C Close file and exit
41  100     CLOSE   (UNIT=1)
42          CALL EXIT
43          END
```

SOLUTION

2.

```
 1          .TITLE  LEX102
 2          .IDENT  /01/
 3          .ENABL  LC                  ; Enable lower case
 4  ;+
 5  ; File LEX102.MAC
 6  ;
 7  ; LEX102 appends records to the end of the file
 8  ; TEST.FIL, setting the input from TI:. TEST.FIL
 9  ; contains variable length records and can be
10  ; created using the editor. A ^Z terminates input
11  ; and closes the file.
12  ;-
13
14          .MCALL  EXST$C,QIOW$C,QIOW$,DIR$
15
16          .MCALL  FSRSZ$,FDBDF$,NMBLK$
17          .MCALL  FDRC$A,FDAT$A,FDOP$A
18          .MCALL  OPEN$A,PUT$,CLOSE$
19
20          .NLIST  BEX                 ; Suppress ASCII
21  IOST:   .BLKW   2                   ; QIO status block
22  PRINT:  QIOW$   IO.WVB,5,1,,,,<OBUFF,0,40>
23  BUFF:   .BLKB   80.                 ; User record buffer
24  OBUFF:  .BLKB   80.                 ; Output buffer for
25                                      ;  error messages
26  ARG:    .BLKW   1                   ; Argument block for
27                                      ;  $EDMSG
28  EFDQIO: .ASCIZ  /DIRECTIVE ERROR ON QIO. ERROR CODE = %D./
29  EFIQIO: .ASCIZ  ?I/O ERROR ON QIO. ERROR CODE = %D.?
30  EFCDIR: .ASCIZ  /FCS DIRECTIVE ERROR. ERROR CODE = %D./
31  EFCSIO: .ASCIZ  ?FCS I/O ERROR CODE. ERROR CODE = %D.?
32
33          .EVEN
34          .LIST   BEX                 ; Show offsets
35
36          FSRSZ$  1                   ; 1 file for record I/O
37
38  FDB:    FDBDF$                      ; File descriptor block
39          FDRC$A  ,BUFF,80.           ; User buffer and size
40          FDOP$A  1,,FILE             ; use LUN 1
41  FILE:   NMBLK$  TEST,FIL            ; TEST.FIL
42
43          .ENABL  LSB
44
45  START:  OPEN$A  #FDB,,,,,,ERR1     ; OPEN  for append; if
46                                      ;  open fails, CALL ERR1
47  10$:    MOV     #80.,R1             ; Size of URB
48          MOV     #BUFF,R2            ; Addr of URB
49  20$:    MOVB    #' ,(R2)+           ; Blank fill record
50          SOB     R1,20$              ;  so no garbage fill
```

SOLUTION

```
51            QIOW$C   IO.RVB,5,1,,IOST,,<BUFF,80.>; Read a
52                                                 ;  line from TI:
53            BCC      DIROK                       ; Branch on Directive ok
54            MOV      #EFIQIO,R1                  ; Set up for $EDMSG
55            MOV      #$DSW,R2                    ;
56            BR       SHOERR                      ; Branch to show error
57                                                 ;  and exit
58   DIROK:   TSTB     IOST                        ; Check for I/O error
59            BGT      OKIO                        ; Branch if I/O ok
60            CMPB     #IE.EOF,IOST                ; Check for EOF
61            BEQ      EXIT                        ; If EQ, close and exit
62            MOVB     IOST,R0                     ; I/O status is sign
63                                                 ;  extended and placed
64                                                 ;  in argument block
65            MOV      R0,ARG                      ;  for $EDMSG call
66            MOV      #ARG,R2                     ; Set up for $EDMSG call
67            MOV      #EFDQIO,R1                  ;
68            BR       SHOERR                      ; Branch to show error
69                                                 ;  and exit
70   OKIO:    MOV      IOST+2,R1                   ; Length of record to R1
71            PUT$     #FDB,,R1,ERR2               ; Write next record
72            BR       10$                         ; Get next record
73
74   EXIT:    CLOSE$   #FDB                        ; Close file
75            BCS      ERR3                        ; Branch on FCS error
76            EXST$C   EX$SUC                      ; Exit with status of 1
77
78   ; Error  Processing
79   ERR1:
80   ERR2:
81   ERR3:    TSTB     F.ERR+1(R0)                 ; Directive error or I/O
82                                                 ;  error
83            BEQ      IO                          ; Branch on I/O error
84            MOV      #EFCDIR,R1                  ; Set up for $EDMSG,
85                                                 ;  directive error
86            BR       FINSET                      ; Branch to finish setup
87   IO:      MOV      #EFCSIO,R1                  ; Set up for $EDMSG, I/O
88                                                 ;  error
89   FINSET:  MOVB     F.ERR(R0),R0                ; FCS error code
90            MOV      R0,ARG                      ;  is sign extended and
91            MOV      #ARG,R2                     ;  placed in arg block
92                                                 ; $EDMSG argument block
93   SHOERR:  MOV      #OBUFF,R0                   ; Output buffer
94            CALL     $EDMSG                      ; Format error message
95            MOV      R1,PRINT+Q.IOPL+2 ; Size of message
96            DIR$     #PRINT                      ; Print error message
97            CLOSE$   #FDB                        ; Close file
98            EXST$C   EX$ERR                      ; Exit with status of 2
99            .END     START
```

SOLUTION

```
 1            PROGRAM LEX102
 2    C
 3    C FILE LEX102.FTN
 4    C
 5    C This task appends records to the file TEST.FIL.
 6    C The records are input from the terminal and copied to
 7    C the file.  The process stops when the operator types
 8    C CTRL/Z at the terminal.
 9    C TEST.FIL contains variable length records and can
10    C be created using the editor.
11    C
12            BYTE BUFF(80)
13            INTEGER LEN
14    C
15    C
16    C OPEN FILE
17    C
18    C Default access is sequential
19    C Default is formatted I/O for sequential files
20    C
21            OPEN    (UNIT=1,NAME='TEST.FIL',TYPE='OLD',
22            1       CARRIAGECONTROL='LIST',ACCESS='APPEND')
23    C
24            TYPE    *,'TYPE IN TEXT, TERMINATE EACH RECORD
25            1 WITH A CARRIAGE RETURN'
26            TYPE    *,'TERMINATE INPUT WITH A CTRL/Z'
27    C Loop
28    10      READ (5,11,END=100) LEN,BUFF    ! Read record
29    11      FORMAT (Q,80A1)
30    C
31            WRITE (1,12) (BUFF(I),I=1,LEN)  ! Write record
32    12      FORMAT (80A1)                   !  to file
33            GO TO 10
34    C Close file and exit
35    100     CLOSE   (UNIT=1)
36            CALL EXIT
37            END
```

SOLUTION

3.

```
1           .TITLE  CREFXA
2           .IDENT  /01/
3           .ENABL  LC                  ; Enable lower case
4   ;+
5   ; File LEX103.MAC
6   ;
7   ; Modified to use FCS instead of QIO's to set ;;EX
8   ; input from TI:                                ;;EX
9   ;
10  ; CREFXA opens FIXED.ASC for write, inputs records
11  ; from TI: and puts them sequentially to the file.
12  ; A ^Z terminates input and closes the file.
13  ;-
14
15          .MCALL  EXST$C,QIOW$C,QIOW$,DIR$
16
17          .MCALL  FSRSZ$,FDBDF$,NMBLK$
18          .MCALL  FDRC$A,FDAT$A,FDOP$A
19          .MCALL  OPEN$W,GET$,PUT$,CLOSE$
20          .MCALL  OPEN$R
21
22          .NLIST  BEX                 ; Suppress ASCII
23  RSIZ    = 30.                       ; Record size (bytes)
24  IOST:   .BLKW   2                   ; QIO status block
25  PRINT:  QIOW$   IO.WVB,5,1,,,,<OBUFF,0,40>
26  BUFF:   .BLKB   RSIZ                ; User record buffer
27  OBUFF:  .BLKB   80.                 ; Output buffer for
28                                      ;  error messages
29  ARG:    .BLKW   1                   ; Argument block for
30                                      ;  $EDMSG
31  EFDQIO: .ASCIZ  /DIRECTIVE ERROR ON QIO. ERROR CODE = %D./
32  EFIQIO: .ASCIZ  ?I/O ERROR ON QIO. ERROR CODE = %D.?
33  EFCDIR: .ASCIZ  /FCS DIRECTIVE ERROR. ERROR CODE = %D./
34  EFCSIO: .ASCIZ  ?FCS I/O ERROR CODE. ERROR CODE = %D.?
35
36          .EVEN
37          .LIST   BEX                 ; Show offsets
38
39          FSRSZ$  2                   ; 2 files for record I/O
40  ;                                                 ;;EX
41
42  FDB:    FDBDF$                      ; File descriptor block
43          FDRC$A  ,BUFF,RSIZ          ; User buffer and size
44          FDAT$A  R.FIX,FD.CR,RSIZ ; Fixed length records,
45                                      ; implied <CR><LF>
46          FDOP$A  1,,FILE             ; use LUN 1
47  FILE:   NMBLK$  FIXED,ASC           ; FIXED.ASC
```

SOLUTION

```
49    ; FDB for TI:                                                    ;;EX
50    FDBI:   FDBDF$                                                   ;;EX
51            FDRC$A  ,BUFF,30.       ; URB addr and size,           ;;EX
52                                    ;  defaults to                 ;;EX
53                                    ;  sequential access           ;;EX
54            FDAT$A  R.VAR,FD.CR     ; Fixed length records,        ;;EX
55                                    ;  implied <CR><LF>             ;;EX
56            FDOP$A  2,DSPTI         ; Use LUN 2, dataset
57                                    ;  descriptor at DSPTI          ;;EX
58    DSPTI:  .WORD   LDEV,DEV        ; Device                       ;;EX
59            .WORD   0,0             ; UIC - not needed for         ;;EX
60                                    ;  TI:                         ;;EX
61            .WORD   0,0             ; File Name - not              ;;EX
62                                    ;  needed for TI:               ;;EX
63    DEV:    .ASCII  /TI:/           ; ASCII device                 ;;EX
64            LDEV=.-DEV              ;                              ;;EX
65            .EVEN                   ;                              ;;EX
66            .ENABL  LSB
67
68    START:  OPEN$W  #FDB,,,,,,ERR1  ; OPEN; if open fails,
69                                    ;  CALL ERR1
70            OPEN$R  #FDBI,,,,,,ERR1 ; OPEN "file" on TI:           ;;EX
71                                    ;  for read                    ;;EX
72
73    10$:    MOV     #RSIZ,R1        ; Size of URB
74            MOV     #BUFF,R2        ; Addr of URB
75    20$:    MOVB    #' ,(R2)+       ; Blank fill record
76            SOB     R1,20$          ;  so no garbage fill
77            GET$    #FDBI           ; Get record from TI:          ;;EX
78            BCC     OKIO            ; Branch on GET$ ok            ;;EX
79            TSTB    F.ERR+1(R0)     ; I/O error or                 ;;EX
80                                    ;  directive error?            ;;EX
81            BNE     DIRERR          ; Branch on directive          ;;EX
82                                    ;   error                      ;;EX
83    ; Stay here for I/O error. Check for ^Z.                        ;;EX
84    IOERR:  CMPB    #IE.EOF,F.ERR(R0) ; Check for EOF              ;;EX
85            BEQ     EXIT            ; If EQ, close and exit        ;;EX
86            BR      IO              ; It is an I/O error,          ;;EX
87                                    ;  so display error            ;;EX
88                                    ;  message and exit             ;;EX
89    OKIO:   PUT$    #FDB,,,ERR2     ; Write next record
90            BR      10$             ; Get next record
91
92    EXIT:   CLOSE$  #FDB            ; Close file
93            BCS     ERR3            ; Branch on FCS error
94            CLOSE$  #FDBI           ; Close "file" at TI:          ;;EX
95            BCS     ERR4            ; Branch on FCS error
96            EXST$C  EX$SUC          ; Exit with status of 1
```

SOLUTION

```
 98   ; Error  Processing
 99   ERR1:
100   ERR2:
101   ERR3:
102   ERR4:   TSTB    F.ERR+1(R0)       ; Directive error or I/O
103                                     ;  error
104           BEQ     IO                ; Branch on I/O error
105   DIRERR: MOV     #EFCDIR,R1        ; Set up for $EDMSG,  ;;EX
106                                     ;  directive error
107           BR      FINSET            ; Branch to finish setup
108   IO:     MOV     #EFCSIO,R1        ; Set up for $EDMSG, I/O
109                                     ;  error
110   FINSET: MOVB    F.ERR(R0),R0      ; FCS error code
111           MOV     R0,ARG            ;  is sign extended and
112           MOV     #ARG,R2           ;  placed in arg block
113                                     ; $EDMSG argument block
114   SHOERR: MOV     #OBUFF,R0         ; Output buffer
115           CALL    $EDMSG            ; Format error message
116           MOV     R1,PRINT+Q.IOPL+2 ; Size of message
117           DIR$    #PRINT            ; Print error message
118           CLOSE$  #FDB              ; Close file
119           CLOSE$  #FDBI             ; Close "file" at TI: ;;EX
120           EXST$C  EX$ERR            ; Exit with status of 2
121           .END    START
```

SOLUTION

4.

```
        .TITLE  LEX104
        .IDENT  /01/
        .ENABL  LC                      ; Enable lower case
;
; File LEX104.MAC
;
; This program opens the file FIXED.ASC and updates
; records in the file using random access. The orisinal
; file was created using CREFXA

        .MCALL  FDBDF$,FDAT$A,FDRC$A,FDOP$A,OPEN$U
        .MCALL  EXIT$S,QIOW$C,QIOW$,QIOW$S,PUT$R
        .MCALL  CLOSE$,FCSMC$

        FCSMC$                          ; Get most of the FCS
                                        ;  macros (FCSMC$ has
                                        ;  .MCALLs for many FCS
                                        ;  macros

        .NLIST  BEX

RSIZ    =30.                            ; Record size (in bytes)
IOST:   .BLKW   2                       ; I/O status block
PRINT:  QIOW$   IO.WVB,5,1,,,,<OUT,0,40>
BUFF:   .BLKB   RSIZ                    ; user record buffer
EMESD:  .ASCIZ  /FCS DIRECTIVE ERROR. CODE = %D./
EMESI:  .ASCIZ  'FCS I/O ERROR. CODE = %D.'
OUT:    .BLKB   100.                    ; Output message buffer
BUFF1:  .ASCII  /THAT'S ALL FOLKS!/ ; Message on success
                                        ;  completion
        LEN1=.-BUFF1
ERRMSG: .BLKB   100.                    ; Error message buffer
MSGERF: .ASCIZ  /DIRECTIVE ERROR, CODE = %D/
CNVER:  .ASCII  /CONVERSION ERROR ON RECORD NUMBER/
        LCNVER=.-CNVER
INPT:   .ASCII  /ENTER RECORD NUMBER AND TEXT:/
        LINPT=.-INPT
        .EVEN

        FSRSZ$  1                       ; 1 file open for record
                                        ; I/O
; FDB for file
FDB:    FDBDF$                          ;FDB
;FD.INS is needed to keep the EOF mark where it is
        FDRC$A  FD.INS!FD.RAN,BUFF,RSIZ ; Random mode, URB
                                        ;  addrs and size
        FDOP$A  1,,DFNB                 ; Use LUN 1, default
                                        ;  filename block
DFNB:   NMBLK$  FIXED,ASC               ; Default name FIXED.ASC
```

SOLUTION

```
50
51          .ENABL  LSB                 ; Allow local symbols
52                                      ;  to cross Psect
53                                      ;  boundaries
54
55  START:  OPEN$U  #FDB,,,,,,ERR1     ; Open file for update
56                                      ;  (includes extend)
57  ; Clear buffer to all blanks each time
58  10$:    MOV     #RSIZ,R1            ; Record size
59          MOV     #BUFF,R2            ; R2 => buffer
60  20$:    MOVB    #' ,(R2)+           ; Move in a blank
61          SOB     R1,20$              ; Continue until done
62
63          QIOW$C  IO.RPR,5,1,,IOST,,<BUFF,RSIZ,,INPT,LINPT,'$>
64                                      ; Prompt and get input
65          CMPB    #IE.EOF,IOST        ; Check for ^Z
66          BEQ     EXIT                ; If ^Z, exit
67          MOV     #BUFF,R0            ; Set up to convert
68          CALL    $CDTB               ;  record # to binary
69  ; Check for good conversion, character after # is
70  ; returned in R2 (it should be a ",")
71          CMPB    #',,R2              ; Is it a comma
72          BEQ     GOOD                ; Branch on good
73                                      ;  conversion
74          QIOW$C  IO.WVB,5,1,,,,<CNVER,LCNVER,40>
75                                      ; Display error message
76          BCS     ERR4                ; Branch on directive
77                                      ;  error
78          BR      10$                 ; Get next input
79  GOOD:   PUT$R   #FDB,,,R1,,ERR2     ; Write record to output
80                                      ;  file
81          BR      10$                 ; Get next input
82  ; Close file, display message, and exit
83  EXIT:   CLOSE$  #FDB,ERR3           ; Close file
84          QIOW$C  IO.WVB,5,1,,,,<BUFF1,LEN1,40> ;Write
85                                      ; message to operator
86          BCS     ERR4                ; Branch on error
87          EXIT$S
88
89  ERR1:
90  ERR2:   CLOSE$  #FDB,ERR3           ; Close file
91  ERR3:   MOVB    F.ERR(R0),R0        ; Move FCS error code
92          MOV     R0,IOST             ;  to argument block
93                                      ;  for $EDMSG
94          MOV     #IOST,R2            ; Set up for $EDMSG
95          TSTB    F.ERR+1(R0)         ; I/O or directive error
96          BEQ     IOERR               ; Branch on I/O error
97          MOV     #EMESD,R1           ; Set up for dir error
98                                      ;  message
99          BR      COMME               ; Branch to common code
100 IOERR:  MOV     #EMESI,R1           ; Set up for I/O error
101                                     ;  message
```

SOLUTION

```
102   COMME:  MOV     #OUT,R0          ; Set up for $EDMSG
103           CALL    $EDMSG           ; Edit error message
104           MOV     R1,PRINT+Q.IOPL+2 ; Length of error
105                                    ;  message
106           DIR$    #PRINT           ; Display error message
107           EXIT$S                   ; EXIT
108
109   ; Here for directive error on QIO
110   ERR4:   MOV     #ERRMSG,R0       ; Set up for $EDMSG
111           MOV     #MSGERF,R1       ;
112           MOV     #$DSW,R2         ;
113           CALL    $EDMSG           ; Edit message
114           QIOW$S  #IO.WVB,#5,#1,,,,<#ERRMSG,R1,#40>
115                                    ; Display message
116           EXIT$S                   ; Exit
117           .END    START
```

```
  1           PROGRAM LEX104
  2   C
  3   C File LEX104.FTN
  4   C
  5   C This task updates records in the file FIXED.ASC using
  6   C direct access formatted writes. The original file was
  7   C created using CREFXA.
  8   C
  9   C Direct access formatted writes are available in
 10   C FORTRAN IV-PLUS and FORTRAN-77 only
 11   C
 12           BYTE REC (30)
 13   C
 14   C Open file
 15           OPEN (UNIT=2,NAME='FIXED.ASC',ACCESS='DIRECT',
 16           1 TYPE='OLD',FORM='FORMATTED')
 17   C Place blanks in buffer
 18   10      DO 15 J=1,30
 19           REC(J)=' '
 20   15      CONTINUE
 21   C Read record from terminal
 22           WRITE (5,20)
 23   20      FORMAT ('$ENTER RECORD NUMBER AND TEXT: ')
 24           READ (5,50,END=900) REC
 25   50      FORMAT (64A1)
 26   C Convert record number to inteser format
 27           DECODE  (2,60,REC) NREC
 28   60      FORMAT (I2)
 29   C Write record to disk
 30           WRITE (2'NREC,80) REC
 31   80      FORMAT (30A1)
 32   100     GOTO 10
 33   C ^Z input, close file and exit
 34   900     CLOSE (UNIT=2)
 35           CALL EXIT
 36           END
```

SOLUTION

5.

```
        .TITLE  BLOCK2
        .IDENT  /01/
        .ENABL  LC              ; Enable lower case
;+
; File LEX105.MAC                       ;;EX
;
; Modified to work on 2 virtual blocks at a time ;;EX
;
; **-BLOCK2 prompts at TI: for a virtual block number
; and then reads and displays that block of "BLOCK.ASC"
;-

        .MCALL  QIOW$,DIR$,QIOW$S,EXST$S
        .MCALL  FDBDF$,FDRC$A,FDBK$A,FDOP$A,NMBLK$
        .MCALL  FSRSZ$,OPEN$R,READ$,WAIT$,CLOSE$

        .SBTTL  MESSAGES
        .NLIST  BEX
CR      = 15
LF      = 12
MES1:   .ASCII  /FIRST VIRTUAL BLOCK: /             ;;EX
LEN1    = . - MES1
MES2:   .ASCII  <CR><LF>/HERE ARE THE BLOCKS : /<CR><LF>
;                                                   ;;EX
LEN2    = . - MES2
MES3I:  .ASCIZ  'I/O ERROR FROM OPEN$R, CODE = %D.'
MES3D:  .ASCIZ  /DIRECTIVE ERROR FROM OPEN$R, CODE = %D./
MES4I:  .ASCIZ  'I/O ERROR FROM READ$, CODE = %D.'
MES4D:  .ASCIZ  /DIRECTIVE ERROR FROM READ$, CODE - %D./
MES5I:  .ASCIZ  'I/O ERROR FROM WAIT$, CODE = %D.'
MES5D:  .ASCIZ  /DIRECTIVE ERROR FROM WAIT$, CODE = %D./
BUFF:   .BLKB   80.             ; STORE RESPONSE HERE

        .LIST   BEX
        .EVEN
        .SBTTL  LOCAL STORAGE

        FSRSZ$  0               ; NO FSR BUFFER NEEDED
                                ; FOR BLOCK I/O

FDB:    FDBDF$                  ; FDB FOR INPUT FILE
        FDRC$A  FD.RWM          ; READ/WRITE MODE
        FDBK$A  BLOCK,1024.,,1,IOSB ; EF 1, BUFFER ADR,;;EX
                                ; SIZE
        FDOP$A  1,,FILE         ; LUN 1, DFNB
FILE:   NMBLK$  BLOCK,ASC       ; NAME IS BLOCK.ASC

VBN:    .WORD   0,1             ; DEFAULT VBN
BLOCK:  .BLKW   512.            ; BLOCK BUFFER      ;;EX
IOSB:   .BLKW   2
```

SOLUTION

```
51
52   PROMPT: QIOW$   IO.RPR,5,1,,IOSB,,<BUFF,6,,MES1,LEN1,'$>
53                                   ; Prompt and get VB #
54   DONE:   QIOW$   IO.WVB,5,1,,,,<MES2,LEN2,40> ; Done
55                                                ;  message
56   DUMP:   QIOW$   IO.WVB,5,1,,,,<0,64.,40> ; Display of VB
57
58           .SBTTL  MAINLINE CODE
59
60   START:
61           OPEN$R  #FDB,,,,,,ERR1  ; Open file
62           DIR$    #PROMPT         ; Ask for a VBN
63           MOV     IOSB+2,R0       ; Put null at end
64           CLRB    BUFF(R0)        ;  of digit string
65           MOV     #BUFF,R0        ; R0 => VBN
66           CALL    $COTB           ; Convert to binary
67           MOV     R1,VBN+2        ; Store as low VBN
68           READ$   #FDB,,,#VBN,,,,ERR2 ; Read in the block
69           WAIT$   ,,,ERR3         ; Wait until done
70           DIR$    #DONE           ; Tell them I/O is done
71
72   ; Now dump 16. lines of 64. characters each          ;;EX
73
74           MOV     #BLOCK,R0       ; R0 => 1st line to dump
75           MOV     #16.,R1         ; # of lines to dump  ;;EX
76   1$:
77           MOV     R0,DUMP+Q.IOPL  ; Addr of current line
78           DIR$    #DUMP           ; Dump it
79           ADD     #64.,R0         ; Point at next line
80           SOB     R1,1$           ; Dump all 8. lines
81
82   ; Now we exit with status = EX$SUC
83
84           MOV     #EX$SUC,R5      ; Put status in R5
85           BR      EXIT            ; And then exit
86
87           .SBTTL  ERROR ROUTINES
88
89   ERR1:
90           TSTB    F.ERR+1(R0)     ; I/O or directive error?
91           BEQ     IOERR1          ; Branch on I/O error
92           MOV     #MES3D,R1       ; => Dir error message 3
93           BR      FCSERR          ; Branch to common code
94   IOERR1: MOV     #MES3I,R1       ; => I/O error message 3
95           BR      FCSERR          ; Branch to common code
96   ERR2:
97           TSTB    F.ERR+1(R0)     ; I/O or directive error?
98           BEQ     IOERR2          ; Branch on I/O error
99           MOV     #MES4D,R1       ; => Dir error message 4
100          BR      FCSERR          ; Branch to common code
```

SOLUTION

```
101    IOERR2: MOV     #MES4I,R1          ; => I/O error message 4
102            BR      FCSERR             ; Branch to common code
103    ERR3:
104            TSTB    F.ERR+1(R0)        ; I/O or directive error
105            BEQ     IOERR3             ; Branch on I/O error
106            MOV     #MES5D,R1          ; => Dir error message 5
107            BR      FCSERR             ; Branch to common code
108    IOERR3: MOV     #MES5I,R1          ; => I/O error message 5
109                                       ; FALL INTO COMMON CODE
110    FCSERR:
111            MOVB    F.ERR(R0),R2       ; Sign extend error code
112            MOV     R2,IOSB            ;  and move into IOSB
113            MOV     #EX$ERR,R5         ; Exit status in R5
114    FORMAT:
115            MOV     #IOSB,R2           ; Set up for $EDMSG
116            MOV     #BUFF,R0           ;
117            CALL    $EDMSG             ;
118            QIOW$S  #IO.WVB,#5,#1,,,,<#BUFF,R1,#40> ; Display
119                                       ;  message
120    EXIT:
121            CLOSE$  #FDB               ; Close the file
122            EXST$S  R5                 ; Exit with status
123            .END    START
```

File Control Services

SOLUTION

```
1           .TITLE  CSI
2           .IDENT  /01/
3           .ENABL  LC                  ; Enable lower case
4   ;
5   ; File LEX106.MAC
6   ;
7   ; Modified to actually delete or display the file ;;EX
8   ;
9   ; CSI illustrates the use of the command string
10  ; interpreter. This task accepts a command line from the
11  ; terminal in the form:
12  ;
13  ;       dev:[x,y]filename.filetype;version/switch
14  ;
15  ; where switch can be:
16  ;                           DE - Delete file
17  ;                           DI:N - Display N copies of file
18
19          .MCALL  GCMLB$,GCML$,CSI$,CSI$1,CSI$2
20          .MCALL  CSI$SV,CSI$SW,CSI$ND
21          .MCALL  FSRSZ$,FDBDF$,FDRC$A,FDOP$A,FINIT$
22          .MCALL  QIOW$S,QIOW$,DIR$,EXIT$S
23          .MCALL  DELET$,OPEN$R,OPEN$W,GET$,PUT$,CLOSE$
24
25          .NLIST  BEX
26  ;       LOCAL DATA
27  TYPE1:  QIOW$   IO.WVB,5,1,,,,<ERR1,SIZ1,40>
28  TYPE2:  QIOW$   IO.WVB,5,1,,,,<ERR2,SIZ2,40>
29  TYPE3:  QIOW$   IO.WVB,5,1,,,,<ERR3,SIZ3,40>
30  TYPE4:  QIOW$   IO.WVB,5,1,,,,<BUFF,,40>                ;;EX
31  ERR1:   .ASCII  /GET COMMAND LINE ERROR/
32          SIZ1=.-ERR1
33  ERR2:   .ASCII  /CSI ERROR. ILLEGAL COMMAND/
34          SIZ2=.-ERR2
35  ERR3:   .ASCII  /CSI ERROR. FILE SPEC ERROR/
36          SIZ3=.-ERR3
37  BUFF:   .BLKB   100.                ; Output text buffer
38  TBUFF:  .BLKB   132.                ; Transfer buffer
39  FMT:    .ASCIZ  /YOU HAVE REQUESTED A %7A JOB/
40  FMTERD: .ASCII  /FCS DIRECTIVE ERROR ON %7A./       ;;EX
41          .ASCIZ  / CODE = %D./                       ;;EX
42  FMTERI: .ASCIZ  ?FCS I/O ERROR ON %7A. CODE = %D.?;;EX
43          .EVEN
44  DATA:   .BLKW   2                   ; Argument block  ;;EX
45  DELTXT: .ASCII  /DELETE/<0>         ; ASCII text
46  DITXT:  .ASCII  /DISPLAY/
47  NOTXT:  .ASCII  /NOTHING/
48  CLTXT:  .ASCII  /CLOSE/<0><0>       ; For close       ;;EX
49          .EVEN
```

SOLUTION

```
 51            CSI$                          ; Define CSI offsets
 52    CBLK:   .BLKB   C.SIZE                ; allocate CSI storage
 53            .EVEN
 54
 55            DEMSK = 1                     ; Delete mask
 56            DIMSK = 2                     ; Display mask
 57    SWTBL:                                ; Switch descriptor table
 58            CSI$SW  DE,DEMSK              ; Delete switch = DE
 59            CSI$SW  DI,DIMSK,,,,NUM       ; Display switch = DI,
 60                                          ;  also allow DI:N
 61            CSI$ND                        ; End of switch table
 62
 63            CSI$SV  OCTAL,COPY,2,NUM      ; Value N for /DI:N is
 64                                          ;  in octal and will
 65                                          ;  be stored in COPY
 66            CSI$ND                        ; End of switch value
 67                                          ;  table
 68    ;GET COMMAND LINE BLOCK DEFINITIONS
 69
 70            FSRSZ$  3                     ; GCML uses record I/O;;EX
 71
 72    GBLK:   GCMLB$  ,CSI,,5               ; Prompt with 'CSI' on
 73                                          ;  LUN 5
 74    FDB:    FDBDF$                        ; FDB for file to delete
 75                                          ;  or display.
 76            FDRC$A  ,TBUFF,132.           ; URB AT TBUFF, length
 77                                          ;  132.
 78            FDOP$A  1,CBLK+C.DSDS         ; LUN 1, dataset
 79                                          ; descriptor from CSI
 80
 81    ; NOTE: Need a 2nd FDB for display
 82
 83    FDBO:   FDBDF$                        ; FDB for output to TI:;;EX
 84            FDAT$A  R.VAR,FD.CR           ; Var length records, ;;EX
 85                                          ;  list format        ;;EX
 86            FDRC$A  ,TBUFF,132.           ; URB at TBUFF, length;;EX
 87                                          ;  132.               ;;EX
 88            FDOP$A  2,DSPTO               ; LUN 2, dataset      ;;EX
 89                                          ; descriptor at DSPTO ;;EX
 90    DSPTO:  .WORD   LDEV,DEV              ; Dataset descriptor  ;;EX
 91            .WORD   0,0                   ;  for TI:. No UIC or ;;EX
 92            .WORD   0,0                   ;  name needed.       ;;EX
 93    DEV:    .ASCII  /TI:/                 ;                     ;;EX
 94            LDEV=.-DEV                    ;                     ;;EX
 95
 96            .EVEN
 97    JMPTBL: .WORD   NONE,DELETE,DISPLY    ; Jump table for
 98                                          ;  subroutines depending
 99                                          ;  on switches
100    COPY:   .WORD   1                     ; Value for N in /DI:N
```

SOLUTION

```
101             .ENABLE LSB
102
103     START:  FINIT$                      ; Initialize FCS, this
104                                         ;  is normally done with
105                                         ;  an OPEN statement.
106                                         ;  For delete we do not
107                                         ;  need an open statement.
108     NEXT:   GCML$   #GBLK               ; Prompt and get command
109             BCC     10$                 ; Branch if command OK
110     ; Check for ^Z. If ^Z, exit.
111             CMPB    #GE.EOF,GBLK+G.ERR ; Is it ^Z?
112             BNE     REALER              ; Branch on other error
113             EXIT$S                      ; Exit
114     REALER: DIR$    #TYPE1              ; Display error text for
115                                         ;  set command line error
116             EXIT$S                      ; Exit
117     ; Parse input for illegal characters
118     10$:    CSI$1   #CBLK,GBLK+G.CMLD+2,GBLK+G.CMLD ; Format
119                                         ; is CSI addr, addr of
120                                         ; command, length of
121                                         ; command
122             BCC     20$                 ; Branch on OK command
123             DIR$    #TYPE2              ; Display error text for
124                                         ;  illegal command
125             EXIT$S                      ; Exit
126
127     ; Create a dataset descriptor from the file specification
128
129     20$:    CSI$2   #CBLK,OUTPUT,#SWTBL ; Expect output file
130                                         ;  spec
131             BCC     30$                 ; Branch on file spec OK
132             DIR$    #TYPE3              ; Display text for file
133                                         ;  spec error
134             EXIT$S                      ; Exit
135
136     ; Call the appropriate subroutine
137
138     30$:    MOV     #FDB,R0             ; Address of file
139                                         ;  descriptor
140             MOV     CBLK+C.MKW1,R1      ; Mask value = 0, 1, or 2
141                                         ;
142             ASL     R1                  ; Double for word offset
143                                         ;  into Jump table
144             CALL    @JMPTBL(R1)         ; Call the subroutine
145             BR      NEXT                ; Get next command line
146
147     ; Subroutine NONE, entered if no switches specified
148
149     NONE:   MOV     #NOTXT,DATA         ; Set up for output of
150                                         ;  message
```

SOLUTION

```
151             CALL    OUTMS           ; Call OUTMS, as a      ;;EX
152                                     ; subroutine            ;;EX
153             RETURN                  ; Return                ;;EX
154
155     ; Common display message code - a subroutine since it ;;EX
156     ; is not a common return point                         ;;EX
157
158     OUTMS:  MOV     #BUFF,R0        ; Set up for $EDMSG
159             MOV     #FMT,R1         ;
160             MOV     #DATA,R2        ;
161             CALL    $EDMSG          ; Edit message
162             QIOW$S  #IO.WVB,#5,#1,,,,<#BUFF,R1,#40> ; Display
163             RETURN                  ; Return
164
165     ; Subroutine DELETE
166     ;
167     ; ***WARNING - THE HIGHEST VERSION NUMBER OF THE FILE  ***
168     ; ***WILL BE DELETED IF NO VERSION NUMBER IS SPECIFIED ***
169
170     DELETE: MOV     #DELTXT,DATA    ; Set up for output of
171                                     ;  message
172             CALL    OUTMS           ; Call display          ;;EX
173                                     ;  subroutine          ;;EX
174             DELET$  #FDB,ERRD       ; Delete file           ;;EX
175             RETURN                  ; Return
176     ; Delete error code
177     ERRD:   MOVB    F.ERR(R0),R5    ; Extend sign on error;;EX
178             MOV     R5,DATA+2       ;  and move to ars block;;EX
179             MOV     #DELTXT,DATA    ; Move pointer to delete;;EX
180                                     ;  text                 ;;EX
181     COMME:  TSTB    F.ERR+1(R0)     ; Check for directive   ;;EX
182                                     ;  error or I/O error  ;;EX
183             BEQ     IOERR           ; Branch on I/O error   ;;EX
184             MOV     #FMTERD,R1      ; Get format string     ;;EX
185             BR      DISPER          ; Branch to common      ;;EX
186                                     ;  error display code  ;;EX
187     IOERR:  MOV     #FMTERI,R1      ; Get format string     ;;EX
188     DISPER: MOV     #BUFF,R0        ; Set up for $EDMSG     ;;EX
189             MOV     #DATA,R2        ;                       ;;EX
190             CALL    $EDMSG          ; Edit message          ;;EX
191             MOV     R1,TYPE4+Q.IOPL+2 ; Size of message    ;;EX
192             DIR$    #TYPE4          ; Display message       ;;EX
193             EXIT$S                  ; Exit                  ;;EX
194
195     ; Subroutine DISPLY - Just display a message
196
197     DISPLY: CALL    $SAVAL          ; Save all registers
198             MOV     #DITXT,DATA     ; Set up for output of
199                                     ;  message
200             CALL    OUTMS           ; Branch to common
201                                     ;  display code
```

SOLUTION

```
202             OPEN$R  #FDB,,,,,,ERRE  ; Open file for input
203             OPEN$W  #FDBO,,,,,,ERRE ; Open TI: for output
204             MOV     COPY,R4         ; Number of copies to
205                                     ;  R4
206             MOV     #FDB,R0         ; Addr of FDB of input
207                                     ;  file
208             CALL    .MARK           ; Save pointers to
209                                     ;  first record for
210                                     ;  resetting. Pointers
211                                     ;  are returned in R1,
212                                     ;  R2,R3
213             BCS     ERRE            ; Branch on error
214     GET:    GET$    #FDB            ; Get record from file
215             BCS     CHECK           ; Branch on error
216     ; Stay here if OK set - output record to TI:
217             MOV     F.NRBD(R0),FDBO+F.NRBD ; Move length of
218                                     ; record to FDBO
219             PUT$    #FDBO,,,ERRE    ; Display record at TI:
220             BR      GET             ; Get next record
221     ; Error code
222     CHECK:  CMPB    #IE.EOF,F.ERR(R0) ; Check for EOF
223             BNE     ERRE            ; Branch if not
224             DEC     R4              ; Decrement copy counter
225             BNE     AGAIN           ; Branch if more to do
226             MOV     #1,COPY         ; Reset number of copies
227             CLOSE$  #FDB,ERRC       ; Close file
228             CLOSE$  #FDBO,ERRC      ; Close TI:
229             RETURN
230     ; More copies to do - reset pointers to start of input
231     ; file and repeat
232     AGAIN:  CALL    .POINT          ; R1,R2,R3 are still set
233             BCC     GET             ; Display next copy
234     ; Here for errors on PUT$s, GET$s, and .POINTs
235     ERRE:   MOVB    F.ERR(R0),R5    ; Extend sign and move
236             MOV     R5,DATA+2       ;  error code to DATA
237             MOV     #DITXT,DATA     ; Move display function
238                                     ;  for display
239             CLOSE$  #FDB,ERRC       ; Close files
240             CLOSE$  #FDBO,ERRC      ;
241             BR      COMME           ; Branch to common error
242                                     ;  code
243     ; Here for errors on close
244     ERRC:   MOVB    F.ERR(R0),R5    ; Extend sign and move
245             MOV     R5,DATA+2       ;  error code to DATA
246             MOV     #CLTXT,DATA     ; Move close text for
247                                     ;  display
248             JMP     COMME           ; Jump to common error
249                                     ;  code
250             .END    START
```