RSX

MULTI-TASKER

MARCH 1985 ISSUE

RSX Multitasker

Table of Contents

POS Futures

Terry Medlin

Many of you use POS. I know that from talking to you at the symposia. I had several chats with some of the POS product managers while at Anaheim and they would like to create a better relationship with the RSX SIG. The Executive Committee discussed this option and voted unanimously to work with them. As many of you know, POS and the PRO has had a roller coaster ride from a marketing perspective. They now are gearing PRO/POS as a single user PDP in many ways and more changes in this area are underway! Since POS was derived from M-PLUS and since they finally removed the mandate of the menu (YEA!!!), we think the similarity of the products has returned to where it should have always been. For that reason, you will see an expansion of many of our sessions in the future.

Now I would like to ask some of you to contribute articles on POS usage to Dom for inclusion in the newsletter. There are many people out there waiting for just your article.

The Bag of Tricks: MACRO-11

Bruce R. Mitchell
Machine Intelligence and Industrial Magic
PO Box 601
Hudson, WI 54016

This column covers MACRO-11 bag-of-tricks routines, as stated in last month's issue of the Multi-Tasker. It will appear as space permits. All MACRO programmers are encouraged to submit their favorite routines to the Multi-Tasker so that these useful, interesting, or just plain bizarre tricks can be put out before the SIG in general for the admiration and edification of all.

In this month's column, we again have something which the RSX Implementers apparently never thought would be that useful to end-user coders: A phase of the moon routine.

Now let's face it, we all know that whether or not a program runs when you give it to an end user is dependent on the user's attitude, how long it has been since Field Service did a PM on the machine, what day of the week it is, and the phase of the moon. You can't do much about the user's attitude or Field Service, but it is well to take into account the phase of the moon - again, a much desired feature for doing output page headers.

This routine was taken from the DECUS C package, compiled, disassembled, cleaned up, modified and recommented for use in other MACRO programs. C users may note that much of the conditionalization in the original has been deleted to make this version more compact; this places a limitation on the routine of not running correctly past the year 2200. Those who feel that this is a severe drawback are invited to look at the original code.

Because the author tends to separate data and code structures in his programs, the labels have been made so that the data and code can be readily separated within a single source file.

The output from PHASEM is a pointer in R0 to an .ASCIZ string containing the name of the day of the week. This is convenient for use of $EDMSG or simply for copying into a target field with a MOVB - TSTB - BNE loop.

```
        .SBTTL  PHASEM  Identify Phase of Moon

;  PPPPPPPPP   HH      HH     AAAA      SSSSSSSSS  EEEEEEEEEE  MMM    MMM
;  PPPPPPPPP   HH      HH     AAAA      SSSSSSSSS  EEEEEEEEEE  MMM    MMM
;  PP      PP  HH      HH    AA  AA    SS          EE          MMMM  MMMM
;  PP      PP  HH      HH    AA  AA    SS          EE          MMMM  MMMM
;  PPPPPPPPP   HHHHHHHHHH   AA    AA    SSSSSSSS   EEEEEEEE    MM  MM  MM
;  PPPPPPPPP   HHHHHHHHHH   AA    AA    SSSSSSSS   EEEEEEEE    MM  MM  MM
;  PP          HH      HH  AAAAAAAAAA          SS  EE          MM      MM
;  PP          HH      HH  AAAAAAAAAA          SS  EE          MM      MM
;  PP          HH      HH  AA      AA          SS  EE          MM      MM
;  PP          HH      HH  AA      AA          SS  EE          MM      MM
;  PP          HH      HH  AA      AA   SSSSSSSS   EEEEEEEEEE  MM      MM
;  PP          HH      HH  AA      AA   SSSSSSSS   EEEEEEEEEE  MM      MM


;  PHASEM - Identify Moon Phase of Current Date
;
;  This subroutine returns a pointer to an ASCIZ string containing
;  the current phase of the moon.  Adapted from the C routine used
;  in the DECUS C package.
;
;  Inputs:  None
;
;  Outputs:  R0 - Pointer to ASCIZ string
```

```
;
;  Register dispositions:  R0 destroyed
;
;  Variable dispositions:  TIMBUF buffer modified


;  GTIM$ data buffer

TIMBUF:     .BLKW   8.


;  Phases of the moon name table

PH0:        .ASCIZ  ®New®
PH1:        .ASCIZ  ®Waxing crescent®
PH2:        .ASCIZ  ®First quarter®
PH3:        .ASCIZ  ®Waxing gibbous®
PH4:        .ASCIZ  ®Full®
PH5:        .ASCIZ  ®Waning gibbous®
PH6:        .ASCIZ  ®Last quarter®
PH7:        .ASCIZ  ®Waning crescent®
   .EVEN


;  Day of the week pointer table

TXTTBL:     .WORD   PH0, PH1, PH2, PH3, PH4, PH5, PH6, PH7


;  Cumulative days in the year to month start table

INDYYR:     .WORD   -1                             ; Invalid and never used
   .WORD   -1                              ; Days in year to January 1
   .WORD   30.                             ; ... February
   .WORD   58.                             ; ... March
   .WORD   89.                             ; ... April
   .WORD   119.                            ; ... May
   .WORD   150.                            ; ... June
   .WORD   180.                            ; ... July
   .WORD   211.                            ; ... August
   .WORD   241.                            ; ... September
   .WORD   272.                            ; ... October
   .WORD   303.                            ; ... November
   .WORD   333.                            ; ... December


;  Subroutine code

PHASEM:     JSR     R5, .SAVR1                 ; Save registers 1 - 5

   GTIM$S  #TIMBUF                    ; Get system time into time buffer

;  Calculate the century number = (Year / 100.) + 1
```

```
        MOV     TIMBUF+G.TIYR, R1           ; Load year into R1 for later use
        ADD     #1900., R1                  ; Change year past 1900 to the real year
        MOV     R1, R5                      ; Load year into R5 for immediate use
        CLR     R4                          ; Zero high order word
        DIV     #100., R4                   ; Divide by 100
        INC     R4                          ; And add 1 to give century number
        MOV     R4, R5                      ; Save it into R5 also

;       Calculate a temporary partial = INT((Century * 3) / 4)

        ASL     R4                          ; Century * 2
        MOV     R4, R3                      ; Copy it into R3 for later use
        ADD     R5, R4                      ; Century * 3
        ASR     R4                          ; / 2
        ASR     R4                          ; / 4, result in R4

;       Calculate a temporary partial = INT(((Century * 8.) + 5) / 25.)

        ASL     R3                          ; Century * 4
        ASL     R3                          ; Century * 8
        ADD     #5, R3                      ; + 5
        CLR     R2                          ; Clear upper word for divide
        DIV     #25., R2                    ; / 25, result in R2

;       Calculate the year in the Mentonic cycle = (Year MOD 19.) + 1

        CLR     R0                          ; Clear upper word for divide
        DIV     #19., R0                    ; MOD 19
        INC     R1                          ; Take remainder and add 1
        MOV     R1, R0                      ; And load it into R0 also

;       Calculate a temporary partial = 11 * Mentonic

        MUL     #11., R1                    ; 11 * Mentonic, result in R1

;       Calculate moon's age on 1-Jan = ((sum of partials) + 27.) MOD 30.

        MOV     R4, R3                      ; Load first partial
        ADD     R2, R3                      ; Add second partial
        ADD     R1, R3                      ; Add third partial
        ADD     #27., R3                    ; Add fixed value of 27
        CLR     R2                          ; Zero upper doubleword
        DIV     #30., R2                    ; MOD 30., result in R3

;       Apply a correction in two special cases

        CMP     R3, #24.                    ; Was moon's age 24 on 1-Jan?
        BEQ     10$                         ; If so, do correction

        CMP     R3, #25.                    ; Was moon's age 25 on 1-Jan?
        BNE     20$                         ; If not, skip correction

        CMP     R0, #11.                    ; And was the Mentonic year > 11?
```

```
   BLE     20$                         ; If not, skip correct

10$:        INC     R3                          ; Apply a correction day

;  Now add number of days in the year up to today to 1-Jan moon's age

20$:        MOV     TIMBUF+G.TIMO, R2           ; Load month into R2
   ASL     R2                          ; Multiply by 2 for dispatching
   ADD     INDYYR(R2), R3              ; Cumulative days in year to month 1st
   ADD     TIMBUF+G.TIDA, R3           ; R3 is now days in year to current

;  Leap year correction for years evenly divisible by 4

   BIT     #3, (SP)+                   ; Is it a multiple of 4 (leap year)?
   BNE     10$                         ; If not, skip around

;  Leap year; special treatment for months past February

   BIT     #3, TIMBUF+G.TIYR           ; Is the year evenly divisible by 4?
   BNE     30$                         ; If not, no leap year correction

   CMP     #4, R4                      ; Is the month past February?
   ADC     R5                          ; If so, add the carry to R5

;  Compute the phase = ((((moon's age * 6) + 11.) MOD 177.) / 22.) MOD 8.

30$:        MUL     #6., R3                     ;  * 6
   ADD     #11., R3                    ;  + 11
   CLR     R2                          ; Clear high order for divide
   DIV     #177., R2                   ;  MOD 177
   CLR     R2                          ; Clear high order for divide
   DIV     #22., R2                    ; / 22
   BIC     #177770, R2                 ;  MOD 8
   ASL     R2                          ; Multiply by 2 for table offset
   MOV     TXTTBL(R2), R0              ; And now it points to the text

   RETURN                              ; Return to the caller


   .END
```

TEM Update (DECUS Library 11-750)

Thomas R. Wyant
E. I. DuPont de Nemours
TFD - P.O. Box 27001
Richmond, VA 23261

It's nice to know that when you submit a package to the DECUS* library someone may actually buy it. An abstract for my terminal emulator package (called TEM) was published in the November 1984 multitasker, and already I have feedback. Of course, I could wish the feedback didn't contain bug reports, but I guess we can't have everything. On January 10, I requested the library to hold all orders, and submitted a revision with the following bugs fixed:

1. The /REMOTE switch won't change the sense of the line setting. For example, you can change the answer speed of a remote line, but not set it /NOREMOTE. The recommended workaround is to use the MCR SET /[NO]REMOTE command to get the desired setting.

2. TEM would attempt to set /SPEED and /REMOTE on DL-11s and DJ-11s. The workaround is not to use these switches when communicating with these interfaces. I have attempted to correct this bug, but have no way to see if the fix works.

3. TEM didn't cancel control/s on initializing with the remote line. If you don't have software to combat the analogous problem with LA180s, the only workaround I can think of is to call the modem and manually issue control/q.

4. TEM sometimes refuses to abort. The documented procedure (using OPEN) will work, but the exit code has been beefed up.

5. HAYSKI.TEM comes off the distribution with the wrong implied carriage control. More on this later.

In addition to the fixing of the above bugs, the January 1985 version of TEM has the following enhancements:

1. A single-argument version of the /SPEED switch.

2. A /TRANSLATE switch similar to /ASSIGN, but for inbound characters.

3. Support for parity generation and checking. This involved renaming the /PAUSE switch to /DELAY, since I wanted two character abbreviations and couldn't think of another word for

/PARITY.

4. Changed the default /MAXBUFFER setting to /NOMAXBUFFER, and the default /RPA setting to /RPA. Not a big deal, unless you've worked with the previous defaults.

5. Added a /STOP switch, to allow TEM to suspend itself and let you poke around on the local system (eg: to find that file you were going to transmit, but whose name you forgot).

6. Attempt to conditionalize for use with RSX-11M V3.2. I have no way to try this out, though.

There are a couple of problems with the distribution, generally because I didn't take sufficient account of what FLX would to to the files. In particular, the file names HAYSINI.TEM and HAYSKIL.TEM got truncated to HAYSIN.TEM and HAYSKI.TEM. You will have to correct this if you intend to use the distributed setup and shutdown files. Also, the text file HAYSKI.TEM provided to clear a Hayes* Smartmodem* needs to have no implied carriage control. Unfortunately, there doesn't appear to be any way to pull it off the distribution without it acquiring "LIST" carriage control. To get around this, you can create an empty file with no implied carriage control, and append the distributed file to it (ignoring PIP's protests). For example:

```
>RUN TEM
TEM>NL:/SE:NL:/LO:HAYSKIL.TMP/-EO/TE/EC/CO/EX/-LI
TEM>©Z
>PIP HAYSKIL.TMP/EOF:1:0
>PIP HAYSKIL.TMP/AP=HAYSKIL.TEM
PIP -- Input files have conflicting attributes.
>PIP HAYSKIL.TEM/RE/NV=HAYSKIL.TMP
```

My excuse for the above problems is that the files were only intended as samples. The above will (I hope) render them more usable.

Several people (including me) have noted that if you are communicating with a Hayes* Smartmodem*, you need to set the Smartmodem*'s line /NOREMOTE under M+ V2.1 or M V4.1. If you don't, the TT: driver seems to autobaud on the first character back from the Hayes* (even if you have SET /NOABAUD), and set itself to some off the wall speed. DEC* (not having TEM) has failed to reproduce this problem. Also, the problem doesn't exist under M+ V2.0 or M V4.0. I don't know what the difference is. The bottom line is that if you want to use TEM with a Hayes* Smartmodem*, you may need to set the line /NOREMOTE in MCR (or in TEM, when and if you get the updated version in which the /REMOTE switch works).

Finally, thanks for the feedback. It's good to know people are interested enough in TEM to want it to work right.

The Hamster Theory of RSX

Bruce R. Mitchell
Machine Intelligence and Industrial Magic
PO Box 601
Hudson, Wisconsin 54016

1.0 INTRODESTRUCTION

The RSX-11 operating system has been described as a "finite state machine driven by external and internal events". This explanation is technically accurate, but is not particularly useful unless the recipient of the explanation is a Computer Science major.

A more user-comprehensible explanation of how RSX works has been needed for some time. There have been many requests for this article - yes, many people have even got down on their knees and begged - but I'm going to go ahead and do it anyway.

This view of RSX (long in the planning) presents RSX in a novel way which hopefully makes all the jargon surrounding the internals of the RSX operating system perfectly clear, even to persons with little or no technical background.

The RSX guinea pig (or, as it is more commonly known, the "kernel"), its associated white mice, hamsters, rats of various colors, and the other rodents associated with RSX are discussed and their interactions explained in an understandable way.

2.0 THE HOST COMPUTER (THE PET STORE)

It's difficult to talk about RSX without talking about the computers on which RSX runs, so let's take a quick look at the PDP-11 series of computers, without which there would be no RSX operating system.

DEC has been pushing the PDP-11 series of small computers for some time now. Since the first PDP-11 (the Model 20) was introduced in 1970, the 11-series has grown into a substantial family of machines.

* - "Hayes" and "Smartmodem" are (I presume) trademarks of Hayes Microcomputer Products, Inc. "DEC", "DECUS", "RSX", and any others I may have omitted are trademarks of Digital Equipment Corporation.

Of all the operating systems for the PDP-11 series, RSX is one of the most popular. One of the reasons for the popularity of RSX is its wide compatibility with the 11-series machines

- from minimal RSX-11S systems on LSI-11 series processors with no mass storage at all,

- to smaller Unibus systems such as the 11/04 with small disk drives running unmapped RSX-11M for limited scale applications,

- to midrange Unibus systems such as the 11/35, 11/45, 11/55 series running mapped RSX-11M for numeric processing, front-end applications, and general purpose systems,

- to the biggest, fastest Unibus member of the 11-series, the PDP-11/70 running mapped RSX-11M-Plus with megabytes of memory, big disks, and serving a variety of applications, with its revolutionary concept of separation of the two-word memory bus (the Cachebus) and the single-word I/O Unibus, increasing the speed of the processor manyfold and allowing the design and implementation of multiprocessor systems *

- and yes, even to the 32-bit VAX series of computers, running RSX-11M-Plus in compatibility mode for RSX applications being migrated to the VAX series.

Yes indeed, RSX is a very successful system, running on a wide variety of computers. And, as DEC proudly says - they've made the computer an industrial tool. Almost singlehandedly so, as a matter of fact. But to get back to the main topic ...

The three variants of RSX -- 11S, 11M, and M-Plus - run on a wide variety of PDP-11 computers.

Well, that's all fine and good. But that only tells us where RSX is today, not where it came from or what the original inspiration for RSX was. So, let's take a look at the background of RSX, which we like to think of as -

Our Software. Digging back through the dusty archives of ancient RK05s and RS04s, nobody is absolutely sure what gave the original inspiration for the RSX operating system. And I mean, I went way back to research this, almost back to the beginning; nay, even before the beginning, when nobody had even built a PDP-11, and the Declaration of DEC Dependence had just been signed.

---

* I like 11/70s.

There are some who think RSX has origins in the industrial environment. This view is somewhat supported by the recent excavations at the Mill Pond, which produced evidence of early, crude device drivers.

There are some who like to think that RSX developed out of early laboratory work with computers.

There are a sizeable number of users who think that RSX is the organic result of long nights of heavy research and long discussion in smoke-filled which finally culminated in the budding and flowering of a noble idea.

At any rate, nobody was quite sure until very recently, when early copies of RSX documentation were discovered. An early version of the RSX-11D System Logic manual first put me onto the track leading to this monumental effort.

Once this momentous discovery had been made, it was easy to pick up the trail through more recent documents. For example, the much lamented System Logic Manual, the most recent release of which is for RSX-11M Version 3.1. In this manual, seldom seen by other than system programmers and certified wizards, lies further confirmation of the early suspicions -

Yes, there it was, in black and white and brown, plain for all to see. RSX was written by a deranged pet store owner, then finally polished into its present shape. As a matter of fact, the approach worked so well that DEC applied it to the VAX/VMS operating system - where, unfortunately, it did not work quite so well as it did on the more elegant, orthogonal, user-friendly and just generally superior PDP-11 line; and so there, all you sneaky slimy VMS users, take that.

But what can you say? It was something that should have been expected.

At any rate, ladies and gentlemen, it is time that the truth is told. No longer shall we remain in the darkness of ignorance.

No more shall the thorny questions of why RSX works as it does trouble us. No more shall the pricklings of ignorance disturb us and cause us to withdraw in search of enlightenment. Mainly, no more shall the system managers be forced to run and hide when users come to them and ask - "Why?"

It is time, my friends, to proceed to the business at hand.

v v v v v v v v v v

>>>*****>>> The Business At Hand <<<*****<<<

© © © © © © © © © ©

For the purposes of this discussion, we will consider the PDP-11 to be a pet store.

Inside the pet store is both hardware and software. The hardware consists of the cages, the walls, the floor, the food trays, a large Habitrail and other items. The software consists of the inhabitants of the pet store - rodents of various types and, in some cases, miniature and even full-size swine.

To break the PDP-11 hardware up further, the computer is composed of a central processor (CPU), which we shall model as a running wheel in the middle of the store; main memory (core), which we shall consider as the store as a whole, with the exception of the running wheel; and external devices, which we shall model as trucks pulled up to the loading dock.

2.1 The Central Processor (The Running Wheel)

The central processor can best be described as a large exercise wheel in the center of the store. Only one rodent is allowed to run on the wheel at any given time, and certain rodents (such as the Exec) are allowed to kick other rodents off the wheel if they want to run.

This is where all the work is done.

2.2 Main Memory (The Inside Of The Store)

For all reasonable purposes, the entire interior of the pet store can be considered to be main memory. The area down near the floor is low memory; the area up near the ceiling is high memory. There are two interesting subareas of memory, however, which deserve special attention.

2.2.1 Vector Area (The Habitrail) -

Down on the floor of the pet shop is a big Habitrail, with bells on the ends of some of the tubes, and openings to a big cage full of mice on the other ends of the tubes. We call this the device vector area. The little bells are connected to the trucks (like disk drives and terminals), and they ring when a device wants service.

Some of the bells are connected to the door of the pet shop and to mousetraps scattered around the pet shop, so that when the owner comes in or one of the rodents gets loose, the offending rodent gets caught and a special bell rings. We call these trap vectors.

Most systems have one special bell connected to the clock on the wall. This bell rings 60 times a second, which is very fast indeed. This bell is called the clock vector. More on this later.

2.2.2 The I/O Page (The Shipping Dock) -

The I/O page is a special area at what would seem the very top of the pet store, up near the roof, but is actually out back. It's the shipping dock, where little green rabbit pellets arrive and are taken away by the trucks.

2.3 The RSX Operating System

With the hardware out of the way, let's now take a look at the RSX operating system and software as a whole, looking at it from a building block standpoint and working up to complex things from more simple ones.

3.0 THE EXECUTIVE (GUINEA PIG)

The entire RSX operating system is founded upon the RSX Executive, so let us now examine the Executive in light of our model.

The RSX Executive, or as it is fondly called by system hackers, "The Exec", is modeled in our pet store analogy by a large, fat, lazy guinea pig.

There may be some who ask, "Why a large, fat, lazy guinea pig?". But if you do, it is clear that you are not familiar with executives. Like most other executives, the RSX Exec likes to think that it is in control of the whole thing. You can form your own judgments about that. Actually, about all it does is determine who gets to run on the wheel next. To extend the executive analogy further, the Exec is jealous of its control and hates to give any of it up.

As a matter of fact, the Executive is so big and fat and jealous of other guinea pigs that there is only room for one in the whole pet store. Under recent releases of RSX, however, it has grown more tolerant and permits clones to live in the pet store too, although strictly in an inferior capacity.

3.1 What Does It Do?

Not a whole lot, unless waked up by one of the other rodents pulling a string to ring a bell on his cage. When this happens, the Executive runs out and gets it over with as fast as possible, then goes back to sleep. The Executive loves to sleep.

What the Executive does most of the time is wake up when the clock bell rings in his ear, push the hand on his cheap watch over one tick, then go back to sleep.

Once in a while one of the mice or hamsters will wake him up, griping and loudly demanding service. The Executive grumpily does what must be done to get the offending rodent off his case, then goes back to sleep.

The Executive <u>really</u> <u>really</u> likes to sleep.

3.2 Where Does It Live?

Unlike other executives, the RSX Executive lives in the low rent district of the pet store, right down on the floor where it can't see a damn thing going on. The only way he finds out about things is when somebody wakes him up.

3.3 What About Executive Commons?

Once upon a time, when men were real men, computers were real computers, small tasks in core were real small tasks in real core, and sheep kept their backs to the wall, the RSX Executive was all one piece. This was back in the good old days when one was apprised (by MCR) (in capital letters, of course), of "FILE PRESRV.SYS NOT FOUND".

Of late, in keeping with the Digital trend that "bigger is better", the Exec has grown so big that the original guinea pig had to be cloned. Digital calls these clones of the original guinea pig "executive commons". Users tend to call them a lot of things.

3.4 Why Should I Upgrade To M-Plus?

Because the Exec and other system structures have been and continue to become larger, it behooves any user who is able to do so to upgrade to RSX-11M-Plus. The M-Plus Executive offers many advantages which are not present under RSX-11M, such as increased use of main memory and disk, and the requirement of a 22-bit processor. (Hey, nobody said they were advantages to <u>users</u>, did they?)

As a matter of fact, Digital is so interested in seeing users upgrade to M-Plus and obtain these advantages that it has in the past (and probably will again in the future) given RSX-11M users a special price break on upgrading to RSX-11M-Plus. Digital calls this program The RSX-11M-Plus Upgrade Special. In light of our discussion here, it could equally well be given another, more appropriate name. *

4.0 POOL (THE BACK ROOM)

One hears a lot of talk about POOL, but somehow nobody ever explains what it is or what it is good for, why it runs out and why the system stops when it runs out. Since the Executive and POOL are related, let's now talk about POOL.

4.1 What It Is

POOL is the back room of the pet store, where all the supplies and raw materials, and most of the rabbit pellets, are stored. Since the back room is not very large, there is only a limited amount of POOL and when that runs out, that's the end of it.

4.2 What It Is Good For

The whole pet store depends on the back room. Pellets come in and go out through the back room, and are stored here in a big feed bin when not in use. The back room contains material to build new tubes for the Habitrail. Trucks are logged in and out through the back room. The back room, as always, is where all the important and useful stuff is stored.

4.3 What Happens When It Runs Out

Nothing. Everything comes to a screeching halt, until the owner kicks whoever is on the wheel off and cleans out the whole store.

---

* You had to be there to understand why that section is funny.

5.0 DEVICE DRIVERS (MICE)

The Executive is assisted in his work by mice of various colors. We call these "device drivers".

5.1 What Do They Do?

The Executive is too big and fat and lazy to fit into the Habitrail where the device bells are located, much less to work that hard, so he rides herd on a flock of trained mice who do all the dirty work. These mice are really stripped down parts of the Exec; small, fast and not too smart.

When a bell rings, the mice run in and out of the Habitrail and very often up to the I/O page, to deliver little green rabbit pellets (Queue I/O packets). Only one mouse to a bell of course (to prevent fights), but one mouse can handle several bells. These multi-bell mice, of course, are bigger than the other mice and have to think a little bit harder when one of their bells rings.

The mice get the rabbit pellets from the feed bin, delivering them to and from the shipping dock and to and from hamsters. It all depends on the direction of the transfer - logical writes are pellets from hamsters to the shipping dock, and logical reads are pellets from the shipping dock to hamsters.

Most of the time, the mice have to deliver the pellets themselves. Some of the very smartest mice, however, are permitted to deliver pellets directly to the cages of other inhabitants with little blowguns. We call these mice "DMA device drivers". Other mice are not permitted to use blowguns, because they're not smart enough and somebody might get hurt. We call these mice "programmed I/O device drivers". Still other mice aren't even permitted out of their cage, because they don't have a bell, aren't smart enough to know it, and might get lost. We call these mice "pseudo-device drivers".

Some people insist on viewing the driver mice as part of the Executive. There is no doubt a certain kinship between them - they all love to sleep - but most system programmers view the mice as a separate entity.

5.2 Where Do They Live?

Most of the device driver mice live in a communal cage, conveniently located just above the floor, and directly connected to the Habitrail in a number of different places. These mice have permanent connection tubes in the Habitrail - we call the connections "device databases".

5.3 What About Loadable Drivers?

Some of the loadable driver mice live in other pet stores, arrving at the shipping dock and only coming in when there is a special job of work to be done. A special gerbil lets them into the store and builds a connection tube for them in the Habitrail from raw materials in the back room. When they have finished their job, a different special gerbil lets them out of the store.

Once you bring in these loadable driver mice, though, you can't get rid of their connection in the Habitrail. Obviously, if you bring in too many of these mice, eventually there are no raw materials left in the back room and everybody suffers.

Here we see a typical loadable driver being loaded into DRVPAR. In some systems, it would be loaded into GEN instead. *

5.4 The Terminal Driver

The terminal driver mouse has all the frills - hair, a tail, whiskers and everything. Not all the other mice do. It is therefore bigger and smarter than most of the other mice. In actual point of fact, it is a mouse and a half, having to handle many bells.

Being a mouse and a half, it is slower than most of the other mice, particularly when his bell rings once per character (interrupt per character devices) as opposed to once per line (DMA devices). When there are non-DMA multiplexors (bell rings once per character, but only one bell for many lines) all hell breaks loose.

The terminal driver has his own private little back room where he stores the little green rabbit pellets. If he runs out of room there, he uses the store's back room.

Because the users can make the terminal driver mouse go through the Habitrail in so many different ways, it is best just to ignore it and hope that it continues to work.

5.5 I/O Rundown

Sometimes, when a task is exiting, it may have I/O active on one or more drivers. The driver mouse, not being too smart, may become so confused that the task trying to exit just sits there and can't leave, because all the little green rabbit pellets are closely accounted for by the guinea pig (who is a hoarder, among other things). When this happens, the situation is called an I/O

* Ditto.

rundown hang. It is a nasty situation indeed, and when it happens, about all that one can do is close up the pet store, clean out the back room, turn out all the lights and reopen in a few minutes. In other words, reboot.

6.0 ANCILLARY CONTROL PROCESSORS (RATS)

The mice are assisted in their work by rats of various colors. We call these rats "ancillary control processors", or ACPs.

6.1 What Do They Do?

Just as the Executive is too big and fat and lazy to fit into the Habitrail, the trained mice are sometimes not smart enough to know what to do with the rabbit pellets once they get them. This is because many of the hamsters are gourmets, and want their pellets done just so. When this is the case, the mice turn the pellets over to ACPs, which grind them up or re-pelletize them according to who is doing the asking and how they want the pellets done.

The disk driver and magtape mice are particularly fond of the ACP rats, and, like the American Express card, are seldom seen without them. The slogan of the ACP rats, in turn, is "A Piece of the Action".

There are white (F11ACP) and black (MTAACP) rats. Never the twain shall meet. The rats all ignore each other, except on M-Plus, where they don't, at least not always, sometimes.

The ACP rats are primarily useful because many of the hamsters in the pet store are stuck up and don't like to talk to the mice, considering them to be too dumb. The hamsters spend most of their time thinking about important things and don't like to interrupt their thoughts to speak to a lowly mouse, especially if it would take some effort to find out what the mouse had to say.

6.2 Where Do They Live?

The ACP rats run all over the place, except down near the floor where the Exec and drivers live. Sometimes the rats have their own special cage, built for them by the owner of the pet shop. You only see this in bigger pet shops, though, in general.

---

* Same here.

6.3 What About User-Written ACPs?

Once in a great while, one sees a rat of a different color. It is always surprising to see a green rat (or whatever) running about, and it is widely rumored that these rats are all descendants of the very first Stamerjohn trained wild rat. You don't see very many of them, though, because training a rat is not a trivial matter.

7.0 SYSTEM TASKS (GERBILS)

In most pet shops, there are special gerbils which handle certain very specific jobs. There are not a great number of these. We call these system tasks.

7.1 What Do They Do?

The gerbils handle all sorts of special tasks which somehow the other rodents never get around to doing. They connect and disconnect bells from the hamster cages, for one thing (some people call that loading and unloading tasks). If one of the hamsters goes crazy, they lock it in its cage (parity memory fixer task). They check to see that the feed hopper is not running low (pool monitor). They do all manner of things.

Probably the most familiar gerbil is PIP, who knows all the mice and ACPs in the system on a first-name basis, and loves to talk to them.

7.2 Where Do They Live?

The gerbils bound all over the place like kangaroos. Some of them are insatiably curious and run all over the place all the time, looking for something interesting, like the pool monitor gerbil. Others are perfectly content to lay around until somebody wakes them up, even to the extent of running out and hiding in a truck (checkpointable tasks).

Hamsters may come and hamsters may go, but gerbils are forever.

8.0 GARDEN VARIETY USER TASKS (HAMSTERS)

8.1 What Do They Do?

The hamsters like to think that they run the system, and that it is there for their own personal use. They are the philosophers of RSX, mulling over such weighty matters as "The temperature on 4th floor is going up; should I turn on the air conditioning or ring the fire alarm?", "What happened to the last 6 months of data?", and "Who put the bop in the bop-she-wawa?"

8.2 Where Do They Live?

The hamsters, by and large, all live up in cages on the walls of the pet store. For reasons best known to the Implementers, there is a big sign on the walls of the pet store which says "GEN". Many of the pet store owners have wondered about this sign. Some have tried to paint over it, as a matter of fact. But whenever they do, the hamsters stop working. It's apparently some sort of religious thing with the hamsters.

8.3 Round Robin Scheduling

Most of the time, more than one hamster wants to run on the wheel. As long as all the hamsters who want to run on the wheel are of different ranks, no problem; the boss hamster gets to run until he gets tired, then the vice-president hamsters, then the middle management hamsters, etc etc etc.

Once in a while though, a bunch of hamsters will all try to pull rank on each other. When this happens, the guinea pig (who, liking to sleep, doesn't like the squabbling) arbitrates who gets to run next. The clever hamster, finding himself in this position, will be very quiet so he can run a long time before having to go back to his cage.

8.4 Vacations (Checkpointing)

From time to time, one of the hamsters will decide to go on vacation. This usually happens when the pet shop is getting crowded, there are lots of hamsters about, and lots of them are out queued up for a run on the wheel, and even more want to get out and run on the wheel. These hamsters, usually the bigger and slower ones, will temporarily run into one of the trucks to get away from it all.

Here we see a typical checkpointed task, waiting for an opportunity to get back in and run again. *

8.5 Where Do They Come From?

Well, this is naturally a very touchy topic and probably not fit to discuss in mixed company. Suffice it to say, when a programmer and an RSX system love each other very much (and believe me, they have to love each other very much to even consider it), sometimes a new task is the result, but usually only after a long wait.

This is so embarrassing to talk about, I'm just going to refer everyone to a good book on the topic. I mean really, nobody talks about taskbuilding in public.

9.0 OTHER USER TASKS (NOCTURNAL RODENTS)

Many RSX systems have tasks which don't fall easily into one of the categories above. These are shy, retiring tasks which only pop up and make themselves known on unusual occasions, so we model these tasks as nocturnal rodents.

You can't say a whole lot about these rodents, because you don't see a lot of them and they're all different. Here are two of them, caught in the act of sneaking out onto the running wheel:

The user monitor, a shy and retiring but nonetheless inquisitive task, * and

The online pool analyzer, here shown inspecting an I/O packet.

10.0 DECNET (MINIATURE SWINE)

DECnet is modeled in our pet store analogy by a miniature swine. That's a little pig.

There may be some who ask, "Why a little pig?". But if you do, it is clear that you have never run DECnet on an 11/35-based RSX-11M system with dual RK05s and 96K of memory.

Here is a typical DECnet executive hard at work. *

---

* Oh, you guessed already?

---

* This one got a BIG laugh, but not by intent.

Like the Executive, DECnet is so big and fat and jealous of other pigs that there is only room for one in the whole pet store. The Exec and DECnet fight over the choices spots on the floor, as a matter of fact. The Exec has an advantage over DECnet, though: DECnet can be unloaded.

10.1 Where Does It Live?

Like the 500 pound gorilla, wherever it pleases, as evidenced by the pig tracks all over the place.

10.2 What Does It Do?

Just like the Exec, not a whole hell of a lot, unless waked up by one of the other rodents pulling a string to ring a bell on his cage. When this happens, DECnet runs out and gets it over with as fast as possible, then goes back to sleep. DECnet loves to sleep, when it isn't fighting the hamsters for more space or the Exec for more pool.

It is rumored that the actual purpose of the DECnet swine is to answer the phone when another pet shop calls up, and to handle the call when calling up another pet shop. This has been hotly debated by owners of 11/34s for some time, who prefer to believe that the actual purpose is to confuse the pet shop owner.

Enough said. The "Pigpen Theory of DECnet" is in the works for next year.

11.0 DATATRIEVE (FULL SIZE SWINE)

Datatrieve is modeled in our pet store analogy by a full size swine. Anyone who has used it will understand.

Shown here is a typical Datatrieve, along with a copy of EDT Version 3.0, peacefully coexisting on an RSX-11M system with 96K of main memory. *

There are those users and system managers who claim that more than Datatrieve can be found in the pet shop at once. Maybe so, but about all you'll see them doing is running in and out from the trucks in the back. Here, for example, is another Datatrieve peacefully occupying (much the same way that East Berlin is

---

* This one got a bigger laugh.

---

* This was a real funny slide, but not such a big laugh.

peacefully occupied), peacefully occupying an 11/70 with 576K of main memory. Note the efficient use of system resources. Almost all of the system resources, as a matter of fact.

12.0 CONCLUSION

Well, hopefully that makes the internals of RSX clear to everyone who didn't understand them before.

In closing, let me leave you with a few thoughts on RSX to ponder:

```
+---------------------------------------+
¶                                       ¶
¶          "E Unibus Unumword"          ¶
¶                                       ¶
¶              - TPG slogan             ¶
¶                                       ¶
¶                                       ¶
¶        "E Cachebus Twoumwords"        ¶
¶                                       ¶
¶            - 11/70 designers slogan   ¶
¶                                       ¶
+---------------------------------------+
```

```
+---------------------------------------+
¶                                       ¶
¶     "RSX doesn't need VAX LUsers!     ¶
¶      RSX needs more tough wizards     ¶
¶    who will get out there and CODE!   ¶
¶                                       ¶
¶           - SIG chairman, Fall 1984   ¶
¶                                       ¶
+---------------------------------------+
```

which is something I think we should all keep close to our hearts. Gentiles and ladleman, thank you for your attention. This concludes the technical paper on "The Hamster Theory of RSX".

DECUS

DECUS SUBSCRIPTION SERVICE
DIGITAL EQUIPMENT COMPUTER SOCIETY
249 NORTHBORO ROAD, (BPO2)
MARLBORO, MA 01752

MOVING OR REPLACING A DELEGATE?

Please notify us immediately to guarant
tinuing receipt of DECUS literature. Allo
six weeks for change to take effect.

( ) Change of Address
( ) Delegate Replacement

DECUS Membership No.: ____________
Name: ____________
Company: ____________
Address: ____________
____
State/Country: ____________
Zip/Postal Code: ____________
Phone No.: ____________

Mail to: DECUS - Attn: Subscription S
249 Northboro Road, (BPO2)
Marlboro, MA 01752 USA

[redacted]

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

Bulk Rate
U.S. Postage
PAID
Permit No. 18
Leominster, MA
01453