THE mini• tasker

DECUS
RT-11 SIG NEWSLETTER

OCTOBER 1982 VOL 8-4

Contributions to the newsletter should be sent to:

Ken Demers
MS-44
United Technologies Research Center
East Hartford, Conn. 06108
(203) 727-7527 or 7240

Other communications can be sent to:

John T. Rasted
JTR Associates
58 Rasted Lane
Meriden, Conn. 06450
(203) 634-1632

or

RT-11 SIG
C/O DECUS
One Iron Way
MR2-3/E55
Marlboro, Mass. 01752
(617) 467-4141

TABLE OF CONTENTS

How To Reach Them - - - - - - - - - - - - - - - - 2
User Input - - - - - - - - - - - - - - - - - - 4
RT-11 Odds & Evens - - - - - - - - - - - - - 4
Miscellaneous Programs - - - - - - - - - - - 7
How to Manage a Large Disk - - - - - - - - - 22
Data Base Management Information - - - - - - - 24
Quick RT-11 Quiz - - - - - - - - - - - - - 25
User Requests
Need for an Optical System Program - - - - - - 28
User Responses
Visicalc Type Product - - - - - - - - - - - - 28
Upcoming Symposium Information
Fall Symposium Preview - - - - - - - - - - - 29
SPR's
XM Monitor Problem - - - - - - - - - - - - - 30
Dec Input
RT-11 Version 5 - - - - - - - - - - - - - - 31

HOW TO REACH THEM

JOHN T. RASTED — RT-11 SIG Chairman
JTR Associates
58 Rasted Lane
Meriden, Ct 06450
(203) 634-1632

KEN DEMERS — Newsletter Editor, DECNET Contact
MS-44
United Tech. Research Ctr.
Silver Lane
East Hartford, Ct 06108
(203) 727-7527

MARK BARTELT — Structured Languages Contact
Hospital for Sick Children
Research Development Corp.
555 University Ave.
Toronto, Ontario
Canada M56 1X8

DOUG BOHRER — APL Contact
First National Bank of Chicago
One First National Plaza
Chicago, Ill 60670
(312) 732-8785

NICK BOURGEOIS / 1738 — TSX Contact, MACRO Contact
Sandia Laboratories
P.O.Box 5800
Albuquerque, NM 87185
(505) 844-8088

JOHN CROWELL — TECO Contact, FART Contact
CROW4ELL Ltd.
145 Andanada
Los Alamos, NM 87544
(505) 662-3893

JAMES R. CUTLER — Standards Coordinator
Intel Corporation
2625 Walsh Ave.
Santa Clara, CA 95051
(408) 496-8084

ART HERMES — RT/RSX Contact
Seismograph Service Corp.
P.O.Box 1590
Tulsa, OK 74102

SHIRLEY HOOPER — RT DECUS Library Contact
Bausch & Lomb
Instrument & Systems Division
9545 Wentworth Street
Sunland, Ca 91040
(213) 352-6011

JOSEPH LACHMAN — Tape Copy Coordinator
Lachman Associates, Inc.
645 Blackhawk Drive
Westmont, Il 60559
(312) 986-8840

CARL LOWENSTEIN — RT-11 Hardware Contact
Univ. of Calif, San Diego
Marine Physical Lab. of the
Scripps Inst. of Oceanography
San Diego, Ca 92152
(714) 294-3678

SUSAN S. RASTED — FMS-11 Contact
Software Dynamics Inc.
1000 Yale Ave.
Wallingford, Ct 06492
(203) 265-2226

NED RHODES — Symposia Coordinator
E-Systems
Melpar Division
7700 Arlington Boulevard
Falls Church, Va 22046
(703) 560-5000

MARILYN RUNYON — Wish List
39 Locust Point Rd.
Locust, N.J. 07760

ED STEVENS — BASIC Contact
Emda Inc.
133 N. San Gabriel Blvd.
Suite 204 Lamanda Park
Pasadena, California 91107
(213) 795-5991

J. W. TIPPIE — CAMAC Contact
Argonne National Labs
9700 S. Cass Ave
Argonne, Ill 60439

RON TRELLUE — FORTRAN Contact; RT-11 LUG Contact
Division 7523
Sandia Labs
Albuquerque, NM 87185
(505) 844-0155

USER INPUT

RT-11 ODDS AND EVENS

Ernst Lopes Cardozo
Westvries Systems BV, Ravenswaay,
The Netherlands

After working with RT-11 for about 2 years it's time for some comments. Let me start saying that as a whole RT-11 is just a fantastic operating system, especially on the outside, ie. if you just use it as is. Nevertheless, there are some odd things in RT-11 that I like to mention, if only to have a chance for improvement. Most of these points I ran into during the development of DRTS-11, a RT-11 based multi-processor operating system; a bunch of LSI's are hooked to one extra LSI that caries peripherals. All the others (user processors) are just processor plus memory machines. The link is made with the Qnector, a 250 Kbyte DMA processor link. The user processors run RT-11, with the standard drivers replaced by pseudo drivers that access the peripherals on the central IO processor.

To start with something simple; why do the RT-11 lineprinter handlers (LP.SYS and LS.SYS) not properly close a file with a formfeed ? This may seem just a matter of taste, but in fact the present situation has resulted in some ugly contructions in RT-11's own CUSPs. PIP will, when copying a file, look at the device type of the output handler, and in case it is one of the printers, send an extra record with just CR-LF. This was done to make sure that the last line realy get's printed, even on some matrix printers that tend to keep the last line in an internal buffer. Now that seems to be a function of the device driver, NOT of a general purpose utility (device-independence, do you recall?). PIP opens the file with the actual file length (the length of the input file), but then sends 1 word too much. My transparant spooler has to allocate one word extra to avoid a file-full condition on normal PRINT commands. QUEUE (the spooler) even sends a CR-LF-FF-LF, just to make sure that the files are separated.

What should be done ? LP.SYS and LS.SYS should be handlers with so-called special directories, ie. drivers that set the LOOKUP, ENTER and CLOSE functions passed by the USR. Then the handler knows when the file is finished and can properly close it, for instance by sending a formfeed (and turning off the motor, or whatever fancy actions are necessary for THIS printer.

Lets pick another problem. You may have noticed that some programs malfunction when RUNed instead of R'ed (Adventure is an example). R loads a program in two phases. First it reads the first block of PROG.SAV in a buffer and moves individual words into low memory, thereby avoiding protected words (EMT vector, etc). Then it looks at location 50 to find the length of the save image, sets up the IO parameters and jumps in the resident monitor, which loads the program in a single IO transfer and starts it. Thus, all of the program image gets loaded from the .SAV file. RUN, however, takes a different approach. After loading the first block in the same way as R does, it looks at the program's bitmap. The bitmap, located in location 370-376 of the save image, contains a 1 bit for each block of the save file where the linker has stored any data. For programs that incorporate .BLKW's of more than 512 bytes (DIMENSION of more than 256 words) some

bits in the midle of the bitmap may be clear, because the linker say's it did not load anything there (that is not true - all such locations are set to zero (or some other value, see the /FILL:nnn option). RUN will NOT load blocks that have a zero bit in the bitmap. So if the program inadvertently relies on the fact that unused loactions are zero, it crashes (or worse - just gives wrong answers!) when RUN.

RUSH TO YOUR MACHINE NOW! and try to link your programs with /FILL:-1 and see what happens... This is a very dangerous situation indeed. And why all this ? Actually there is no fundmental reason for the difference between R and RUN. At the cost of a few extra locations in the resident monitor the RUN command can be dropped altogether, and R extended to take a device specification. That would allow us to run virtual jobs from other devices as well. For the time being I wrote a little program that examins a .SAV file and fills up the bitmap up to the last block in the file. (The difference between R and RUN is historical. It stems from the OS/8 system, designed in 1970 for the PDP8, that typically had 8K memory and two DECtapes. The resident monitor of OS/8 is just 256 words and therefore could not load directly from anything but the system disk (tape). There is much more in RT-11 that reminds of OS/8; the CSI still accepts "<" instead of "=" and the directory structure is almost identical.)

Some minor frustrations: DUP has an internal table that tell's it which devices have variable media size (eg. RL01/RL02, RX01/RX02). That again is device-dependent information that has no place in a general purpose program. Rather, DUP should check if the driver supports special function calls (there is a bit in the driver status word that tells you so) and if he does, try .SPFUN 373 (get device size). If the function fails, DUP knows that it has to rely on the standard length given in the drivers header, otherwise it got the actual length right from the lions mouth. This would make adding new devices much cleaner, and saves DEC publishing and republishing inacurate DUP patches.

Why have LP.SYS and LS.SYS different device types ? This created much trouble with FORTRAN, that would not recognize LS as a printer. But what program would ever have to know whether the printer is interfaced paralell or serially ? (IFF we should have different types of printers, then let's make a distinction between narrow and wide printers: that could be usefull for some programs in order to adjust their output format). All printers should be just printers, and the actual interface is of concern to the driver only. (On PDT machines, the printer looks like paralell from within, but is actually serial; a microprocessor does the XON/XOFF procedure and controls the READY bit in the interface accordingly).

Formatting is a similar case; presently the FORMAT program seems to know all about formatting and therefore is completly device-dependent. (FORMAT does not even look at the CSR/vector address in the driver !). A better way would be to have the formatting routine included in the driver (remember, thats the place where ALL device-dependent information should be concentrated). Don't worry, I'm not proposing to have all the fromat routines resident with the drivers. Rather, they should be included in the driver file (XX.SYS), just like the bootstrap code is. A pointer in the first block of the file would indicate where the format code would reside. The program FORMAT would load the code and execute it (passing the CSR and vector address from the driver part of the file). Again, adding a new device would be much cleaner - just write a driver and the device is fully integrated in the system.

Let me add some of my whishlist items. Some are realy weird...

I would like to have a programmed request that passes a file name to the monitor and makes the monitor chain to this program after .EXIT or ^C^C. This would allow me (and all of us) to write menu programs, that chain to various other programs, but always get control back. The same request with a null filename would reenable KMON.

Logical disks/Big disk support. At some point in time, RT-11 will really run out of its 16 bit blocknumbers used in the IO request. So internally we will have to go to two-word (or three-byte) block numbers. The idea would be to have a disk driver that can access 4 Tbyte (Tera-byte) drives, that are subdivided in a large number of logical disks of up to 2^16 blocks each. Logical disks should not be allocated contiguous, but rather use a coarse cluster/mapping mechanism.

If you look at a (big) disk subdivided into many logical disks, only one or two logical disks are beeing used at any moment. Typically a user occupies 50-100% more disk space during his session than remains after he has left the machine. If each of these logical disks are to have sufficient working space, you are loosing 1/3 of your disk. So logical disks should be like virtual memory, if you start using it, it's there. After your session, you run some nice CUSP that squeezes the logical disk (maybe deletes .BAK etc.) and then deallocates any clusters beyond the last file. Cluster size should be quite large (and an INIT parameter obtained from the handler) so that mapping overhead is minimal. On average, you're loosing half a cluster for each logical disk.

Essentially, we would have a two-level disk structure. A Master Directory holds the names, mapping pointers and password of each logical disk (why not call these ACCOUNTS anyhow?). It also contains the master bitmap. The MFD could be in a file of the first logical disk. This first logical disk should be created by INIT; its first cluster should always map to the first physical cluster on the disk. The HOME block of this logical disk can contain the bitmap. Note that the first cluster of a logical disk never gets deallocated (unless the logical disk is deleted). The mapping should be done by RMON (maybe in a kind of pseudo system driver, that calls the true (and replaceble) disk driver (cross-cluster transfers have to be split of course)).

My particular usage of these structures would be as follows; I'm running a local network (DRTS-11, Distributed RT System). It is star-shaped and based on very fast DMA links between LSI's. The host (IO processor) is running RT-11 also, and the satellites run from virtual disks, actually *.USR files on the RT-11 system. I would like to use the logical disks as user accounts, and would do the allocation/ deallocation of clusters on-the-fly, each time a user touches a new cluster, or logss off. By the way, this makes a much more efficient file systems than FILES-11: the overhead of clustering is only marginal (with large clusters) and the file open/close uses the simple and fast RT-11 structure! Each user sees a 32Mbyte virtual disk, yet actually occupies half a cluster of unused space. On realy big disks, cluster size could be as large as 1 Mbyte (so that 32 mapping pointers map the full 32Mbyte range), on smaller disks it could be as small as 128 blocks.

I would like to see some of the optional patches be implemented as SET options, eg. SET RUNDEV DL1:, SET INIT QUIET, SET CLOCK 50HZ/60HZ, etc.

Can't we have SET options on CUSPs?! Eg. SET MACRO PAGE=77 (change listing page size), SET FORTRAN FIS, etc. Implementation follows the same schema as the handler set options (probably using the same SYSMAC macros). Just allow SET to have a full device:file spec.

Handlers should be made 'upward compatible' with regard to system options (timeout and error logging). The .FETCH/LOAD code should look at the handler config bits to determine which pointers can be filled. Makes changing monitors simpler and avoids playing with the monitor suffix.

Some users want to do .GTLIN with completion routines; a simple sollution would be to let TT: (on input) process a single line only, and then zero-fill the buffer and return. .READ from TT: is very seldom used now, and if you think of it, it just makes little sense to read a keyboard block-wise. The proposed change is probably transparant to existing programs and is simple to implement. (This technique was introduced in OS/8 in version 3 (must be about 5 years back now).

And further, I repeat that Fortran IV PLUS should be made available to the RT-11 world.

```
        .NLIST
        .ENABL  LC
        .LIST   TTM
        .NLIST  BEX,TOC
        .LIST   TTM
        .LIST
        .TITLE  BITMAP.MAC 1-MAR-82
        .SBTTL  PREFACE
        .IDENT /V01.00/

;               Copyright (c) 1982
;               Westvries Systems B.V.
;               P.O.Box 438
;               4100 AK Culembors
;               The Netherlands
;               Tel. 03452 - 202
;
; This software is made available to the RT11 commu-
; nity free of charge. Title to and ownership of the
; software  shall  at all  times remain in Westvries
; Systems BV.  The information  in  this document is
; subject to change without notice and should not be
; construed as a commitment by Westvries Systems BV.
;
; Author:       E. Lopes Cardozo
;
; This program insures that the bitmap in the CCB of an
; RT11 .SAV file is contiguous, in order to make sure that
; all dataarea's are always initialised in the same way.
```



```
        .SBTTL  MACRO DEFINITIONS

;RT11 SYSTEM MACROS USED:
        .MCALL  .PRINT, .EXIT,  .CSIGE, .READW, .WRITW

        .SBTTL  Main program

START:  .CSIGEN #SPACE,#EXT             ;Process command string
        .READW  #AREA,#3,#BUF,#400,#0   ;Read CCB
        BCS     REDERR                  ;Read error ?
        MOV     BUF+50,R0               ;Get program high limit
        ADD     #777,R0                 ;Round up
        ASR     R0                      ;Devide by 1000
        BIC     #100000,R0              ;Don't sign-extend !
        ASR     R0                      ;
        ASR     R0                      ;
        ASR     R0                      ;
        ASR     R0                      ;
        ASR     R0                      ;
        ASR     R0                      ;
        ASR     R0                      ;
        ASR     R0                      ;Gives number of blocks to load
        MOV     #BUF+360,R1             ;Set pointer to bitmap
30$:    MOV     #200,R2                 ;Start with highest bit in byte
40$:    BISB    R2,(R1)                 ;Set a bit
        DEC     R0                      ;All blocks marked ?
        BEQ     50$                     ;Yes, quit
        ASR     R2                      ;No, shift the mask. Mask empty ?
        BNE     40$                     ;No, keep going
        INC     R1                      ;Yes, bump pointer to next byte
        BR      30$                     ; and continue there

50$:    .WRITW  #AREA,#3,#BUF,#400,#0   ;Rewrite the CCB
        JMP     START
        .EXIT                           ;

REDERR: .PRINT  #REDER                  ;Error reading CCB
        .EXIT                           ;

WRTERR: .PRINT  #WRER                   ;Error writing CCB
        .EXIT                           ;

        .SBTTL  DATA

EXT:    .RAD50  /SAVSAVSAVSAV/          ;Defualt file type is .SAV
AREA:   .BLKW   5
BUF:    .BLKW   400

REDER:  .ASCIZ  /?BITMAP-F-Error reading program file/
WRER:   .ASCIZ  /?BITMAP-F-Error writing program file/
HELLO:  .ASCIZ  /BITMAP V01.00/
        .EVEN

SPACE=.

        .END    START
```

```
;This is a patch for single-user Basic. It's purpose is to
;let Basic read its first line of input through a .GTLIN in
;order to allow the user to RUN a basic program from an
;indirect command line.
;
;The methode is to patch the .TTYINR request in Basic and defer
;it to a routine that initially reads a line through .GTLIN and
;then feeds BASIC the characters one by one. After the first line,
;the routine undoes the patch to let any further input run through
;.TTYNR's.
;
;Relink BASIC, using this module as the very first entry to LINK.
;This insures that the entrypoint of BASIC.SAV will be in this
;module.

        .MCALL  .GTLIN

        .GLOBL  BLOCKI, STARTI

        .PSECT  TRICK

BUFFER:
TRICK:: MOV     ADR,R0                  ;Get patch address
        MOV     -(R0),SAVE              ;Keep original code
        MOV     CAL,(R0)+               ;Apply the patch
        MOV     (R0),SAVE+2             ;
        MOV     CAL+2,(R0)              ;
        JMP     STARTI                  ;Now start BASIC
        .BLKB   BUFFER+82.-.-10         ;Rest of line buffer
                                        ;(overflows next code with 10 bytes)

FIRST:  ADD     #SECOND-FIRST,@ADR      ;Bump entrypoint
        .GTLIN  #BUFFER                 ;Get that first line now
SECOND: MOVB    @POINTR,R0              ;Get a character
        BEQ     10$                     ;End of line, unpatch now
        INC     POINTR                  ;Bump the pointer
        BR      QUIT                    ;Return with carry clear

10$:    ADD     #THIRD-SECOND,@ADR      ;Bump entrypoint
        MOV     #15,R0                  ;
        BR      QUIT                    ;Done

THIRD:  MOV     ADR,R0                  ;Get patch address
        MOV     SAVE+2,(R0)             ;Restore original code
        MOV     SAVE,-(R0)              ;
        MOV     #12,R0                  ;
QUIT:   CLC                             ;
        RETURN                          ;And disappear !

ADR:    .WORD   BLOCKI+15062-14770
SAVE:   .WORD   0,0
CAL:    JSR     PC,@#FIRST              ;Patch template

POINTR: .WORD   BUFFER                  ;Buffer pointer

        .END    TRICK                   ;True start address
```



```
;RT11 macro's
        .MCALL  .GVAL,  .LOOKU, .EXIT,  .READW, .WRITW, .PRINT
        .MCALL  .TTYIN, .CLOSE

;Constants:
        MONAME  = 406

COMAND::.PRINT  #S0             ;Are you sure ?
        .TTYIN                  ;Get answer
        CMPB    #'Y,R0          ;Ok ?
        BNE     NOGO            ;No
        .GVAL   #AREA,#MONAME   ;
        BCS     GVERR           ;Error ?
        MOV     R0,NAME+2       ;First part of monitor name
        .GVAL   #AREA,#MONAME+2 ;
        BCS     GVERR           ;
        MOV     R0,NAME+4       ;
LOOP:   TST     @PNT            ;Patched all files ?
        BEQ     DONE            ;Yes
        .LOOKUP #AREA,#0,@PNT   ;Open file
        BCS     LKPERR          ;Error ?
        ADD     #2,PNT          ;Bump pointer to table start
        MOV     @PNT,R5         ;Get address of patch table
        ADD     #2,PNT          ;
        CLR     R1              ;Start at offset 0
SEARCH: MOV     R5,R4           ;
        MOV     (R4)+,R3        ;Number of compare words
        BEQ     30$             ;All done !
10$:    CALL    GETWRD          ;
        ADD     #2,R1           ;Bump word pointer
        BIC     (R4)+,R0        ;Zero don't care bits
        CMP     (R4)+,R0        ;
        BNE     SEARCH          ;False
        DEC     R3              ;End of compare list ?
        BNE     10$             ;No, check next word
        ADD     (R4)+,R1        ;Offset for first patch word
        MOV     (R4)+,R3        ;Number of patches
        BNE     20$             ;If zero, call special patch routine
        CALL    @(R4)           ;
        ADD     #2,R4           ;
        BR      25$             ;
20$:    MOV     (R4)+,R0        ;
        CALL    PUTWRD          ;
        ADD     #2,R1           ;
        DEC     R3              ;All patches done ?
        BNE     20$             ;Yes
25$:    MOV     R4,R5           ;
        BR      SEARCH          ;Go for next patch

30$:    CLR     R1              ;
        CALL    GETWRD          ;Write last block if necessary
        .CLOSE  #0              ;Close this file
        BR      LOOP            ;Go for next file
```

```
DONE:   .PRINT  #S9             ;
NOGO:   .TTYIN                  ;Flush rest of line
        CMP     #12,R0          ;
        BNE     NOGO            ;
        .EXIT                   ;

GVERR:  .PRINT  #S1             ;
        BR      NOGO            ;

LKPERR: .PRINT  #S2             ;
        BR      NOGO            ;

REDERR: TSTB    @#52            ;End of file ?
        BNE     10$             ;No
        .PRINT  #S5             ;Patern not found
        BR      NOGO            ;
10$:    .PRINT  #S3             ;
        BR      NOGO            ;

WRTERR: .PRINT  #S4             ;
        BR      NOGO            ;

;Routine GETWRD is called with the word offset in R1. Returns
;the contents of that address from the file in R0

GETWRD: MOV     R1,-(SP)        ;Save word offset
        BIC     #777,R1         ;
        SWAB    R1              ;Get block number
        ASR     R1              ;
        CMP     R1,CURBLK       ;Is it in memory now ?
        BEQ     10$             ;Yes
        TST     CHANGD          ;Was the current block changed ?
        BEQ     5$              ;No, no rewrite
        .WRITW  #AREA,#0,#BUFF,#400,CURBLK
        BCS     WRTERR          ;Write error ?!
        CLR     CHANGD          ;Reset written flag
5$:     MOV     R1,CURBLK       ;
        .READW  #AREA,#0,#BUFF,#400,CURBLK
        BCS     REDERR          ;Read error !
10$:    MOV     (SP),R1         ;Get word offset
        BIC     #177000,R1      ;Get offset within block
        MOV     BUFF(R1),R0     ;
        MOV     (SP)+,R1        ;Return original R1
        RETURN                  ;

;PUTWRD writes R0 in the file location R1

PUTWRD: MOV     R0,-(SP)        ;Save new value
        CALL    GETWRD          ;Do addressing
        MOV     (SP)+,R0        ;Restore R0
        MOV     R1,-(SP)        ;
        BIC     #177000,R1      ;
        MOV     R0,BUFF(R1)     ;Put the word
        MOV     SP,CHANGD       ;Set changed flag
        MOV     (SP)+,R1        ;Restore R1
        RETURN                  ;
```



```
PNT:    .WORD   .+2
        .WORD   NAME,MTABLE     ;RT11 monitor
        .WORD   0               ;End of table

NAME:   .RAD50  /SY .....SYS/
MTABLE:
;Put a + in the monitor name (bootstrap message)
        .WORD   3               ;Check two words
        .WORD   0,12*400+15     ;
        .WORD   0,'RT           ;
        .WORD   0,'-1           ;
        .WORD   0,0             ;Search end of boot message, add '+'
        .WORD   PLUS            ;Address of + routine

;Let undefined command execute as RUN <command>
        .WORD   3               ;Check 3 words
        .WORD   0,105722        ;
        .WORD   0,3376          ;
        .WORD   0,100762        ;
        .WORD   0,0             ;Call special routine
        .WORD   PTCH1           ;

;End of patch list is here:
        .WORD   0


PLUS:   ADD     #2,R1           ;Bump file address
        CALL    GETWRD          ;Get next word from the file
        TSTB    R0              ;Low byte zero ?
        BEQ     10$             ;Yes, found end of string
        BIT     #177400,R0      ;Highbyte zero
        BNE     PLUS            ;No, go for next word
        CMPB    #'+,R0          ;Is the + already there ?
        BEQ     30$             ;Yes, quit
        BIS     #'+*400,R0      ;Put a +
        BR      20$             ;
10$:    SUB     #2,R1           ;Backup one word
        CALL    GETWRD          ;
        SWAB    R0              ;
        CMPB    #'+,R0          ;Is the + already there ?
        BEQ     30$             ;Yes, quit
        ADD     #2,R1           ;No, advance file pointer
        MOV     #'+,R0          ;
20$:    CALL    PUTWRD          ;Write it back
30$:    RETURN                  ;Done

PTCH1:  CALL    GETWRD          ;Get the 'BR BADCOM'
        BIT     #177000,R0      ;Must be a pure branch
        BNE     10$             ;Nope, skip it
        MOV     R0,-(SP)        ;And keep it
        MOV     (PC)+,R0        ;Store some instructions
         INC    R5              ;Get first command letter back
        CALL    PUTWRD          ;
        ADD     #2,R1           ;
        MOV     (PC)+,R0        ;
         MOV    (PC)+,R0        ;'RUN' is fifth command in list
```

```
          CALL    PUTWRD          ;
          ADD     #2,R1           ;
          MOV     (PC)+,R0        ;
           .WORD  5               ;
          CALL    PUTWRD          ;
          ADD     #2,R1           ;
          MOV     (PC)+,R0        ;
           BR     .+10            ;Join main stream
          CALL    PUTWRD          ;
          ADD     #6,R1           ;Skip 'BITB #100,@R2'
          MOV     (SP)+,R0        ;Get brach to BADCOM
          SUB     #6,R0           ;Moved brach over 14 bytes
          SUB     (PC)+,R0        ;Change 'BR' in 'BNE'
           BR     .+2             ;
          ADD     (PC)+,R0        ;
           BNE    .+2             ;
          CALL    PUTWRD          ;
  10$:    RETURN                  ;Done

;Messages:
S0:       .ASCII  <15><12>
.ASCII    <11>/This program can patch the currently running RT11 V4/<15><12>
.ASCII    <11>/monitor with the effect that non-RT11 commands will/<15><12>
.ASCII    <11>/RUN the program of that name, eg. if you type 'ABC',/<15><12>
.ASCII    <11>/RT11 translates this to 'RUN ABC'. RT11 already/<15><12>
.ASCII    <11>/allows you to pass program parameters by typing them/<15><12>
.ASCII    <11>/beyond the program name in a RUN command: RUNOFF A B/<15><12>
.ASCII    <11>/will process the file A, giving file B/<15><12><12>
.ASCII    <11>/This program is ofered to you by Westvries Systems BV,/<15><12>
.ASCII    <11>/P.O.Box 438, Culemborg, The Netherlands./<15><12><12>
.ASCIZ    /?COMAND-W-Patching current monitor - Are you sure ? /<200>
S1:       .ASCIZ  /?COMAND-F-.GVAL fails/
S2:       .ASCIZ  /?COMAND-F-Program file not found/
S3:       .ASCIZ  /?COMAND-F-Read error/
S4:       .ASCIZ  /?COMAND-F-Write error/
S5:       .ASCIZ  /?COMAND-F-Patern not found/
S9:       .ASCIZ  /?COMAND-I-Done - Reboot now !/
          .EVEN

;DATA:
AREA:     .BLKW   10
CURBLK:   .WORD   -1
CHANGD:   .WORD   0
BUFF:     .BLKW   400

          .END    COMAND
```



```
; This is a serial driver for the Digital LA34/LA38/LA120
; printers.  The printer controled with the XON/XOFF pro-
; tocol.  The SET LA PITCH n and SET LA MARGIN n commands
; are supported.

          .SBTTL  MACROS AND DEFINITIONS

          .MCALL  .DRDEF

          .DRDEF  LA,3,SPECL$!WONLY$!SPFUN$,0,176500,300

          LA.CSZ = 132.
          LA.PSZ = 66.
          LA$CSR  = LA$CSR+4
          LA$VEC  = LA$VEC+4
          IE      = 100
          RDY     = 200
          ERR     = 100000
          BELL    = 7
          HT      = 11
          LF      = 12
          FF      = 14
          CR      = 15
          ESC     = 33
          CTRLO   = 'O-100
          CTRLQ   = 'Q-100
          CTRLS   = 'S-100
          SPACE   = 40
          COLSIZ  == LA.CSZ
          PAGSIZ  == LA.PSZ
          DELAY   = 30.             ;Set equal to baud rate / 10.


.SBTTL  SET OPTION PARAMETER TABLE

.DRSET  WIDTH,30.,0.WIDTH,NUM
.DRSET  FORM0,177,0.FORM0,NO
.DRSET  CSR,2,0.CSR,OCT
.DRSET  VECTOR,477,0.VEC,OCT
.DRSET  FORM,100000!LF,0.FF,NO
.DRSET  LENGTH,1,0.LENG,NUM
.DRSET  SKIP,-1,0.SKIP,NUM
.DRSET  PITCH,'4,0.PTCH,NUM
.DRSET  MARGIN,'1,0.MARG,NO
.DRSET  PAUSE,5067,0.PAUS,NO


          .SBTTL  SET OPTION PROCESSING ROUTINES

0.WIDT:   MOV     R0,COLCNT
          MOV     R0,@#SWIDT
0.TST:    CMP     R0,R3
          RTS     PC
```

```
O.FORM0:MOV     (PC)+,R3
        .WORD   FF
        MOV     R3,O$FORM
        RTS     PC

O.CSR:  MOV     R0,176
        MOV     R0,LAIS
        ADD     R3,R0
        MOV     R0,LAIB
        ADD     R3,R0
        MOV     R0,LAS
        ADD     R3,R0
        MOV     R0,LAB
        CMP     R0,#160000
        RTS     PC

O.VEC:  MOV     R0,LA$VTB
        ADD     #4,R0
        MOV     R0,LA$VTB+6
        CMP     R3,R0
        RTS     PC

O.FF:   MOV     (PC)+,R3
        .WORD   FF
        MOV     R3,O$FF
        RTS     PC

O.LENG: MOV     R0,O$LENG
        MOV     R0,LINCTR
        BR      O.TST

O.SKIP: TST     R0
        BEQ     10$
        INC     R0
10$:    MOV     R0,O$SKIP
        CMP     O$LENG,R0
        RTS     PC

O.PTCH: ADD     #'0,R0
        MOVB    R0,O$PTCH
        CMPB    R3,R0
        RETURN

O.MARG: MOV     (PC)+,R3
        .WORD   '9
        MOVB    R3,O$MARG
        RETURN

O.PAUS: MOV     (PC)+,R3
        .WORD   5267
        MOV     R3,NEWPAG
        RETURN

.IIF LT <1000-.> .ERROR ;Set routines reach too far !
```



```
        .SBTTL  DRIVER ENTRY

        .DRBEG  LA              ;Driver entrypoint
        MOV     LACQE,R4        ;Get pointer to request block
        TSTB    Q$FUNC(R4)      ;Is this a function request ?
        BEQ     10$             ;No, a normal read/write request
        CLR     Q$WCNT(R4)      ;Set filelength to zero (for .Enter)
        CMPB    #1,Q$FUNC(R4)   ;Is this a .Colse ?
        BNE     LADONE          ;No, other functions are no-op
        TST     OPEN            ;Did we ever print ?
        BEQ     LADONE          ;No, this is a dummy .Close, ignore it
        INC     EOF             ;Set the end-of-file flag
        CLR     OFLAG           ;Make sure the final FF comes out
        MOV     SP,(R4)         ;Set blocknumber to non-zero
        BR      20$             ;Continue with setup code

10$:    ASL     Q$WCNT(R4)      ;Make the wordcount to a byte-count
        BEQ     LADONE          ;If zero, were done now (a seek request)
        BCC     LAERR           ;If no carry here, it was a read - fatal
        MOV     SP,OPEN         ;Set the LP-is-open flag
20$:    BIS     #IE,@(PC)+      ;Enable input interrupts
LAIS:    .WORD  LA$CSR          ;(address of input CSR)
RET:    BIS     #IE,@LAS        ;Enable output interrupts as well
        RETURN                  ; and wait for the next interrupt

INISTR=.-1                      ;Initial string
        .ASCII  <ESC>/[/        ;set horizontal spacing
O$PTCH: .ASCII  /1w/<ESC>/[/    ; to n ;Set left margin to 0 or ?
O$MARG: .ASCII  /0;s/<15>       ; then do a CR and maybe
O$FORM: .ASCIZ  <0>             ; a FF (when SET LA FORM0 is on)
        .EVEN

        .DRVTB  LA,LA$VEC,LIINT ;Device vector table
        .DRVTB  ,LA$VEO,LAINT   ;

        .SBTTL  INPUT INTERRUPT SERVICE

        .DRAST  LI,4,LADONE     ;Here enter input interrupts
        MOV     @(PC)+,R4       ;Read the incomming character
LAIB:    .WORD  LA$CSR+2        ;(address is input data buffer)
        BIC     #^C<177>,R4     ;Clear parity and error bits
        CMP     R4,#CTRLQ       ;Is it control/Q ?
        BNE     10$             ;No, pass on
        CLR     LAWAIT          ;Yes, clear the wait-with-output flag
        BR      RET             ;enable output interrupts to restart output

10$:    CMP     R4,#CTRLS       ;Is it control/S ?
        BNE     20$             ;No, pass on
        INC     LAWAIT          ;Yes, set the wait-with-output flag
        RETURN                  ;

20$:    CMP     R4,#CR
        BNE     30$
        CLR     PAUSE
        BR      RET

30$:    CMP     R4,#CTRLO
        BNE     RTI
        COM     OFLAG
RTI:    RETURN
```

```
        .SBTTL  OPERATION COMPLETE

LAERR:  BIS     #HDERR$,@-(R4)
LADONE: CLR     LAWAIT
        CLR     @LAS
        CLR     @LAIS
        CLR     LAFBLK+2
        .DRFIN  LA


        .SBTTL  OUTPUT INTERRUPT SERVICE

        .DRAST  LA,4,LADONE
        MOV     LACQE,R4
        BEQ     RTI
        TSTB    @(PC)+
LAS:     .WORD  LA$CSO
        BPL     RTI
        CLR     @LAS
        .FORK   LAFBLK
        MOV     LACQE,R4
        TST     (PC)+           ;Control/O typed ?
OFLAG:  .WORD   0
        BNE     LADONE          ;Yes, quit
        TST     EOF
        BNE     DOEOF
        TST     @R4
        BEQ     BLKO
LANEXT: TSTB    @LAS
        BPL     RET
        TST     (PC)+
LAWAIT: .WORD   0
        BNE     RTI
        TST     PAUSE
        BGT     DOPAUS
        MOV     (PC)+,R5
FFFLAG:  .WORD  0
        BNE     DOFORM
IGNORE: TST     Q$WCNT(R4)
        BEQ     LADONE
        INC     STRPNT          ;bump initial string pointer
        MOVB    OFFSET(PC),R5   ;get a byte from the initial string
STRPNT=.-2                      ;pointer to initial string
OFFSET=INISTR-.                 ;offset to first byte of initial string
        BNE     INICHR          ;got a valid character
        DEC     STRPNT          ;reset pointer
.IF EQ  MMG$T
        MOVB    @Q$BUFF(R4),R5
        INC     Q$BUFF(R4)
.IFF
        JSR     PC,@$GTBYT
        MOV     (SP)+,R5
.ENDC
        INC     Q$WCNT(R4)
        BIC     #^C<177>,R5
        BEQ     IGNORE
```



```
INICHR: ASRB    (PC)+
SKPFLG:  .WORD  0
        BCC     10$
        CMPB    #FF,R5
        BEQ     IGNORE
10$:    CMPB    #SPACE,R5
        BHI     CHRTST
PCHAR:  DEC     COLCNT
        BLT     IGNORE
        ASLB    (PC)+
TABCNT:  .WORD  1
        BEQ     RSTTAB
PRINTC: MOVB    R5,@(PC)+
LAB:     .WORD  LA$CSO+2
        BR      LANEXT


CHRTST: CMPB    #HT,R5
        BEQ     HDWTAB
        CMPB    #FF,R5
        BEQ     SENDFF
        CMPB    #CR,R5
        BEQ     RSTC
        CMPB    #LF,R5
        BNE     PRINTC
        CMP     LINCTR,(PC)+
O$SKIP:  .WORD  0
        BLE     SKIPFF
DOFORM: DEC     LINCTR
        BGT     RSTC
NEWPAG: INC     PAUSE
        MOV     (PC)+,(PC)+
O$LENG:  .WORD  LA.PSZ
LINCTR:  .WORD  LA.PSZ
        CLR     FFFLAG
RSTC:   MOV     (PC)+,(PC)+
O$WIDT:  .WORD  LA.CSZ
COLCNT:  .WORD  LA.CSZ
RSTTAB: MOV     #1,TABCNT
        BR      PRINTC

HDWTAB: ASLB    TABCNT
        BEQ     RSTTAB
        DEC     COLCNT
        BR      HDWTAB

SKIPFF: INC     SKPFLG
        BR      SENDFF

BLKO:   INC     @R4             ;set blocknumber to non-zero
        MOV     #OFFSET,STRPNT  ;set stringpointer to start of init string
        BR      IGNORE

DOEOF:  CLR     (PC)+
EOF:    .WORD   0
        CLR     (PC)+
OPEN:   .WORD   0
        MOV     #-1,(PC)+
PAUSE:  .WORD   0
```

```
SENDFF: TST     LINCTR
        BEQ     LANEXT
        MOV     #CR,R5
O$FF    = .+2
        MOV     #100000!LF,FFFLAG
        BMI     RSTC
        CLR     LINCTR
        BR      RSTC

DOPAUS: CLR     R5              ;
        INC     PAUSE           ;
        CMP     PAUSE,#DELAY    ;Timed out the bell-delay ?
        BNE     PRINTC          ;No, send another null
        MOV     #BELL,R5        ;Yes, send a bell
        MOV     #1,PAUSE        ;Reset delay counter
        BR      PRINTC          ;

LAFBLK: .WORD   0,0,0,0

        .DREND  LA

        .END
```

I would like to include the following program in the next issue of the Mini-Tasker, if possible.

The file DATTIM.MAC when compiled and linked may be run (as R DATTIM) or included in STARTS.COM.

When called as the last program from within STARTS.COM, it will :

- **Print the last entered Date at the terminal and accept a new Date,**
- **Print the last entered Time at the terminal and accept a new Time.**

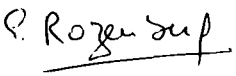
The user may :

- **Either enter a new Date/Time**
- **Or enter RETURN ; in that case the last Date/Time entered will be passed to RT11.**

See the examples following the program.

Best Regards.

S. ROZENBERG
Principal Software Consultant




```
        .TITLE  DATTIM
        .ENABL LC
        .NLIST BEX
;
; Author :      S. ROZENBERG
; Societe       INFI
; 12, rue de l'Etang St-Denis
; 92370, CHAVILLE       FRANCE
; tel: 750-92-94
;
; This file accepts a Date and Time from the console terminal
; and passes it to RT11.
; If no Date or Time is entered, the previous date / time is passed.
; Include DATTIM into STARTS.COM as the last command for forcing
; the operator to enter the date & time.
;

        .MCALL .EXIT .GTLIN .PRINT .LOOKUP .READW .WRITW

CHNIFS=4000
JSW=44

;                       RT11 communication area
        .ASECT
        .=510
        .WORD AR1-ARD
ARD:    .ASCII /DATE /
AREA:   .ASCII /        /
        .BYTE 200
ART:    .ASCII /TIME /
AREA1:  .ASCII /        /
        .BYTE 200
AR1=.
        .PSECT
GVAL:
        BIS #40000,@#JSW                ; enable lower case on input
        MOV #ARD,R0
        MOV #CLEAN,R3                   ; clean new date/ time area
        MOV #BUFF+ARD,R2                ; save old date / time
        MOV @#510,R1
10$:    MOVB (R0),(R2)+
        MOVB (R3)+,(R0)+
        SOB R1,10$


        .PRINT #BUFF+ARD                ; print old date
        .GTLIN #AREA,#EXCLAM            ; accept new date
        .PRINT #BUFF+ART                ; print old time
        .GTLIN #AREA1,#EXCLAM           ; accept new time

        MOV #2,R0                       ; flag : 0= nothing entered
```

```
        TSTB AREA                       ; date entered ?
        BNE 1$                          ; yes, skip
        MOV #BUFF+AREA,R3               ; no, restore previous date
        MOV #AREA,R2
        MOV #ART-AREA,R1
11$:    MOVB (R3)+,(R2)+
        SOB R1,11$
        DEC R0                          ; flag= 1
1$:     TSTB AREA1                      ; time entered ?
        BNE 2$                          ; yes, skip
        MOV #BUFF+AREA1,R3              ; no, restore previous time
        MOV #AREA1,R2
        MOV #AR1-AREA1,R1
12$:    MOVB (R3)+,(R2)+
        SOB R1,12$
        DEC R0                          ; flag= 0
2$:     TST R0                          ; smthg entered ?
        BEQ 5$                          ; no, skip
        .LOOKUP #PGAREA,#0,#FILE        ; open file : SY:DATTIM.SAV
        .READW  #PGAREA,#0,#BUFF,#<AR1+1/2>,#0  ; read in block 0
        MOV #ARD,R0                     ; transfer data entered into buffer
        MOV @#510,R1
        MOV #BUFF+ARD,R2
3$:     MOVB (R0)+,(R2)+
        BNE 4$
        MOVB #' ,-1(R2)                 ; make null byte (from .GTLIN)= space
4$:     SOB R1,3$
        .WRITW  #PGAREA,#0,#BUFF,#<AR1+1/2>,#0 ; write back block 0
        BCS 6$

5$:     BIS #CHNIFS,@#JSW               ; exit & pass command to RT11
        CLR     R0
        .EXIT
6$:     .PRINT #WRTERR
        .EXIT

FILE:   .RAD50  /SY DATTIMSAV/
WRTERR: .ASCIZ  /?-DATTIM-F error writing on SY:DATTIM.SAV/
CLEAN:  .ASCII /DATE /
        .ASCII /        /
        .BYTE 200
        .ASCII /TIME /
        .ASCII /        /
        .BYTE 200
EXCLAM: .ASCII  / ? /
        .BYTE   200
        .EVEN
PGAREA: .BLKW   5
BUFF:   .BLKW   256.

        .END GVAL
```


What with the prices of disk storage systems in the 20-100 MByte range becoming more attractive each month, many RT-11 systems are aquiring such disk drives, along with the housekeeping problems that go with them. This is actually a problem in DP management; because PDP-11's are called "mini"computers, we often run into people who seem to think that they get put into a corner and take care of themselves. PDP's are very reliable machines, and RT-11 is a relatively simple, clean, and fast operating system, but management problems are a function of the size of the installation. The following is a description of one solution to managing a large-capacity disk on an RT-11 system.

If you are putting a big disk on your RT-11 system, it is usually because you have a large number of files being handled by a number of users (we'll ignore the case of a system with one user who produces a small number of very large files). In effect, your system is a multi-user system, even though the users work serially rather than in parallel. If you use STAR-eleven or TSX, your system is a true multi-user system. The problem consists of two halves: 1) handling a large number of files, and 2) user protection. RT-11 is limited on both these points. The solution, essentially, is to divide the mass storage into a number of virtual units. A complete solution must say something about the allocation of these virtual units, however.

Our institute consists of 5 laboratory groups; in all, we have about 20 users. We have a STAR-eleven/RT-11 system with an 11/34 host supporting 9 satellites (11/03 and 11/23). The laboratories use the real-time capabilities of the satellite computers for data handling in on-line experiments, plus data evaluation, text editing, etc., and there are a couple of satellites for program development and text editing. When our system had grown to the point where we absolutely had to have more random access storage, we wanted to make use of our experience during the time when we only had 2 RK05's. A review of all our disks and discussions with other PDP-11 installation managers led to the discovery of the following remarkable General Principle: free space on all random access devices asymptotically approaches 200 blocks, regardless of the size of the device. This is apparently independent of availability of backup media. Actually, this principle is not so remarkable; it is the same Law of Elastic Dimensions that holds when you move into a bigger or smaller house: you either finally have room for a few more things in the bigger place, or you finally have to get rid of some of your junk to fit into the smaller place. The important thing is, once you are settled, you need the stuff you have and can rarely be persuaded to move much of it even just out to the garage. We got a grant to add a 64 MByte drive to our system, and we didn't want our dear users to fill it up with annotated histories of all their experiments, complete with all the drafts of all the papers they wrote, various business and private correspondence, program sources, memos, games, and other magnetic compost. Another disconcerting tendency caused by availability of random access storage is the proliferation of different versions of the same program, almost never documented. After some altercations, we decided not to just divide the 64 MBytes into equal portions for each lab and one for the computer group, but to make categories of mass storage and to limit the amount of private space for

each group rather severely. We arrived at the following scheme: We kept the RK05 as the operating system disk, so that seeks for operating system programs would be overlapped with those for files on the large disk. This has the added advantage of direct compatiblility with the distribution media: we never need a SYSGEN to implement a new version or to apply patches. We can also run the system in an emergency from the RK05's if the large disk should fail. The large disk was partitioned as follows:

1) 7 private areas, each 4.2 MByte.

2) A public "scratch area", 21.3 MByte.

3) A library area, 4.2 MByte

4) A documentation area, 4.2 MByte

5) An area for systems and test programs, 4.2 MByte

The "areas" were implemented in two stages: first, the DP:-handler was rewritten to access the disk as DP0:, DP1:, and DP2:, each 21.3 Mbytes. Then, using Ian's DISKS-11 virtual device handler, we defined the "areas" on these as virtual devices. DP0: was left intact as the scratch area; this gave a safety "firewall" between the scratch area and the private area, since DP1: and DP2: contain only virtual devices and no other files. All this does not add appreciably to the system overhead: we have a net system throughput of approximately 8,900 16-bit words/sec to the virtual devices from each 11/03 satellite (the 11/23's do 11,000) using the FAST option. Reads go even faster because of caching.

The relatively small size of the private areas forces the users to distinguish between active ("live") and inactive ("dead") files and roll out the files they don't really need onto tape or floppies. The amount of free space left in the private areas (asymptotically 200 blocks) is often too small for the data processing tasks, so almost all users make use of the scratch area. This is large enough to read in an entire MAGtape (800 BPI) if necessary, so we can copy tapes with just a single drive. This is the only place where user protection becomes a problem. We use the name convention devised by Mike and Ian: users begin file names with their initials. We have had no collisions in 2 years. This public area is initialized at the start of each week without backup; it is the users' responsibility to rescue the files they want to keep. Because of this, there is almost always enough space available for our largest data processing tasks, tape copying, etc. In addition, our users have become accustomed to storing all their object and list files, temporary backup files and test versions of programs in the scratch area, since these are automatically cleared out of the system the next week.

If on-line random-access capacity is small, you usually only keep 2 kinds of files on-line: those which need to be accessed fast (current data, overlays) and those which have to be accessed often (CUSP's and whatever you bought your computer for). The other stuff is kept off-line in storage volumes which are loaded as needed. If capacity is large, you can keep less-used files in on-line libraries. The less a file is accessed, the more important "taste" and "feeling" are in deciding to have it available on-line, but, properly, you should rank your files according to frequency of access (RT-11 does not have a way to do this -- yet). There is one other reason for keeping a file on-line:

it says that this is the current version of this file. It gives users one place to look for the latest, accepted versions of files.

We included 3 on-line libraries in our scheme: an applications program and subroutine library (LIB:), a documentation library (DOC:), and a library of systems programming routines and hardware test programs. The documentation library includes descriptions of our system hardware, writeups of frequently used programs, indexes and reference lists for all applications programs, system software, and system documentation, guidlines for the use of the system, help files of various kinds, and even addresses and telephone numbers of the staff. It is just easier to do a TYPE DOC:xxxx at the computer than to search through desks and shelves for writeups.

I don't want to get carried away with the details of our system, or how easy it was to implement the above scheme. The point is that we now have a genuinely midi-computer system with 10 simultaneous users accessing the same disk and having no problems. All with RT-11. RT-11 (and STAR-eleven) can do it!

Howard Schultens
Zentrum Physiologie
Humboldtallee 7
D-3400 Goettingen
W. Germany

I have been looking for a data base management system to run on a PDP-11/23 under RT-11. I have located several sources and felt you may want to publish them in the "Mini Tasker".

Best regards,

P.O. BOX 647, WARSAW, INDIANA 46580 (219) 267-3226

Jeff Hagerlin
Chemist

Contel Information Systems — "RESIDENT"
4330 East-West Hwy., Suite 300
Bethesda, MD 20814
301-654-6410

California Systems Associates — "DATAMANAGER"
2845 Mesa Verde Drive East, Suite 4
Costa Mesa, CA 92626
714-546-9710

Se-Tre Corp.
421 Seventh Avenue
New York, NY 10001
212-736-7257

"SIMILE"

EEC Systems Inc.
286 Boston Post Road
Wayland, MA 01778
617-443-6376

"LEX-11"

Quick RT-11 Quiz

Compiled by John Yardley, JPY Associates Ltd.

Answers may be found at the back of this newsletter.

Question 1. (4 Marks, 2 for part a, 2 for part b)

Assume that only the following files exist on RK1:

FILE1.FOR
FILE2.FOR
FILE3.FOR

a) Write the following commands as a single command as concisely as possible:

```
COPY RK1:FILE1.FOR RK2:FILE1.FOR
COPY RK1:FILE3.FOR RK2:FILE3.FOR
```

b) Write the following command as concisely as possible:

```
COPY/CONC RK1:FILE1.FOR,RK1:FILE2.FOR,RK1:FILE3.FOR TEMP.FOR
```

Question 2. (6 Marks, 1 for each part)

What is wrong with the following commands?

a) `RENAME FILE%.FOR FRED%.FOR`

b) `SET TT LENGTH=20`

c) `COPY/SINCE:1:JUN:82 RK1: RK2:`

d) `FORT FILE*.FOR`

e) `DIR/ORDER:ALP`

f) `CREATE/START:100./ALLOC:10 FRED`

Question 3. (3 Marks)

You have just returned from a DECUS conference with a pirate RT-11 V4.0 distribution kit. In your haste (you wanted to go to the VT18X presentation and were running short of time) you forgot to copy the bootstrap on to the kit. You do have a bootable V3B kit however. How do you get your V4.0 kit up and running?

Question 4. (2 Marks)

Which one of the following system components is placed highest in memory?

a) System Device Handler
b) System communication area
c) Resident Monitor
d) User Service Routine

Question 5. (8 Marks, 1 for each part)

Which of the following monitor requests is illegal under a V4.0 SJ monitor sysgenned with ALL options.

a) .ENTER
b) .DSTATUS
c) .MRKT
d) .CSTAT
e) .MTSET
f) .DEVICE
g) .QUEUE
h) .LOCK

Question 6. (8 Marks, 1 for each part)

Which of the following monitor requests use the USR?

a) .READ
b) .READC
c) .LOOKUP
d) .SYNCH
e) .GTLIN
f) .QSET
g) .REOPEN
h) .PURGE

Question 7. (5 Marks, 1 for each part)

You have just arrived at a customer's site in the Outer Hebrides bringing with you a system sysgenned on your own VAX (running RT-11 of course). The customer's system consists of an LSI-11 with a TU58 system device. You discover that some of the sysgen options are wrong and you want to change them without doing a sysgen on the TU58 system. Which of the following can you do without a sysgen?

a) Change the level of indirect command file nesting.
b) Increase the number of free device slots.
c) Include month rollover.
d) Have a 50-cycle clock rate.
e) Change the default device for indirect command files.

Question 8. (2 Marks)

You wish to install an additional serial printer on your system, but already have one using the LS handler. How can you do it?

Question 9. (3 Marks)

A friend comes to stay and to look at your new home computer. This is a VT100 with the personal computer option, the VT18X running RT-11 V4 under the RT-11 emulator for CP/M. Your friend notices that you are using TECO V29. You know that you should be running V36 but are too embarrased to mention that the distributed VEG.TEC always gives an error message? Why is this?

Question 10. (2 Marks)

You are at an unfamiliar site, using RT-11 V3B. You have just created a MACRO program called TEST.MAC on SY: but every time you try to MACRO it you get the message:

?KMON-F-File not found

You have double checked to make sure TEST.MAC is there, what is the problem? (You cannot ask DEC for help because they won't support old versions of RT-11).

Question 11. (3 Marks)

You have nearly finished a lengthy sysgen, when an RSX user distracts you and types two control/C's at the console. What should you do?

Score:

40-46 **Send me a CV.**

35-39 **Write an article for the Single-User.**

25-34 **Write an article for the MiniTasker.**

15-24 **Go back to RSX.**

0-14 **Ahh so you work for DEC do you?**

USER REQUESTS
---

we are interested in acquiring a program for Optical System and Lens Design for use on RT-11 Operating System. If you know of the availability of this program, please contact,
Miriam Maney , Decus # 160067
State College of Optometry
100 East 24th Street
New York, NY 10010
212-477-7051

USER RESPONSES

Visual Information Systems, Inc.

P.O. BOX 2502 WEST LAFAYETTE, IN 47906

I would like to take this opportunity to respond to a request in the July Mini-tasker about an RT-11 "Visicalc type product". We have experienced great success with a product written by Computer Systems Corporation called CALC-11. CALC-11 is written in C and it is available for RT-11, RSTS/E, and RSX-11M. This product might be of interest to all three sigs. I am sure Computer Systems can provide more detailed information about their product. Their mailing address is

Computer Systems Corporation
5540 Rock Hampton
Indianapolis, IN 46468

Sincerely,

Chuck Reed

Chuck Reed
Visual Information Systems

UPCOMING SYMPOSIUM INFORMATION

Fall Symposia Preview

The Fall Symposia has been scheduled now and most of you will be receiving the preliminary program very soon. The most frequently asked question I receive is 'which days should I attend?' Let me answer that by going over the day-by-day schedule for RT-11. In general, you will notice that there is no white space on the schedule. We had to schedule sessions through lunch and long into the night. Mike Allen and I worked together to minmize conflicts between RT-11, LABS, LSI, and HMS sessions and I think we did a good job. If you have a complaint, you'll have to buy me a beer at the convention and tell me about it.

Monday, December 6, 1982

The day will start with a combined RT-11, LABS, LSI, and HMS roadmap session. We decided to combine all the roadmaps into one because these four SIGs have common interests. The afternoon will have sessions on Micro Power/PASCAL, MACRO/FORTRAN interactions, operating systems comparison for RT-11 users, RT-11 Product Panel and a Version 5 overview. The theme for the evening is device handlers for RT-11. Les Parent will discuss logical disk subsetting for V5 while two users will present papers on special purpose device drivers that they have written. Dan Kingsbury will conclude the evening with a tutorial on advanced techniques for writing RT-11 handlers.

Tuesday, December 7, 1982

This is a 'light' RT-11 day. One session will explore RT-11 DCL internals and the other will look at an input file formatter for data files. The LSI foreign peripherals forum is also scheduled for this day. The evening will end with the reception.

Wednesday, December 8, 1982

The highlight of today will be the two Extended Memory talks. Mr. XM himself (Greg Adams) will present the ABC's of the XM monitor. Following that talk, a knowledgeable user will discuss XM gotcha's and workarounds. This would be a good day to take in a couple of DEC personal computer sessions or some networking sessions.

Thursday, December 9, 1982

This is the RT-11 marathon day. We will start the day with a tutorial on Indirect Control Files for RT-11 V5. We will continue on with a TSX-Plus 'magic' session. A FORTRAN tutorial will look at the various options such as EIS, FIS, and FPU. Two other users will look at text editing and automated test equipment environments. Things will start to get serious for the evening sessions. Mr. XM will give us a look at RT-11 XM internals, while Les will present the illustrated history of RT-11. We will finish the evening (one way or

the other) with the Speakout session. This session is user driven and any RT-11 problem or war story is welcomed. If no one has anything to say, Jack Crowell and I will discuss our favorite TECO macros for the entire evening.

Friday, December 10, 1982

Today, we can start to wind down. Two user papers will look at the IEEE 488 Bus subroutine package and the results of a survey of RT-11 users. The infamous Feedback session will be a little different this time because Version 5 will be coming out soon. For this session, the RT-11 developers plan on presenting a report from the various RT-11 test sites. The RT-11 business meeting will conclude the morning.

So, there should be a little bit for everyone. There will be a number of sessions that will preview the features of Version 5 and a number of user papers on how to do it with RT-11. I hope to see you there!

SPR'S

SUBMITTED BY: Ned W. Rhodes

PHONE: 560-5000 (X 2673)

CAN THE PROBLEM BE REPRODUCED AT WILL? YES [X] NO [ ]

ATTACHMENTS: MAG TAPE [ ] FLOPPY DISKS [ ] LISTING [ ] DECTAPE [ ] OTHER:

COULD THIS SPR HAVE BEEN PREVENTED BY BETTER OR MORE DOCUMENTATION? PLEASE EXPLAIN IN PROVIDED SPACE BELOW. YES [X] NO [ ]

CPU TYPE	SERIAL NO.	MEMORY SIZE	DISTRIBUTION MEDIUM	SYSTEM DEVICE	DO NOT PUBLISH
11/23	AB02254	128K	RX02	RL01	[ ]

Monitor subroutines should be as free of side effects as possible and should work in a manner that makes sense. There is a problem with the MTPS and MFPS monitor subroutines in the XM monitor. The problem cost me three weeks of wasted time because I assumed that a monitor subroutine to change the PSW would always work.

The problem with the MTPS routines is that it is a monitor subroutine and is run in user mode. If the user has mapped away the I/O page, then the MTPS routine will fail if the code moves the new priority to the hardware address of the PSW. This limitation is hinted at in the manual, but it indicates that it is only for virtual jobs. It really should read that the MTPS routine cannot be used anytime the I/O page is mapped away by the user and that probably means that it cannot be used with FORTRAN virtual arrays in the XM monitor.

There is another more basic problem with the MTPS subroutine as it relates to the memory management hardware. According to the hardware book, the MTPS instruction cannot affect the processor priority bits while in the user mode. Only when in the kernal mode can the processor priority bits be changed. This means that if the monitor subroutine uses an MTPS instruction and is called in user mode, then the processor priority bits will not be changed and the user will only think that he has changed them.

I believe that the MTPS routine in the monitor should be changed to some sort of TRAP or EMT so that the monitor can field the change in kernal mode. I feel that the MTPS routine in the monitor should be able to change the PSW no matter what the user mapping is due to the fact that the I/O page may be mapped and the MTPS instruction won't change while in user mode. I had to implement a user TRAP to change my priority, which is no big deal, but I had assumed that the monitor would be able to change the priority for me and that was the last place I looked in solving my user problem. Let Greg fix it.

RT-11 Version 5

The following is an overview of some of the PROPOSED features for RT-11 Version 5. None of them are definite yet but the information did come (indirectly) from the development group. There is no official date for the release of V5 but a sensible guess might be spring/summer 1983.
It should be noted however that the RT-11 development group also have to answer all SPR's. If V5 arrives late then I suggest that you complain to Dave Walesby.

XM Enhancement:

022 Support (up to 4MB) BUT UNIBUS MAP (i.e. 11/24) will NOT be supported.
RUN a Virtual Job (from other than SY:).
Privileged Foreground Jobs with Virtual Overlays.
VM: handler to be shipped with distribution. (allows extended memory to be used as a block replaceable device. 022 support.)

Logical Disk Subsetting:

Ability to split large disk devices into several logical units, by using one large RT-11 file as a device within a device.

User Defineable Commands:

Ability to define DCL-like commands with options.
Add/Rename options for existing DCL commands.
Simple command definition procedure.

Control File Processor:

IND indirect command file processor to be shipped with distribution. A superset of the RSX-11M IND and allows string substitution and File operations.

Console Logging:

Console I/O can be logged to disk or lineprinter. Useful if console is a VDU or for logging IND output - alternative to BATCH.

KMON Line Editor:

Line and Last Line editor using KED command subset. available to programs via (.TTYIN, .MTTIN, .GTLIN).

New Programmed Requests:

.FPROT - Set/Reset File protection status.
.SFDAT - Set File creation date.
.ABTIO - Abort outstanding I/O for Job.
.CSTAT - In SJ Monitor.

Write Protect for Floppy Handlers:

i.e. SET [NO] WRITE for DX: and DY:

Variable Retry Count for handlers:

DX, DY, DD, DM, D1, RK handlers

Error Logging:

Supported in SJ ???
SET dd: [NO] SUCCESS controls logging of successful operations.

Swapping:

SET KMON [NO] SWAP command presumably similar to USR [NO] SWAP .

PIP:

COPY/ [NO] PROTECTION	set/remove protection.
COPY/ SINCE:date	just files since date.
COPY/ DATE:date	just files from date.

DIR:

DIR / [NO] PROTECTION	dir of just protected files or just unprotected files.

QUEMAN:

PRINT /LOG	as COPY command.
PRINT /SINCE:date	print filessince date.
PRINT /DATE:date	print files from date.

QUEUE:

Protects file while despooling.
File by File output to RT-11 File structured devices.

LINK:

Global Cross Reference map.
Routines allowed in multiple overlays (not pulled into root).
XM Enhancements.

MACRO:

Completes assembly if CREF fails.
Compatible error messages?

FORMAT:

Allows formatting while FG active.

FILEX:

/WAIT option.

SLP:

Allows checksums for source patches.

BINCOM:

Has wildcard comparison.
Device to device comparison.

BACKUP:

New utility allows backing up of large files to multiple volumes of smaller media, and vice-versa.
COPY/MULTIVOLUME ?

SYSGEN:

IND command files, allow:
High speed ring buffer support.
DZ-11 baud rates to 9600.
Remote Console option.
Console logging option.
KMON line editor option.

Unsupported Software:

TTYSET - set terminal characteristics (Multi terminal systems).
MTYSET - set auto baud terminals (Multi terminal systems).
DATIME - Request date and time at startup and chain to indirect file.

Last but not Least -

Documentation:

System users guide to be split into Keyboard command manual and utilities manual.

Customer Installation guides will be configuration specific?

An RSX-like Mini Reference instead of Pocket Guide (presumably more useable than the RSX guide).

RT-11 Quick Quiz - Answers

Question 1.

a) COP RK1:FILE(1,3).FOR RK2:

b) COP/C RK1:FILE%.FOR TEMP.*

Question 2.

a) You can't use wildcards like this.

b) No such TT option as LENGTH.

c) SINCE not a valid switch for COPY.

d) Wild cards not supported by FORTRAN.

e) /ORDER:ALP not legal, use /ALP.

f) Options must FOLLOW filename.

Question 3.

You Can't. (oh yes you can - Ed).

Question 4.

Option a is correct - 2 Marks. 1 Mark for option c.

Question 5.

a) legal.

b) legal.

c) legal.

d) illegal (F/B only).

e) legal.

f) illegal (F/B only).

g) illegal (no such request).

h) legal.

Question 6.

a) NO.

b) NO.

c) YES.

d) NO.

e) YES.

f) YES.

g) NO.

h) NO.

Question 7.

a) YES.

b) NO.

c) NO.

d) YES.

e) YES.

Question 8.

Copy LS.SYS to some other unique 2-character device name, install the handler and refer to the second printer by the new name. Use SET command to change vector and CSR assignments.

Question 9.

The "^W" command in TECO V29 is changed to just "W" in TECO V36. The distributed version of VEQ.TEC was not ammended to take account of this.

Question 10.

MACRO.SAV is not present on the system device. If TEST.MAC were not found, you would get the message:

```
?MACRO-F-File not found
```

Question 11.

Knee Him/Her in the Groin - 3 marks.

Any other answer - 2 marks.

Fall '82 DECUS Symposium

AIRPORT TRANSPORTATION

Luxe Livery Service, Inc. has offered to provide our attendees Airport transportation to and from the Los Angeles Airport at a discounted fee. The total cost for our attendees will be fifteen dollars ($15.00) each way excluding driver gratuity.

You can arrange to be met at the airport outside your airline by a uniformed driver who knows your name and where you are going. The driver will load your baggage and take you directly to your hotel.

FROM LOS ANGELES AIRPORT

After claiming your baggage please proceed to the center island, across from the baggage claim area. The van service is only permitted to park to load passengers. If you do not see Luxe Livery's green van in your area, watch for them in traffic - they may be circling the airport. To be recognized by the driver detach and wave the green portion of your transpass as he approaches. In the event you do not make contact with your driver within 10 minutes, please phone (800) 422-4267 or (714) 558-1413 and ask to speak with the dispatcher.

TO LOS ANGELES AIRPORT

Call (714) 558-1413 after 7:00 p.m. the evening before service to confirm your reservation and pick-up time. Meet the van at the hotel's main entrance. Be prompt! The drivers are not permitted to wait more than 10 minutes. Remember, Airport check-in lines are usually quite long.

SPECIAL NOTES:

1. Acquire a transpass from the DECUS office. A special mailing will be done for all preregistered attendees and DECUS Leadership. Anyone else wishing to recieve the transpass must call the DECUS Office at (617) 467-4875.

2. Each attendee is required to have their own transpass. Therefore, when requesting transpasses from the DECUS Office please state the correct number you need.

3. Phone Luxe Livery Service and make an advanced reservation. The phone number for outside California is (800) 854-8171 and within California (800) 422-4267.

4. The transpass must be presented to the driver at the time of transfer to be eligible for the discounted rate.

5. The transpass has no value in itself, cannot be bought or sold, and must be used in conjunction with CASH ONLY at the time of transfer.

6. Please pass this information on to your SIG/LUG members so they can also benefit from this offering.

DIGITAL EQUIPMENT COMPUTER USERS SOCIETY
ONE IRON WAY, MR02-3/E55
MARLBORO, MASSACHUSETTS 01752

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 129
NORTHBORO, MA
01532

MOVING OR REPLACING A DELEGATE?

Please notify us immediately to guarantee continuing receipt of DECUS literature. Allow up to six weeks for change to take effect.

( ) Change of Address
( ) Delegate Replacement

DECUS Membership No.: _____

Name: _____

Company: _____

Address: _____

_____

State/Country: _____

Zip/Postal Code: _____

Mail to: DECUS - ATT: Membership
One Iron Way, MR02-3/E55
Marlboro, Massachusetts 01752 USA

Affix mailing label here. If label is not available, print old address here. Include name of installation, company, university, etc.