THE mini-tasker

DECUS
RT-11 SIG NEWSLETTER

MAY 1983 VOLUME 9 - 2

Contributions to the newsletter should be sent to:

Ken Demers
MS-44
United Technologies Research Center
East Hartford, Conn. 06108
(203) 727-7139 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
MRO2-1/C11
Marlboro, Mass. 01752
(617) 467-4141

TABLE OF CONTENTS

FROM THE EDITOR

USER INPUT

TSX

DECUS LIBRARY

It is assumed that all articles submitted to the editor of this newsletter are with the authors' permission to publish in any DECUS publication. The articles are the responsibility of the authors and, therefore, DECUS, Digital Equipment Corporation, and the editor assume no responsibility or liability for articles or information appearing in the document. The views herein expressed are those of the authors and do not necessarily express the views of DECUS or Digital Equipment Corporation.

FROM THE EDITOR

This is the next to last issue of the "Mini-Tasker" that you will receive for free. I have always felt that the "Mini-Tasker" is priceless due to the information it makes public that is not obtainable anywhere else in print. However, if you are wondering if a yearly subscription to the "Mini-Tasker" is really worth $12 ($8.33, $7.50 etc.), the following is a list of the immediate improvements you will see in the newsletter.

1. There will be a published schedule of the dates you will receive your issues. The tentative 1983-84 schedule is September 1983, December 1983, March 1984, and June 1984. That is the minimum number of issues you will receive. It is more likely that six issues will be published (based on the current amount of material being received from RT-11 users).
2. The future newsletters will contain articles developed from notes written by the newly formed Symposia scribe service. This service will allow local journalism and computer science students from colleges near a Symposia, to attend Symposia sessions of interest to each SIG. These notes will be edited into articles and included into the "Mini-Tasker" as soon after a Symposia as possible. The sessions being covered at the St. Louis Symposia for RT-11 are:
    a. RT-11 Version 5 overview
    b. creation & handling of multi-volume directories
    c. MACDBG/RT-11 - a user's critique
    d. RT-11 Link internals
    e. RTEM-11 - the RT emulation
    f. RT-11 user application workshop
    g. FORTRAN IV/RT & its relation to FORTRAN standards
    h. FORTRAN/RT tutorial - EIS,FIS,FPU
    i. RT-11 MACRO/FORTRAN interactions
    j. RT-11 XM - gotchas and workarounds
    k. DRTS - a multi-processor operating system
    l. how to use logical disks with RT-11 Version 5
    m. RT-11 users speakout ! (workshop)
    n. RT-11 feedback session

   They should really generate many informative articles.
3. The "Mini-Tasker" will be getting more timely information directly from the RT-11 development group at DEC.

I hope that the approximately 7,500 of you that currently receive the "Mini-Tasker" will continue to do so in the future.

Sincerely,

Ken Demers

DECUS

PUBLICATIONS SUBSCRIPTION SERVICE

The past few years have brought an almost explosive growth in DIGITAL products and DECUS membership. Growing even faster is the information we need to exchange with each other. The DECUS mission is simple – to promote the exchange of information among users of Digital Equipment Corporation products and services.

The main means for exchanging this information are DECUS publications: newsletters, proceedings and catalogs. DECUS publications are the result of the efforts of many active volunteers. Too often, the newsletter editors and staffs have been limited by lack of resources such as equipment, travel and training. The publications service fee will help DECUS volunteers to publish regular, timely newsletters.

Each user service fee includes the latest DECUS Library catalog. The user may then choose from a list of 14 different newsletters and the DECUS Symposia Proceedings. Some of the Special Interest Group newsletters have been grouped to reduce your costs. Each group counts as one selection. If you choose, one or both of the Symposia Proceedings can be substituted for a newsletter(s). Please note that Symposium attendees receive a copy of the Proceedings as part of their registration and will not have to order one through subscription.

OFFERINGS

- MUMPS/Structured Languages Newsletter
- LABS/HMS/Site Management Newsletter
- Office Automation/DIBOL/COBOL/Graphics Newsletter (includes 12-Bit)
- VAX/VMS Newsletter (Pageswapper)
- RSX/IAS Newsletter (Multi-Tasker)
- RT-11 Newsletter (Mini-Tasker)
- RSTS Newsletter (Cache Buffer)
- Large Systems Newsletter (At-Large)
- EDUSIG Newsletter (EDUSIG)
- DATATRIEVE Newsletter (Wombat Examiner)
- NETWORKS Newsletter (NETwords)
- SS&OS Newsletter (Toolkit)
- BASIC Newsletter
- APL Newsletter (Special Character Set)
- Fall Proceedings
- Spring Proceedings

The Data Management SIG articles will be included in the appropriate operating system SIG publications rather than in a separate newsletter.

PRICES

MEMBERS and DIGITAL Employees: When placing an order the following prices will apply:

One (1) publication	$ 12.00/year	Up to six (6) publications	$ 45.00/year
Up to three (3) publications	$ 25.00/year	All publications	$120.00/year

Employees of Digital Equipment Corporation who subscribe to this service are automatically members of DECUS, so should check the price box marked "Members" on the order form.

For DIGITAL Employees, we will be able to cross charge your cost center. Please be sure to include your BADGE NUMBER and COST CENTER on the order form where indicated. Direct payment will not be necessary.

NON MEMBERS: When placing an order the following prices will apply:

One (1) publication	$ 24.00/year	Up to six (6) publications	$ 90.00/year
Up to three (3) publications	$ 50.00/year	All publications	$240.00/year

Special Interest Groups can only publish material they receive from users. To insure continued publication of all newsletters, please contribute articles to those publications in which you are interested.

DISCLAIMER

Neither DECUS nor Digital Equipment Corporation is responsible for the material contained in a newsletter. Absolutely no refunds will be made for any reason. No changes are to be made to a subscription in mid-year. You may, however, place a new order at any time. The number of issues published by a Special Interest Group in a subscription year is the responsibility of the SIG and is not guaranteed by DECUS or DIGITAL.

DECUS SUBSCRIPTION SERVICE ORDER FORM

RETURN TO: Subscription Service
DECUS
One Iron Way, MRO2-1/C11
Marlboro, MA 01752

- All checks payable to DECUS
- All orders MUST be paid in full
- No refunds will be made
- Prices indicated are FY'84 prices

Name ______________________ (First) ______________________ (Last) DECUS Membership No. __________

Company/Affiliation ____

Mailing Address ______________________________ Mail Stop __________

City __________ State/Country __________ Zip Code __________ Phone (   ) __________

PUBLICATIONS SUBSCRIPTION SELECTIONS

CODE	PUBLICATION	CODE	PUBLICATION
MSL	MUMPS/STRUCTURED LANGUAGES NEWSLETTER	RST	RSTS NEWSLETTER
LHS	LABS/HMS/SITE MGMT NEWSLETTER	LGS	LARGE SYSTEMS NEWSLETTER
OAD	OA/DIBOL/COBOL/GRAPH NEWSLETTER	EDU	EDUSIG NEWSLETTER
VAX	VAX/VMS NEWSLETTER	DTR	DATATRIEVE NEWSLETTER
RSX	RSX/IAS NEWSLETTER	NTW	NETWORKS NEWSLETTER
RT	RT11 NEWSLETTER	SOS	SS&OS NEWSLETTER
SPR	Spring Proceedings	BAS	BASIC NEWSLETTER
FAL	Fall Proceedings	APL	APL NEWSLETTER
		ALL	ALL PUBLICATIONS PRODUCED

	Insert Code From Above:	Check One:		
BASIC PLAN: This plan allows you to receive one (1) selection for one year	□	□	Member/DIGITAL Employee	$ 12.00
		□	Non Member	$ 24.00
STANDARD PLAN: This plan allows you to receive up to three (3) selections at one low price.	□ □ □	□	Member/DIGITAL Employee	$ 25.00
		□	Non Member	$ 50.00
DELUXE PLAN: This plan allows you to receive up to six (6) selections for one year.	□ □ □ □ □ □	□	Member/DIGITAL Employee	$ 45.00
		□	Non Member	$ 90.00
ALL: This will allow you to receive all publications listed above for one year for only one price.	ALL	□	Member/DIGITAL Employee	$120.00
		□	Non Member	$240.00

TOTAL AMOUNT OF ORDER $__________

I understand that neither DECUS nor Digital Equipment Corporation is responsible for any publication not published by a Special Intetest Group or the contents of any publication published by a Special Interest Group. I also understand that there will be no refunds even if I decide to cancel my subscription.

Signature ______________________________ Date __________

DIGITAL Employees Only: Badge No. ______________________ C.C. __________

Cost Center Manager's Signature ______________________ C.C. __________

USER INPUT

From: Tim Parker
Brigham & Women's Hospital
Neurophysiology Dept.
75 Francis St. EEG +2
Boston, MA
02115

RE: Wish lists, etc.

After reading the wish lists in the March issue of the Mini-Tasker, I am finally moved to write regarding my view of RT11's shortcomings. Hence a list of random notes..

1) Why is no one talking of UNIX-like characteristics for RT?? I am currently considering writing a SH handler and SHELL to emulate UNIX command interpretation (Ideas?? Suggestions??). Notes on the subtleties of BATCH internals would be very useful. This would get around the problem of command passing, command names, and other such things.. The current set of monitor commands is very useful, but the present implementation allows no flexibility. MUNG and MAKE possibly should not be standard SJ features, but the user should have the option to add them without extreme difficulty. Would it be unreasonable to ditch R/RUN F/Srun alltogether and interpret the first word in a command line as a program to run? Is this too revolutionary??

2) RE: Upper/Lower case conversion... Dive into RMON and move the case conversion code from interupt time to .TTYIN/.TTINR time. This has the added benefit of allowing lower case command files. (BEWARE.. CSI is not happy about lower-case switches in some places.. Best to keep your switches upper case).. This allows the terminal to echo whatever is typed at the keyboard and still provides the case conversion for situations where it is needed. Also the problem of editor special mode is gotten around. (How many times have you typed ahead after issuing the exit command to your editor and had KMON reply with illegal command messages?. I consider this a bug, and am anxious to see a patch of this sort in the RT11 product..)...

3) DECUS 'C' - Wonderful. This is the prime motivation for #1 (above), as this style of user interface seems the most sensible. Now all we need is to pass the command line onward so we can ditch the

```
Argv:
```

request at run-time. Doesn't

```
.fix myfile.txt
```

make more sense than

```
.RUN FIX
Argv: MYFILE.TXT
```

??

4) The command file interpreter should be smarter. Wouldn't it be nice to be able to squeeze the device on which your command file resides without risk to life and limb? O.K... I admit this can be avoided by where the command file is placed on the disk, but isn't this alittle crazy? I suggest either .LOOKUP at each call to the command file (Insane for floppy users!!) or an internal flag set by SQUEEZE to force command files to be .LOOKUP'd afterward to insure no oddities. .SAV files generally contain lousy command

syntax!!

5) There should be a terminal pass-all mode. I have recently patched my monitor to do this, checking bit 0 of the JSW. This, coupled with editor special mode, allow me to talk directly to the terminal without having to worry about ^C/^S/^Q/^O etc. Make for easier EMACS implementations. XON/XOFF is nice, but the monitor should provide the rope with which to hang yourself if you don't want it.

6) ^T - as on TOPS-10/20 would be nice - something that would spit out such goodies as PC/PS, Program running (for command files with TT: QUIET set..), High memory limit (?), Overlay status (?), I/O status, etc.. This function, as on TOPS-10/20 should be optional (I.E. the user should be able to disable it) to keep from blowing real-time processes away..

7) There should be a system EMT to disable (all) tty input. The familiar code:

```
ttint:  rti

ttset:  mov     @#tt.int,ttsav
        mov     #ttint,@#tt.int
        return

ttrst:  mov     ttsav,@#tt.int
        return
```

should look familiar to anyone who has gone all out for data-collection performance... but what happens if you forget the call to ttrst?? What happens on a KMON/RMON failure or trap to 4/10?? REBOOT TIME!! TT: disable should be reset by the monitor at exit time, regardless of the cause of exit.

8) CLRBFI should be a standard RT system call. For those of you unfamiliar with TOPS-10/20, this call clears the (tt:) input buffer. Useful for error handling and elimination of unwanted type-ahead (How many times have you stuttered on the return and over-run an important PAUSE??? Very frustrating....

9) I want WALLPAPER. (From VMS) - Very nice for debugging, correspondence re: system bugs, command file building, SYSGEN, and so on.. Am I dreaming too much? BATCH has it, why can't us interactive slobs have it too?

10) There should be an option to extend file names beyond 6 characters. Is 9 characters unreasonable?? Also, is it so unreasonable to have extension-less file names that typing

.TYPE MYFILE

comes back with not found messages for MYFILE.LST?? What earthly reason for .LST over any other logical choice (nothing at all, .TXT, .MAP, .RNO, .POD <CMU - "Prince Of Darkness" - SCRIBE output file for DIABLO>, .MAC, .FOR, .PAS, .C ... I could go on for hours..).. I know it's big-system-ish, but why not have it look for MYFILE.* and report ambiguity if it finds more than one? I.E. if I have MYFILE.HLP and MYFILE.TXT, should it complain about no matching file or should it type them both or should it complain that MYFILE is not a unique file spec? Has this even been considered??

11) Editors. Has anyone out there got a good display editor? TECO is great, and the Screen versions kicking around are hard to beat, but there are several limitations I find horrible. I.E. the paging scheme. I want to be able to back up without limitations. Also, the long startup time of most screen versions is akin to watching paint dry. If no one out there has one, I would be very interested in hints/suggestions/help re: implementing one using the DECUS C. <Why C?? I have no use for FORTRAN as a reasonable programming language [Yes - I am one of those structured programming bigots], and I don't have the time or the energy to do it all in MACRO.> I am especially interested in other peoples ideas regarding internal swapping algorithms and command sets/structures...

There you have it. My Want/Gripe/Grump/So-Fix-It-Yourself list...

P.S. How many people are running DECUS C under RT?? From the release it seems that most are running under RSTS/RT.. I have several patches for FWILD/FNEXT and am working on DTOA, FSCANF, and ATOF with a mind to killing off the last few RT native bugs..

Sincerely

Tim Parker

APPLICATION NOTE :   RT 11 (FORTRAN IV SOURCES)
****

****

DOST1 : A READY-TO-USE PROGRAM FOR STATISTICAL ANALYSIS OF THE CONTENTS AND THE STRUCTURE OF EXITING BIBLIOGRAPHIC FILES.

****

BY DANIEL GUINIER

LABORATOIRE DE PHYSIOLOGIE COMPAREE DES REGULATIONS
GROUPE DE LABORATOIRES DU CNRS DE STRASBOURG-CRONENBOURG
23 RUE DU LOESS
B.P.20 CR
67037 STRASBOURG CEDEX   FRANCE

****

INTRODUCTION :
****

BEFORE HANDLING EFFICIENTLY BIBLIOGRAPHIC RECORDS, IT IS NECESSARY TO DETERMINE THE QUANTITIES AND THE STRUCTURES OF THE INFORMATION WHICH CONSTITUTE THEM.

THE REALISED PROGRAM "DOST1" FURNISHES STATISTICS ON THE NUMBERS OF CHARACTERS FOR EACH DOCUMENTATION FIELD (AUTHORS, TITLES, PUBLICATIONS, KEY-WORDS,...) AND SUB-FIELD ON THE NUMBERS OF AUTHORS AND KEY-WORDS PER ITEM AND ON THE FREQUENCY DISTRIBUTIONS OF THE AUTHOR NAMES, THE PUBLICATION NAMES, THE KEY-WORDS AND YEARS OF PUBLICATION. IT IS ALSO INTERESTING TO BE ABLE TO ELIMINATE KEY-WORDS OF LOW FREQUENCY WHICH DO NOT REPRESENT A REAL INTEREST IN A SINGLE SCANNING WITH AN APPROPRIATE SECONDARY PROGRAM EASIELY REALISABLE BY THE USER.

OUR PURPOSE IS TO APPLY THE PROGRAM "DOST1" TO THE ANALYSIS OF A DOCUMENTATION FILE HOLDING ABOUT 10000 ITEMS WITH SOME 6000 KEY-WORDS, AND TO REALISE A NEW PROGRAM THAT PERMITS OPTIMISED CONVERSATIONAL AND INTERACTIVE MANAGEMENT OF SCIENTIFIC DOCUMENTATION WITH MAXIMUM GUARANTEE AND IN DIRECT ACCESSIBLE ORGANIZED. AT PRESENT, THE STUDIED FILE IS MANAGED BY THE PROGRAM "SABIR" AT THE "CENTRE DE CALCUL DU C.N.R.S. DE STRASBOURG-CRONENBOURG" ON UNIVAC 1110. THIS WAS THE UNDERTAKING OF THE DOCUMENTATION PERSONAL OF OUR LABORATORY (G.BIELLMANN,M.-A.NEISS AND M.-J.SCHWOERER) IN COLLABORATION WITH DIFFERENT RESEARCHERS.

BIBLIOGRAPHY :
****

D.GUINIER,R.KIRSCH (1977) : COMPLETE MANAGEMENT OF BIBLIOGRAPHIC FILES. (DECUS MINI-TASKER ,VOL.3, NO.3, SEPTEMBER RT11 SIG).

F.H.AYRES,E.J.YANNAKOUDAKIS (1979) : THE BIBLIOGRAPHIC RECORD : AN ANALYSIS OF THE SIZE OF ITS CONSTITUENT PARTS. (PROGRAM, JULY, VOL.13, NO.3, PP.127-142).

W.WILDGRUBE (1979) : A FORTRAN IV PROGRAM FOR QUANTITATIVE CONTENT ANALYSIS. (EDUC.AND PSYCHOL.MEASUREMENT, 39, PP.695-696).

STRUCTURE OF THE PROGRAM :

THE OVERLAY STRUCTURE OF THE PROGRAM IS :

```
                        ----------
                        I        I
                        I DOST1  I  (RESIDENT PART)
                        I        I
                        ----------
                            *
                            *
    **************************************************
    *            *          *            *           *
    *            *          *            *           *
----------  ----------  ----------  ----------  ----------
I        I  I        I  I        I  I        I  I        I
I DEVLEC I  I  FIL   I  I CUTART I  I RTYPE  I  I OUTFIL I
I        I  I        I  I        I  I        I  I        I
----------  ----------  ----------  ----------  ----------
                        I        I                  I
                        I CUTBUF I                  I---->FILE FOR STATISTICS
                        I        I
                        ----------
                            *
                            *
                      *************
                      *           *              FILES :
                      *           *
                 ----------   ----------
                 I        I   I        I      1) DATA FILE (A() AND OCC)
                 I  HDAN  I   I DFMDAN I----> 2) ADDRESS TABLE FILE
                 I        I   I        I      3) COLLISIONS TABLE FILE
                 ----------   ----------
```

DESCRIPTION OF THE MODULES :

DOST1 : MAIN RESIDENT CALLING MODULE.

DEVLEC : READ THE ITEMS FROM ANY DEVICE OR FILE, FORMATTED OR NOT,
WITH DIRECT-ACCESS OR SEQUENTIAL ORGANIZATION.

FIL   : CREATES A FORTRAN LOGICAL UNIT FROM ANY DEVICE
OR / AND FILE WITH DIRECT-ACCESS OR SEQUENTIAL ORGANIZATION

CUTART : RETRIEVES THE DIFFERENT FIELDS IN ANY ITEM.

CUTBUF : SUBDIVIDES THE FIELDS : AUTHOR NAMES, EDITION AND
KEY-WORDS INTO SUBFIELDS.

HDAN   : CALCULATES AN ADDRESS FROM CHARACTER INFORMATION
BY A "HASH-CODING" TECHNIQUE.

DFMDAN : DATA MANAGEMENT SUBROUTINE USING THREE FILES IN THIS CASE
AND THE RESULT OF THE PREVIOUS FUNCTION "HDAN".

"DFMDAN" BEGINS REALLY A DATA FILE MANAGEMENT SYSTEM WHEN ASSOCIATED TO A "HASH-CODING" FUNCTION WITH THE 3TH., 4TH. AND 5TH. FUNCTIONS COMPLETED BY THE USER (SEE END OF THE LISTING OF "DFMDAN", IN THIS CASE THE SUBROUTINE USES FOUR FILES).'

RTYPE : COMPUTES STATISTICS (MEAN, STANDARD-ERROR, ...)

OUTFIL : STORES STATISTICS IN THE CORRECT FILE, OUTPUTS THE RESULTS ON THE CONTROL TERMINAL.

DESCRIPTION OF THE FORTRAN UNITS : (DEVICES OR/AND FILES)

A) INPUT OF THE BIBLIOGRAPHIC FILE (IN DEVLEC) : ANY FILE, DISK, MAGTAPE (MT:). IN THE PRESENT CASE, THE RECORDS ARE UNFORMATTED WITH A LENGTH OF 512 CHARACTERS. THE EXACT STRUCTURE OF EACH RECORD IS DESCRIBED IN THE COMMENTS AT THE BEGINNING OF THE MAIN PROGRAM "DOST1".

B) MANAGEMENT OF THE DATA (IN DFMDAN) : (THE THREE FOLLOWING FILES REPRESENTED BY THE 1E.,2E. AND 3E. FORTRAN UNITS NAME MUST BE DIRECTLY ACCESSIBLE FILES).

1E. FILE ("DATA FILE") : MAXIMUM LENGTH : NENR(1), 18 EQUIVALENT WORDS PER RECORD REPRESENTED BY : 4 REAL*8 (A(1),A(2),A(3),A(4)) + THE REAL NUMBER OF OCCURRENCES : OCC, (OCC=NOCC+I(1)/100 000). I(1) IS THE LOADING INDEX OF THE FILE 1 AND NOCC IS THE NUMBER OF OCCURRENCES OF AN ELEMENT FOR A GIVEN DICTIONARY (AUTHORS OR/AND PUBLICATIONS OR/AND KEY-WORDS). OCC PERMITS TO SORT NOCC IN DECREASING ORDER WITH THE POSSIBILITY OF RESTITUTION OF THE CORRESPONDING I1 BY THE SINGLE FORMULA : I1=(OCC-INT(OCC))*100000. I1 BECOMES THE POINTER OF THE CORRECT A()'S WHICH THEMSELVES CAN BE TRANSFORMED INTO CHARACTERS STRING BY THE "R50ASC()" FUNCTION OF "SYSLIB" FOR A FUTURE EDITION.

2E. FILE ("ADDRESS TABLE FILE") : MAXIMUM LENGTH : NENR(2), 2 EQUIVALENT WORDS REPRESENTED BY TWO INTEGERS : I1 AND I3 (I1 IS THE DATA POINTER AND I3 IS THE COLLISION POINTER). I(2) IS THE LOADING POINTER OF THE FILE 2. THE NUMBER OF POSSIBLE ADDRESSES IS NADR=NLOG()=NENR(2)-102, BECAUSE THE NADR+1 E. RECORD IS ASSIGNED TO THE LOADING POINTERS I(1) AND I(2), THE NADR+2 E. RECORD IS ASSIGNED TO THE LOADING POINTERS I(3) AND I(4) AND THE 100 LAST ADDRESSES (FROM NADR+3 TO NADR+102=NENR(2)) ARE RESERVED FOR THE REAL NUMBER OF OCCURRENCES FOR THE PUBLICATIONS YEARS (DATES) FROM <=1901 TO 2000.

3E. FILE ("COLLISIONS TABLE FILE") : MAXIMUM LENGTH : NENR(3), 3 EQUIVALENT WORDS PER RECORD REPRESENTED BY : I1, THE DATA POINTER, NF, THE NEXT I3 ("SON") AND NP, THE PRECEDENT I3 ("FATHER"). I(3) IS THE LOADING POINTER OF THE FILE 3.

```
              FILE 1 :
              --------

         ----------------------------   I(1)=1
         I  A(1) I  A(2) I  A(3) I  A(4) I OCCI
         I       I       I       I       I   I
         I       I       I       I       I   I
         I       I       I       I       I   I
         .   .   .   .   .   .   .   .   .  . .
         .   .   .   .   .   .   .   .   .  . .
         .   .   .   .   .   .   .   .   .  . .
         I       I       I       I       I   I
         I       I       I       I       I   I
         I       I       I       I       I   I
         ----------------------------   I(1)=NENR(1)
                    FILE 2 :
                    --------

               -----------      I(2)=1
               I I1 I I2 I
               I    I    I
               I    I    I
               .  .  .  .  .
               .  .  .  .  .
               .  .  .  .  .
               I    I    I
               I    I    I
               I    I    I
               I    I    I      I(2)=NADR=NENR(2)-102
               -----------
               II(1)II(2)I      I(2)=NADR+1
               -----------
               II(3)II(4)I      I(2)=NADR+2
               -----------
               I OCC DATEI      I(2)=NADR+3
               I         I
               I         I
               I         I
               .    .    .
                    .    .
                    .    .
               I         I
               I         I
               I         I      I(2)=NENR(2)
               -----------


                    FILE 3 :
                    --------

               ----------------
               I I1 I NF I NP I     I(3)=1
               I    I    I    I
               I    I    I    I
               I    I    I    I
               .    .    .    .
               .    .    .    .
               .    .    .    .
               I    I    I    I
               I    I    I    I
               I    I    I    I     I(3)=NENR(3)
               ----------------
```

C) OUTPUT (IN OUTFIL) :

-CONTROL TERMINAL FOR STATISTICS : STATISTICS ARE LISTED FOR ALL COMPLETE SERIES OF M ITEMS FROM 1 TO THE LAST ITEM REALLY ENCOUNTERED < OR = TO MAXB. THE LISTING IS GENERALLY OUPUTTED ON LP: OR TT: AND REPRESENTS THE MEANS OF X(1) TO X(16) AND THE STANDARD-ERROR OF THE MEANS FOR X(1) TO X(13). FOR THE MEANING OF THE X()'S, SEE THE END OF THE LISTING OF THE SUBROUTINE "CUTART".

-FILE FOR STATISTICS : THE FILE 4 STORES THE RESULTS OF THE PRECEDING STATISTICS FOR X(1) TO X(17).

REMARKS :

THE COMPILATION OPTION /N!7 MUST BE USED WITH RT11 FORTRAN.

THE MODULARITY OF THE PROGRAM "DOST1" PERMITS THE USE OF THE LINK OPTION /O (OVERLAY) : ON TWO DIFFERENT LEVELS (/O:1 AND /O:2) TO RUN THE PROGRAM WITH 16 KWORDS OF MEMORY. WITHOUT THIS OPTION, THE PROGRAM RUNS CORRECTLY WITH OUR CONFIGURATION (24 KWORDS).

SET USR NO SWAP BEFORE RUNNING THE PROGRAM.

INSTRUCTIONS TO USE :
****

AFTER SETTING USER SERVICE ROUTINES NO SWAP, RUN DOST1 AND ANSWER THE QUESTIONS :

```
---------------------------------------------------------------------------

MAX.NBR.OF ITEMS :                                          MAXB
STATISTICS ALL THE M ITEMS;   M=                            M
 1E.FORTRAN UNIT NAME (DEVICE AND/OR FILE) :                NAME OF FILE 1
MAX. RECORDS NUMBER :                                       NENR(1)
EQUIVALENT WORDS NUMBER PER RECORD :                        18
 2E.FORTRAN UNIT NAME (DEVICE AND/OR FILE) :                NAME OF FILE 2
MAX. RECORDS NUMBER :                                       NENR(2)
EQUIVALENT WORDS NUMBER PER RECORD :                        2
 3E.FORTRAN UNIT NAME (DEVICE AND/OR FILE) :                NAME OF FILE 3
MAX. RECORDS NUMBER :                                       NENR(3)
EQUIVALENT WORDS NUMBER PER RECORD :                        3
FILE 1 WITH AUTHORS NAMES (1) OR NOT (0) :                  0 (OR 1)
FILE 1 WITH PUBLICATIONS NAME (1) OR NOT (0) :              0 (OR 1)
FILE 1 WITH KEY-WORDS (1) OR NOT (0) :                      0 (OR 1)
THESE FILES ARE EXISTING AT TIME (1) OR NOT (2) :           2 (OR 1)
NAME OF THE RANDOM-ACCESS FILE FOR STAT. :                  NAME OF FILE 4
FILE NAME OR / AND DEVICE NAME :                            MT: (OR FILE NAME)
CONTROL TERMINAL FOR STATISTICS (LP:, TT:, FILE) :          LP:

---------------------------------------------------------------------------
```

MAXB, M, NENR(1), NENR(2) AND NENR(3) MUST BE SELECTED AS A FUNCTION OF THE EXPECTED MAXIMA.

LISTING OF THE DIFFERENT MODULES : (MAIN, SUBROUTINES AND FUNCTION)
****

```
C
C  ****************************************************************
C
C  DOST1 : AUTHOR : DANIEL GUINIER
C
C  DOST1 : PROGRAM FOR CONTAIN STATISTICAL ANALYSIS OF THE BASIS
C          BIBLIOGRAPHIC ASCII FILE WHICH STRUCTURE IS :
C
C  EXAMPLE :
C  ----------------------------------------------------------------
C
C  $1$000019$2$01/3/1/76$3$MUNSHI J.S.D.,DUBE S.C.$4$***$5$OXYGEN U
C  PTAKE CAPACITY OF GILLS IN RELATION TO BODY SIZE OF THE AIR-BREA
C  THING FISH, ANABAS TESTUDINEUS (BLOCH).$6$***$7$ACTA PHYSIOL.ACA
C  D.SCI.HUNG.,1973,44,113-123$8$0,M02,GILL,AIR,AIR-BREATHING,BODY
C  WEIGHT,SURFACE GILL,TELEOSTEI,ANABAS TESTUDINEUS,$9$***&
C
C  ----------------------------------------------------------------
C
C  FOR AN ITEM CORRESPONDING TO A 256 WORDS BLOCK (512 CHARACTERS).
C
C  DESCRIPTION OF THE DIFFERENT FIELDS (WITH '$N$' SEPARATORS) :
C
C  $1$ : INTERNAL RESERVATION.
C  $2$ : DATE (DAY,MONTH,YEAR).
C  $3$ : AUTHORS NAME AND BIRTHNAMES.
C  $4$ : -----FREE-----
C  $5$ : TITLE.
C  $6$ : -----FREE-----
C  $7$ : NAME OF THE REVIEW, PUBLICATION YEAR, VOL. OR NBR., PAGES.
C  $8$ : KEYWORDS (WITH SEPARATORS (',')).
C  $9$ : -----FREE-----
C
C  &   : FOR END OF ITEM.
C
C  DOST1 MUST BE COMPILED WITH THE /N!7 OPTION TO RUN.
C                                  ...................
C  DOST1 MUST BE USED WITH THE "SET USR NO SWAP" SPECIFICATION.
C
C                                  .........  ........
C
C  DOST1 USES 5 FILES :
C
C  1 : N1 RECORDS OF 18 WORDS EQUIVALENT  (4*REAL*4 RAD50 TRANS.+1 REAL*4)

C  2 : N2=N2+2+100 RECORDS OF 2 WORDS :
C      N2 : NBR.OF POSSIBLE ADDRESSES FOR THE "HASH-CODING" FUNCTION.
C             (2 INTEGER*2)
C         2 : 2 ADDRESSES FOR THE I()  (2*2 INTEGER*2 ).
C       100 : RECORD-FIELD OF THE OCCURENCES OF YEARS  (1 REAL*4).
C
C  3 : N3 RECORDS OF 3 WORDS FOR TREATMENTS OF COLLISIONS.
C  4 : M RECORDS OF 17 REAL*4 FOR STATISTICS.
C      M1 FOR THE X()
C      M2 FOR THE MEANS OF THE X() EACH M ITEMS.
C  5 : THE ASCII FILE USE BY DEVLEC(...) FOR EACH ITEM CONTAIN  (256 WORDS
C   FOR THE FILES DECLARATION : N2 + N3 > N1
C   IF N1 IS THE MAX.VALUE OF THE DICTIONARY WANTED (AUTHORS NAME,
C   REVIEWS NAME, KEYWORDS).
C
C
C
C   ****************************************************************
C
C
          BYTE ART(512)
          INTEGER*2 NLOG(9),INDX(3),NENR(3)
          REAL*4 X(17)
C
          DATA LEC,IMP/5,7/NENR/3*0/NLOG/9*0/J2/0/OCC/0./
C
          WRITE(IMP,100)
    100   FORMAT('$MAX.NBR.OF ITEMS : ')
          READ(LEC,200)MAXB
    200   FORMAT(I5)
          WRITE(IMP,300)
    300   FORMAT('$STATISTICS ALL THE M ITEMS;   M = ')
          READ(LEC,200)M1
          M2=MAXB/M1
          M =M1+M2
C
C   FILES DECLARATION (FILES : 1 TO 3).
          DO 1 I=1,3
    1     CALL FIL(LEC,IMP,I,1,NENR(I),NBRMOT,U,INDX(I))
          WRITE(IMP,400)
    400   FORMAT('$FILE 1 WITH AUTHORS NAME (1) OR NOT (0) : ')
          READ(LEC,200)J
          IF(J.NE.0)NLOG(3)=NENR(2)-102
          WRITE(IMP,500)
    500   FORMAT('$FILE 1 WITH PUBLICATIONS NAME (1) OR NOT (0) . ')
          READ(LEC,200)J
          IF(J.NE.0)NLOG(7)=NENR(2)-102
          WRITE(IMP,600)
    600   FORMAT('$FILE 1 WITH KEY-WORDS (1) OR NOT (0) : ')
          READ(LEC,200)J
          IF(J.NE.0)NLOG(8)=NENR(2)-102
C   IND1=1 : "LOOKUP" ; IND1=2 : "ENTER".
          WRITE(IMP,700)
    700   FORMAT('$THESE FILES ARE EXISTING AT TIME (1) OR NOT (2) : ')
          READ(LEC,200)IND1
C
C   FILE FOR STATISTICS ON X() : (FILE : 4).
          WRITE(IMP,800)
    800   FORMAT('$NAME OF THE RANDOM-ACCESS FILE FOR STAT. : ')
          CALL ASSIGN(4,'TT:',-1)
          DEFINE FILE 4(M,34,U,INDX4)
C
C   FILE 2 IS USED FOR NBR.OF OCCURENCES FOR A YEAR FROM (1900 TO 2000
C   FROM THE ADDRESS NENR(2)-99 TO NENR(2).
```

```
      C  ZERO THIS PART OF THE FILE 2 WHEN NECESSARY.
0038          IF(IND1.EQ.1)GO TO 3
0040          I1=NENR(2)-99
0041          I3=NENR(2)
0042          DO 2 IND=I1,I3
0043  2       WRITE(2'IND)OCC
      C
      C  WHEN DEVLEC IS USED DON'T FORGET TO "SET USR NO SWAP"
      C  BEFORE RUNNING THE MAIN PROGRAM.
      C  CALL DEVLEC(...) CAN BE CHANGE BY CALL ASSIGN(...)
      C  AND DEFINE FILE X(...)
      C  IF NO SEQUENTIAL UNFORMATED DEVICE IS USED.
      C
0044  3       LOOK=0
0045          I=0
0046          DO 4 IN=1,MAXB
0047          I=I+1
0048          CALL DEVLEC(LEC,IMP,ART,   IN, IN,LOOK,NOCAN,IERR)
0049          IF(IERR.EQ.-1)STOP 'END OF FILE AND OPERATIONS !'
0051          CALL CUTART(ART,NLOG,X,IND1,IERR)
      C
0052          IF(IERR.EQ.0)GO TO 5
0054          WRITE(IMP,888)IN,ART
0055  888     FORMAT(/' ERROR IN ITEM NBR.',I5//8(4X,64A1/))
0056          I=I-1
0057          GO TO 4
      C
0058  5       IND=X(9)-1900
0059          IF(IND.LE.  0)IND=  1
0061          IF(IND.GT.100)IND=100
0063          IND=IND+NENR(2)-100
0064          READ(2'IND)OCC
0065          OCC=OCC+1
0066          WRITE(2'IND)OCC
      C
0067          IND=MOD(I,M1)
0068          READ(2'NENR(2)-101)I1,K
0069          READ(2'NENR(2)-100)I3,K
0070          X(15)=I1
0071          X(16)=I3
0072          X(17)=I
0073          CALL OUTFIL(IMP,4,M1,J2,X,IND)
0074  4       CONTINUE
0075          IN=IN-1
0076          WRITE(IMP,900)I,IN
0077  900     FORMAT(I5,' CORRECT ITEMS ON',I5/)
0078          STOP
0079          END
```

```
      C
      C  ****************************************************************
      C
0001        SUBROUTINE DEVLEC(LEC,IMP,ART,MAXB,NB1,IND,NOCAN,IERR)
      C
      C  LEC : LOGICAL FORTRAN UNIT FOR  INPUT (TT:)
      C  IMP : LOGICAL FORTRAN UNIT FOR OUTPUT (TT:)
      C  ART : READ BUFFER (512 CAR.).
      C  MAXB: MAXIMUM BLOCK NUMBER.
      C  NB1 : FIRST BLOCK NUMBER.
      C  IND : INDEX FOR LOOKUP IF IND=0; IN THIS CASE, DO :
      C        "SET USR NO  SWAP" BEFORE RUNNING THE CALLING PROGRAM.
      C NOCAN: SYSTEM CHANNEL NUMBER.
      C  IERR: IERR<0 IF END-OF-FILE OR ERROR IN READ OPERATION.
      C
      C
      C
      C  READ ANY FILE ON ANY DEVICE, FORMATED OR NOT PER BLOCK
      C  OF 256 16 BITS WORDS FROM THE NB1 TH. TO THE MAXB TH. BLOCK.
      C  IF DIRECT ACCESS MEDIA : NB1 MAY BE > 1 AND
      C  NB1 MAY BE EQUAL TO MAXB TO READ THE NB1TH. BLOCK.
      C
0002        BYTE ART(512)
0003        INTEGER*2 SPECIF(39)
0004        REAL*4 EXT(2)
      C
0005        DATA EXT/2*6RDATDAT/
      C
      C  LOOKUP : FILE OR / AND INPUT DEVICE
      C  CONTAINING THE ITEMS.
      C
0006        IF(IND.NE.0)GO TO 2
      C  TO READ ITEMS
0008        WRITE(IMP,100)
0009  100   FORMAT('$FILE NAME OR / AND DEVICE NAME : ')
      C
      C  STRING WITHOUT ANY SWITCHES.
0010  1     IF(ICSI(SPECIF,EXT,,,0).NE.0)GO TO 1
      C
      C  ASSIGN FORTRAN LOGICAL NUMBER 1 FOR THE INPUT READ-ONLY FILE.
      C  32: READ-ONLY FILE.
      C   4: THE FIRST I/O OPERATION DETERMINES THE NATURE OF THE FILE.
      C  THE DEFAULT EXTENSION IS "DAT".
0012        NOCAN=IGETC()
      C
0013        IF(NOCAN.LT.0)STOP ' NO CHANNEL !'
0015        IF(IFETCH(SPECIF(16)).LT.0)STOP ' BAD FETCH !'
0017        IF(LOOKUP(NOCAN,SPECIF(16)).LT.0)STOP ' BAD LOOKUP !'
      C  FREE THE USR.
0019        CALL UNLOCK
0020        IND=1
      C  CONTROL DEVICE FOR STATISTICS
      C
0021        WRITE(IMP,110)
```

```
0022  110    FORMAT('$CONTROL TERMINAL FOR STATISTICS (LP:,TT:,FILE) : ')
0023         CALL CLOSE (IMP)
0024         CALL ASSIGN(IMP,'TT:',-1)
      C
      C  READ PER BLOCK (256 16 BITS WORDS).
      C
      C
0025  2      DO 4 NOBLOC=NB1,MAXB
      C  DECREMENTATION OF 1 FOR THE SYSTEM INDEX TO READ THE GOOD BLOCK
0026         NB=NOBLOC-1
0027         IERR=IREADW(256,ART,NB,NOCAN)
      C  FOR A MISSING '&', LAST CHARACTER IS '&'
0028         ART(512)=38
0029         IF(IERR.GE. 0)GO TO 3
0031         IF(IERR.LT.-1)WRITE(IMP,200)
0033  200    FORMAT(' ERROR IN READ OPERATION !')
0034         IF(IERR.EQ.-1)WRITE(IMP,300)
0036  300    FORMAT(' END OF FILE !')
0037         RETURN
      C
0038  3      CONTINUE
      C  IF CONTROL OPERATIONS.
      C  3  WRITE(IMP,400)NOBLOC,ART
      C  400    FORMAT(10X,'BLOCK NBR.',I5//8(2X,64A1/))
0039  4      CONTINUE
0040         RETURN
0041         END
```

```
FORTRAN IV      V01C-03A   MON 20-DEC-82 10:47:05

       C
       C  ************************************************
       C
0001          SUBROUTINE RTYPE(N,RMOY,STDEV,COEVAR,ERTYPE,X)
       C
       C  RTYPE : COMPUTES STATISTICS ON THE MEAN OF THE X().
       C
0002          DIMENSION X(N)
       C
0003          SOM=0.
0004          SCAR=0.
0005          ERTYPE=0.
0006          DO 1 I=1,N
0007          SOM=SOM+X(I)
0008          SCAR=SCAR+X(I)*X(I)
0009   1      CONTINUE
0010          RMOY=SOM/N
0011          IF(N.EQ.1)RETURN
0013          Y=ABS((SCAR-(SOM*SOM)/N)/(N-1))
0014          STDEV=SQRT(Y)
0015          RN=N
0016          IF(RMOY.NE.0.000)COEVAR=STDEV*100/RMOY
0018          ERTYPE=STDEV/SQRT(RN)
0019          RETURN
0020          END
```

```
C   ************************************************************
C
          SUBROUTINE FIL(LEC,IMP,NLOGIC,IACCES,NENR,NBRMOT,U,INDX)
C
C   CREATE A FILE WHICH NAME AND / OR DEVICES
C   CORRESPONDING TO THE M LOGICAL FORTRAN UNIT NLOGIC.
C   IACCES = 0 : SEQUENTIAL ACCES ; IACCES = 1 : RANDOM ACCES.
C
          WRITE(IMP,100)NLOGIC
  100     FORMAT('$',I2,'E.FORTRAN UNIT NAME (DEVICE AND/OR FILE) : ')
          CALL ASSIGN(NLOGIC,'TT:',-1)
          IF(IACCES.EQ.0)RETURN
C
C   FOR RANDOM-ACCES.
C
          IF(NENR.GT.0)GO TO 1
          WRITE(IMP,200)
  200     FORMAT('$MAX. RECORDS NUMBER : ')
          READ(LEC,300)NENR
  300     FORMAT(I5)
          WRITE(IMP,400)
  400     FORMAT('$EQUIVALENT WORDS NUMBER PER RECORD : ')
          READ(LEC,300)NBRMOT
C
  1       DEFINE FILE NLOGIC (NENR,NBRMOT,U,INDX)
          RETURN
          END
C   ************************************************************
C
          SUBROUTINE CUTART(ART,NLOG,X,IND1,IERR)
C
C   EXAMINES THE DIFFERENTS FIELDS BETWEEN '$N$' SEPARATORS
C   IN THE BASIS BIBLIOGRAPHIC ASCII FILE.
C
C   ************************************************************
C
C   $1$ : INTERNAL RESERVATION.
C   $2$ : DATE (DAY,MONTH,YEAR).
C   $3$ : AUTHORS NAME AND BIRTHNAMES.
C   $4$ : -----FREE-----
C   $5$ : TITLE.
C   $6$ : -----FREE-----
C   $7$ : NAME OF THE REVIEW, PUBLICATION YEAR, VOL. OR NBR., PAGES.
C   $8$ : KEYWORDS (WITH SEPARATORS (',')).
C   $9$ : -----FREE-----
C
C   &   : FOR END OF ITEM.
C
C   DECIMAL VALUE OF SUB-SEPARATORS : '$'=36; '&'=38; ','=44
C                                     '-'=45; '.'=46; '/'=47
C                                     '0' TO '9'=48 TO 57
C
C   NLOG() : IS THE MAX.NBR.OF POSSIBLE ADDRESSES FOR FILE 1.
C   IF NLOG(J)>0 : VALUES OF THE JTH.SUB-FIELD MOVED IN FILES 1 TO 3.
C   ------------------------------------------------------------
C
          BYTE ART(512),BUF(512),CAR(128),CARN
          INTEGER*2 NLOG(9)
          REAL*4 X(17)
```

```
C
        I1=0
        IERR=0
        X( 9)=0.
        X(10)=0.
C
C  THE N FROM '$N$' MAY BE IN ANY RANK.
 101    I1=I1+3
        DO 102 M=1,512
        BUF(M)=ART(I1+M)
 102    IF(BUF(M).EQ.36.OR.BUF(M).EQ.38)GO TO 103
C  M CHARACTERS FOR THIS FIELD.
 103    M=M-1
        CARN=ART(I1-1)
        I1=I1+M
        DECODE(1,1000,CARN,ERR=999)J
 1000   FORMAT(I1)
C
        IF(BUF(M+1).EQ.38)GO TO 10

        GO TO (1,2,3,4,5,6,7,8,9),J
C
C  $$$$$$$$$$111111111$$$$$$$$$$
 1      GO TO 101
C  $$$$$$$$$$222222222$$$$$$$$$$
C  THIS FIELD MAY BE : XX/X/X/XX OR XXXXXX/XX OR XXXXXX0XX
 2      CAR(1)=BUF(M-1)
        CAR(2)=BUF(M   )
        DO 21 J=2,4
 21     CAR(J*2-1)=48
        DO 22 J=2,4
        JJ=2*J
        MM=M-2*J
        IF(BUF(MM).NE.47)CAR(JJ-1)=BUF(MM)
 22     CAR(JJ)=BUF(MM+1)
        DECODE(M,2000,CAR)X(2)
 2000   FORMAT(F8.6)
        X(2)=X(2)+1900
        GO TO 101
C  $$$$$$$$$$333333333$$$$$$$$$$
 3      CALL CUTBUF(BUF,CAR,M,ICAR,NBR,100,NLOG(3),IND1)
        X(3)=NBR
        X(4)=ICAR
        X(5)=X(4)/NBR
        X(4)=M
        GO TO 101
C  $$$$$$$$$$444444444$$$$$$$$$$
 4      GO TO 101
C  $$$$$$$$$$555555555$$$$$$$$$$
 5      X(6)=M
        GO TO 101
C  $$$$$$$$$$666666666$$$$$$$$$$
 6      GO TO 101
C  $$$$$$$$$$777777777$$$$$$$$$$
 7      CALL CUTBUF(BUF,CAR,M,ICAR,NBR,   1,        0,IND1)
C  IF NO LAST SUB-FIELD (X(9) TO X(10) EXCLUDED)
        X(7)=M
        X(8)=M
        IF(M.EQ.ICAR)GO TO 101
```

```
C  CAR() CONTAINS THE LAST SUB-FIELD (YEAR OR EDITOR CONTINUATION).
C  TREATMENT FOR EVENTUAL EDITOR NAME CONTINUATION.
         DO 71 J=2,5
71       IF(BUF(J+ICAR).LT.48.OR.BUF(J+ICAR).GT.57)GO TO 72
C  NO CONTINUATION.
         CALL CUTBUF(BUF,CAR,M,ICAR,NBR,  1,NLOG(7),IND1)
         X(7)=ICAR
         DECODE(4,7000,BUF(ICAR+2))X(9)
7000     FORMAT(F4.0)
         GO TO 73
C  CONTINUATION.
72       BUF(ICAR+1)=32
         GO TO 7
73       CALL CUTBUF(BUF,CAR,M,ICAR,NBR,100,      0,IND1)
         X(8)=M
C  TREATMENT OF THE LAST SUB-FIELD (PAGES : XX-YY OR XXA-YYB).
         DO 74 J=1,10
74       CAR(J)=32
         JJ=10
         DO 77 ICAR=1,2
         DO 75 J=M,1,-1
         IF(BUF(J).EQ.44.OR.BUF(J).EQ.45)GO TO 76
         IF(BUF(J).LT.48.OR.BUF(J).GT.57)GO TO 75
         CAR(JJ)=BUF(J)
         JJ=JJ-1
75       CONTINUE
76       JJ=5
         M=J-1
77       CONTINUE
         DECODE(5,7100,CAR(6))X(10)
         DECODE(5,7100,CAR(1))XP
7100     FORMAT(F5.0)
         X(10)=ABS(X(10)-XP+1)
C  PROTECTION IN THE SUB-FIELD WHEN YEAR BUT NO PAGES PRESENT.
         IF(X(10).GT.1900.)X(10)=0.
         GO TO 101
C  $$$$$$$$$$888888888$$$$$$$$$$
8        CALL CUTBUF(BUF,CAR,M,ICAR,NBR,100,NLOG(8),IND1)
         X(11)=NBR
         X(12)=ICAR
         X(13)=X(12)/NBR
         X(12)=M
         GO TO 101
C  $$$$$$$$$$999999999$$$$$$$$$$
9        GO TO 101
C
C  RESULTS.
10       X(1)=M+I1-2
         X(14)=0.
C
C  X( 1)=TOTAL NBR.OF CHAR.  PER ITEM ($1$......&).
C  X( 2)=DATE (YEAR,MONTH,DAY : YYYY.MMDD).
C  X( 3)=NUMBER OF AUTHORS.
C  X( 4)=TOTAL NBR.OF CHAR.  FOR FIELD AUTHORS (INCLUDED ',' SEPARATORS).
C  X( 5)=MEAN OF NBR.OF CHAR.  PER AUTHOR NAME.
C  X( 6)=TOTAL NBR.OF CHAR.  FOR TITLE.
C  X( 7)=NBR.OF CHAR.FOR EDITOR NAME.
C  X( 8)=TOTAL NBR.OF CHAR.  FOR FIELD EDITION (NAME,DATE,VOL,PAGES,...)
C  X( 9)=YEAR OF EDITION.
```

```
      C   X(10)=NBR. OF PAGES.
      C   X(11)=NBR. OF KEYWORDS.
      C   X(12)=TOTAL NBR. OF CHAR.  FOR FIELD KEYWORDS (INCLUDED ',' SEPARATORS).
      C   X(13)=MEAN OF NBR. OF CHAR.  PER KEYWORD.
      C   IN CALLING PROGRAM : X(14)  : NO.  OF THE HUNDRED (50.,150.,250.,...).
      C                        X(15)  : VALUE OF I(1) FOR THE ACTUAL I TH. ITEM.
      C                        X(16)  : VALUE OF I(3) FOR I.  (NBR. OF COLLISIONS).
      C                        X(17)  : VALUE OF I.
      C
```

```
FORTRAN IV      V01C-03A   MON 20-DEC-82 10:48:58                PAGE 004

0095         RETURN
0096   999   IERR=1
0097         RETURN
0098         END

         C   ************************************************************
         C
   0001          SUBROUTINE CUTBUF(BUF,CAR,M,ICAR,NBR,MAXI,NADR,IND1)
         C
         C   SUBDIVIDE THE CONTAIN OF '$3$", '$7$', '$8$' : CORRESPONDING TO
         C   AUTHORS NAMES, EDITION AND KEYWORDS.
         C
         C   BUF() : BUFFER CORRESPONDING TO THE N TH.  FIELD '$N$'.
         C   CAR() : BUFFER FOR SUB-FIELDS IN EACH FIELD '$N$'.
         C   M     : NBR. OF CHARACTERS FOR THE N TH.  FIELD.
         C   ICAR  : SUM OF CHARACTERS FOR EACH CAR() SUB-FIELD.
         C   MAXI  : MAX. NUMBER OF CHARACTERS POSSIBLE (EX: 128).
         C   NADR  : IF NADR>0 , MANAGEMENT OF THE CORRESPONDING FILES IN
         C           RANDOM-ACCESS (FORTRAN LOGICAL NBR.  1 TO 4).
         C           NADR IS THE MAX. NBR. OF POSSIBLE ADDRESSES.
         C   I       NBR. OF CHAR. FOR THE TH. SUB-FIELD (<129), ONLY THE
         C           44 FIRST CHAR. ARE CONVERTED IN REAL*8 A(4) RADIX50
         C           LIKE : 4*(1 SPACE + 11 CHAR. ).
         C
   0002          BYTE BUF(512),CAR(128)
   0003          REAL*8 A(4)
         C
   0004          ICAR=0
   0005          BUF(M+1)=44
   0006          J =1
   0007          DO 34 NBR=1,MAXI
   0008          I =0
   0009          II=J
   0010          DO 31 J=II,M
   0011          IF(I. EQ. 128)GO TO 32
   0013          I=I+1
```

```
          CAR(I)=BUF(J)
    31    IF(BUF(J+1).EQ.44)GO TO 32
    32    J=J+2
          ICAR=ICAR+I
C  CONVERSION ADDRESS = H (KEY)   ; ("HASH-CODING").
C  RECORD NUMBER = HDAN (CAR)
          IF(NADR.EQ.0)GO TO 33
          I2=HDAN(CAR,A,I,NADR)
          CALL DFMDAN(NADR,I2,A,IND1)
C  THE SUCCESSIVE CAR() CAN BE RE-WRITTEN IN BUF(), ASSOCIATED TO
C  THE I'S IN N() FROM 1 TO NCAR. IF NECCESSARY FOR MODIFICATIONS.
    33    IF(J.GE.M)RETURN
    34    CONTINUE
          END
C  *************************************************************
C
          FUNCTION HDAN(CAR,A,I,NADR)
C
C  RETURN THE ADDRESS FOR THE KEY CAR() CONTAINING ASCII CHAR.
C
C  (THIS "HASH-CODING" FUNCTION IS NOT WEIGHTED WITH
C  THE PROBABILITY DISTRIBUTION OF THE FIRST CHARACTER OF CAR).
C
C  *************************************************************
C
C  CAR() IS TREATED LIKE THE 44 FIRST CHAR. OF CAR() ARE USED LIKE
C  THE FOLLOWING FORMAT :  4*(<SPACE>,11 CHAR.).
C  EXAMPLE :
C   FIRSTFIRSTF SECONDSECON THIRDTHIRDT FOURTHFOURT
C  -           -           -           -
C  IS THE RESULT OF THE TREATMENT OF THE ORIGINAL CAR() :
C  FIRSTFIRSTFSECONDSECONTHIRDTHIRDTFOURTHFOURTXXXXXX...XXX
C
C  AND THE 4*(1+11)=48 CHAR.ARE REAL*8 RADIX 50 CONVERTED AND MOVED IN A()

C
C  AT THE END THE ADDRESS IN CALCULATED BY MEAN OF A(1).
C
C
C  I      : ACTUAL NBR.OF CHAR.DETERMINED.
C  NADR   : NBR.OF POSSIBLE ADDRESSES.
C
          BYTE CAR(128)
          REAL*8 A(4)
C
C  <SPACE> COMPLETION.
          I1=I+1
          DO 1 J=I1,48
     1    CAR(J)=32
          K1=I/12+1
          IF(K1.GT.4)K1=4
          DO 4 K=4,1,-1
          I1=11*K
          I2=I1-10
          IF(K.GT.K1)GO TO 3
          DO 2 J=I1,I2,-1
     2    CAR (J+K)=CAR(J)
     3    M=I2+K-1
          CAR(M)=32
```

```
      C   RAD50 TRANSFORMATION
0019         J=IRAD50(12,CAR(M),A(K))
0020  4      CONTINUE
      C   ADDRESS CALCULATION FROM A(1).
      C   CAR(2) (AT THE BEGINING : CAR(1)) MUST BE INCLUDED IN [A-Z].
      C   IF NOT; HDAN=NADR.
0021         HDAN=NADR
0022         IF(CAR(2).GT.90.OR.CAR(2).LT.65)RETURN
0024         HDAN=(DLOG(DABS(A(1)))+89.14405060)*(NADR-1)/5.56083+1.5
```

```
FORTRAN IV      V01C-03A   MON 20-DEC-82 10:54:45

      C   I IS RETURN WITH THE VALUE I-K1
0025         RETURN
0026         END
      C   ****************************************************************
      C
0001         SUBROUTINE DFMDAN(NADR,I2,A,IND)
      C
      C   DATA FILE MANAGEMENT SYSTEM USING THREE FILES TO INSERT OR GET AND ONE
      C   FILE MORE TO DELETE OR MODIFY.
      C   THE ACCESS IS REALIZED BY A "HASH-CODING" FUNCTION AND THE FOUR FILES
      C   ARE RANDOM-ACCESSIBLE.
      C
      C   1E.FILE : "DATA FILE". (18 WORDS).
      C   2E.FILE : "ADDRESSES TABLE FILE" IS ACCESS BY I2=HDAN(). (2 WORDS).
      C   3E.FILE : "COLLISION TABLE FILE". (3 WORDS).
      C   4E.FILE : "FREE RECORDS FILE". (1 WORD). (NECESSARY IF IND>3).
      C
      C   NADR    : NADR+2 POSSIBLE ADDRESSES FOR THE SECOND FILE.
      C             NADR ARE ACCESSIBLE BY THE "HASH-CODING" FUNCTION
      C          AT NADR+1 : I(1),I(2)
      C          AT NADR+2 : I(3),I(4).
      C   I()     : THE I() CONTAIN THE ACTUAL STATE OF THE INDEX.
      C             THAT PERMITS TO SORT THE OCC() IN CORE MEMORY TO OBTAINS
      C             THE RANK FROM 1 TO I(1) FOR THE NOCC'S.
      C   OCC     : REAL NUMBER OF OCCURANCES (OCC=NOCC+INDX(1)/100000.)
      C             (EX. : OCC=12.0360 MEANS NOCC=12 AND INDX(1)=360).
      C   I2      : I2=HDAN(CAR,A,I,NADR) IN THE CALLING PROGRAM.
      C             IN THE CALLING PROGRAM AND IN THE FUNCTION : HDAN() :
      C   A()     : REAL*8 RADIX 50 TRANSFORM FOR THE 44TH. FIRST CHAR. OF CAR()
      C
      C   IND     : TO ACCESS THE DIFFERENT FUNCTIONS OF DFMDAN() :
      C
      C   =1 : TO INSERT A RECORD OR INCREMENT OCC  --(EXISTING CONFIGURATION).
      C   =2 : TO INSERT A RECORD                   --     (NEW CONFIGURATION).
      C   =3 : TO GET A RECORD.
      C   =4 : TO DELETE A RECORD.
      C   =5 : TO MODIFY A RECORD.
      C
      C
      C   ----------------------------------------------------------------
```

```
C
        INTEGER*2 I(4)
        REAL*8 A(4),B(4)
C
        DATA I/4*0/IZ/0/
C
        GO TO (1,2,3,4,5),IND
C
C  INSERT A RECORD IN AN EXISTING CONFIGURATION.
C  READ I() IN FILE 2 AND FORCE A "LOOKUP" FOR THE OTHERS.
1       READ(1'1)B,J,J
        READ(2'NADR+1)I(1),I(2)
        READ(2'NADR+2)I(3),I(4)
        READ(3'1)J,J,J
         IF(IND.LE.3)GO TO 11
         READ(4'1)J
C  FREE RECORDS OR NOT.
        IF(I(4).EQ.0)GO TO 11
        READ(4'I(4))I(1)
        I(4)=I(4)-1
        GO TO 211
11      I(1)=I(1)+1
        GO TO 211
C
C  INSERT A RECORD IN A NEW CONFIGURATION.
C  ZERO THE FILE 2 AND FORCE AN "ENTER" FOR THE OTHERS.
2       WRITE(1'1)B,IZ,IZ
        N2=NADR+2
        DO 21 J=1,N2
21      WRITE(2'J)IZ,IZ
        WRITE(3'1)IZ,IZ,IZ
        IF(IND.GT.3)WRITE(4'1)IZ
C  FOR THE NEXT ACCESS AFTER RETURN : "LOOKUP" IF IND=1
        IND=1
        I(1)=I(1)+1
C
211     READ(2'I2)I1,I3
        IF(I1.NE.0)GO TO 221
C  FREE RECORD (I1=0) IN 2ND.FILE.
        OCC=1+I(1)/100000.
        WRITE(1'I(1))A,OCC
        WRITE(2'I2)I(1),IZ
        GO TO 241
C
C  RECORD NOT FREE.
221     READ(1'I1)B,OCC
        DO 222 J=1,4
222     IF(A(J).NE.B(J))GO TO 231
C  IDENTITY BETWEEN A() AND B().
        I(1)=I(1)-1
C  FOR CORRECT PRECISION.
        J=OCC
        OCC=J+1+I1/100000.
        WRITE(1'I1)A,OCC
        GO TO 241
231     IF(I3.NE.0)GO TO 232
C  NON-IDENTITY BETWEEN A() AND (B() WITH NO EXISTANT COLLISION (I3=0)
        OCC=1+I(1)/100000.
        WRITE(1'I(1))A,OCC
```

```
      I(3)=I(3)+1
      WRITE(2'I2)I1,I(3)
      WRITE(3'I(3))I(1),IZ,IZ
      GO TO 241
C  WITH ONE OR MORE EXISTANT COLLISIONS.
232   READ(3'I3)I1,IF,IP
C
C  SEARCH AN EQUALITY A()=B() IN THE CHAIN
      READ(1'I1)B,OCC
      DO 234 J=1,4
234   IF(A(J).NE.B(J))GO TO 235
C  IF A()=B()
      I(1)=I(1)-1
C  FOR CORRECT PRECISION
      J=OCC
      OCC=J+1+I1/100000.
      WRITE(1'I1)A,OCC
      GO TO 241
235   IF(IF.EQ.0)GO TO 233
      I3=IF
      GO TO 232
233   OCC=1+I(1)/100000.
      WRITE(1'I(1))A,OCC
      I(3)=I(3)+1
      WRITE(3'I3)I1,I(3),IP
      WRITE(3'I(3))I(1),IZ,I3
C
C  ACTUALISATION OF THE I().
C  TO INCREASE THE VELOCITY (BUT DECREASE SECURITY),
C  DO ONCE IN THE CALLING PROGRAM AND ADD INTEGER*2 I(4).
241   WRITE(2'NADR+1)I(1),I(2)
      WRITE(2'NADR+2)I(3),I(4)
      RETURN
C
C  COMPLETE THE NECESSARY FUNCTIONS (3 TO 5) FOR THIS SUBROUTINE.
C
3     CONTINUE
4     CONTINUE
5     CONTINUE
      RETURN
      END
C  ***********************************************************
C
      SUBROUTINE OUTFIL(IMP,NLOGIC,M1,J2,X,IND)
C
C  STORE X() IN FILE NLOGIC AND STATISTICS ALL THE M1TH.
C
      REAL*4 X(17),XB(17),SXB(13),Y(128)
C
C  STORE.
      I=IND
      IF(I.EQ.0)I=M1
      WRITE(NLOGIC'I)X
      IF(IND.NE.0)RETURN
```

```
C
C  STATISTICS. (ON FILE 4; 17 COLUMNS).
C
         DO 2 K=1,13
         J1=0
         XB(K)=0.
         DO 1 J=1,M1
         READ(NLOGIC'J)X
         IF(X(K).EQ.0.)GO TO 1
         J1=J1+1
         Y(J1)=X(K)
1        CONTINUE
2        IF(J1.GT.0)CALL RTYPE(J1,XB(K),STDEV,CV,SXB(K),Y)
         XB(14)=M1*(J2+0.5)
         XB(15)=X(15)
         XB(16)=X(16)
         XB(17)=X(17)
         J2=J2+1
C  STORE THE RESULTS FROM M1+1 TO M INDEX.
         WRITE(NLOGIC'J2+M1)XB
C  OUTPUT THE RESULTS ON THE CONTROL TERMINAL.
         WRITE(IMP,100)J2,(XB(J),J=1,16),SXB
100      FORMAT(' STATISTICS - GROUP NBR.',I4,' :'/2(1X,16F5.0/))
C
         RETURN
         END
```

TSX

TSX-Plus Real-time Support Facility

John Yardley

JPY Associates Ltd

1. What is a "Real-Time" Application?

A real-time application may be defined as one which demands some process to be performed within a specified time of some event. If the process is not performed, then future events may be affected. For example, a simple process to transfer data from an analogue to digital converter to a disk must perform each transfer within the sampling period of the digitising clock, otherwise data will be irretrievably lost. On the other hand, a desk-calculator say, is unlikely to be classed as a real-time process since the operator can wait almost any amount of time for the result without upsetting the calculation.

Before we look at how real-time events are handled under TSX-Plus we need to know something of the way TSX-Plus works.

2. How does TSX-Plus work?

Although TSX-Plus services several users, it does, of course only run on a single processor. This being the case, it can only execute one piece of program code at any one time. This will be either:

a) A user's program

or b) The TSX-Plus monitor (which includes device handlers)

The monitor is only ever executed in response to an EVENT. This may be a software event such as an EMT or TRAP instruction - or a hardware event such as a device (eg disk, terminal or clock) INTERRUPT. Once a given user program is executing, then it will continue executing until some event occurs. When this happens, TSX-Plus will first respond to the event in the appropriate way (eg taking a character from the keyboard buffer) and then pass control to the TSX-Plus SCHEDULER.

It is the job of the scheduler to decide which user job should be run after the event processing has been completed. Sometimes, when the user program has requested a service of TSX-Plus by means of an EMT instruction, the scheduler will return control to the same user program. At other times, the scheduler may decide that a different user should be given processor time.

Whenever the scheduler decides to "move" the processor from one user job to another, it must "remember" exactly the state of the current job so it may be continued at some later stage. It need only remember data items which may be destroyed by the new user program. These include the general-purpose registers (GPRs), the page address registers (PARs) and the floating-point registers (if any). The GPRs include the program counter (PC) and stack pointer (SP), so define the address at which the user program was suspended. The contents of these "shared" data areas are often said to give the CONTEXT of the job, so the scheduler is, in fact, a CONTEXT SWITCH.

2.1 How does the scheduler decide which job should run?

The scheduler first establishes the PRIORITY of every active user (NOTE: Priority here, relates strictly to software priority NOT the hardware priority associated with devices on the bus). It then runs the job which has the highest priority. Jobs which have the same priority are QUEUED to execute one after the other.

The priority of jobs can only ever change when an event occurs, since it is events which drive the scheduler. The scheduler is normally guaranteed to run periodically, since interrupts will constantly be generated by the processor clock.

As an example of the scheduler at work we may consider a system with 3 jobs executing at one of three priorities thus:

JOB £1	NORMAL	RUNNING
JOB £2	LOW	
JOB £3	LOW	

An event occurs which makes JOB £2 HIGH priority. This immediately causes JOB £2 to run thus:

JOB £1	NORMAL	
JOB £2	HIGH	RUNNING
JOB £3	LOW	

When an event occurs to make JOB £3 HIGH, then a later event to make JOB £1 HIGH as well, JOB £2 will continue running so long as its priority is the same or greater than any other job, thus:

JOB £1	HIGH	
JOB £2	HIGH	RUNNING
JOB £3	HIGH	

As soon as the priority of JOB £2 drops, JOB £3 will run. This is because it became a high priority job <u>before</u> JOB £1, thus:

JOB £1	HIGH	
JOB £2	LOW	
JOB £3	HIGH	RUNNING

2.2 What are the priorities for a TSX-Plus job?

TSX-Plus assigns 12 priority levels, split into 4 classes. Jobs in the lowest three priority classes are "time-sliced". This means that they are automatically suspended after a fixed period of time. The highest priority class is reserved only for REAL-TIME COMPLETION ROUTINES (this term is used synonymously with INTERRUPT SERVICE ROUTINES) and at this level jobs may run indefinitely. The levels are as follows:

REAL-TIME COMPLETION ROUTINES	level 7 level 6 level 5 level 4 level 3 level 2 level 1		NOT TIME-SLICED
	level 0	(QUAN1A)	TIME-SLICED
HIGH PRIORITY	ACTIVATION CHARACTERS	(QUAN1)	
	I/O COMPLETION	(QUAN1A)	
NORMAL PRIORITY	COMPUTE BOUND (Physical line & Detached Job)	(QUAN2)	
LOW PRIORITY	COMPUTE BOUND (Virtual line)	(QUAN3)	

As we have said, once a job has been scheduled to run as either a level 0 completion routine, a high, normal or low priority job, it will continue to do so until either another job achieves a higher priority or its time-slice elapses. The value of the time-slice may be selected by the system manager at sysgen time or by means of a SET command. There is not usually much to be gained by altering these time-slices from their "delivered" values unless the same program is always being run on all terminals. In general, QUAN1 and QUAN1A should always be long enough to allow a job to get BLOCKED (ie waiting for some I/O to complete or a character to be typed). If a program's I/O activity is interleaved with a regular amount of computation (say 5 secs), then QUAN2 should be at least as long as the computation. It is always better that time-slice values be too great than too small.

Important points to note:

1. Since real-time priorities are not time sliced, they can "hog" the processor indefinitely.

2. Low priority jobs (ie compute-bound jobs on virtual lines) will NEVER be executed while there is any other job at a higher priority. This is a fairly likely event on a system with many active users.

3. All user jobs execute at a processor priority of zero - hence interrupts should always be serviced by TSX-Plus.

2.3 More about context switching

We have already noted that to change context from one user to another involves saving and restoring:

1. General purpose registers (GPRs)

2. Page address registers (PARs)

3. Page descriptor registers (PDRs)

4. Floating point registers

On an LSI-11/23 it may take up to 2 milliseconds to switch between memory resident jobs.

It is often asked if it takes this amount of time to switch from a user job to the TSX-Plus monitor each time an event occurs - since surely TSX-Plus uses GPRs, and PARs at very least. The answer to this is that it only takes a few microseconds to execute the switch to TSX. This is because TSX-Plus makes use of the PDP-11 KERNEL MODE of operation when executing. Whenever a event occurs while the processor is executing a user program, the processor status word is reloaded from the TRAP VECTOR. This is set up such that processor goes automatically into kernel mode. When this happens, the PDP-11 hardware switches to another set of GPRs, PARs and PDRs. On return from the trap, the processor is switched back into USER MODE.

3. Real-time Applications

There are basically three approaches to the handling of real-time applications under TSX-Plus. These are:

1. Programmed I/O

2. Interrupt service routines

3. Device Handlers

3.1 Programmed I/O (See Appendix I for example)

In this method, the user must first map the I/O page into his own user address space using the necessary TSX-Plus EMT. He can then do a programmed data transfer without reference to the device vector. For example, to read a character from a keyboard would involve some program code like this:

```
HANG:  TSTB  @£CSR                ; NOTE THAT THIS IS
       BPL   HANG                 ; A COMPUTE-BOUND LOOP
       MOV   @£BUFF,R0
       BR    HANG
```

Because the flag check is compute-bound (after all, TSX-Plus doesn't know CSR is a device register), there is no guarantee when it will be done, since it is highly eligible for being time-sliced out. When the flag goes up, it is most likely the processor will be in the context of another job. Programmed I/O like this would be hopeless for say sampling a serial line, since failure to return to the context of the job quickly enough would result in a data overrun.

TSX-Plus version 2.2 provides a solution to this by providing an EMT to lock the processor onto a user's job. It does not change the processor's priority, so will not lock out events. Given version 2.2, programmed I/O must be fastest way to sample devices. It is of course, at the expense of other users, so must used carefully.

3.2 Interrupt service routines (See Appendix II for example)

TSX-Plus version 2.0 (and later) provides a mechanism for connecting interrupts to user real-time service routines resident in the user's address space. This involves:

1. Re-mapping the I/O page to gain access to device buffers and registers.

2. Setting up the address (in user space) of the user's completion routine.

These functions are accomplished by means of special TSX-Plus EMTs. The EMT to connect the service routine to the vector address does five things:

1. Checks that Real-time support has been sysgenned into TSX-Plus monitor.

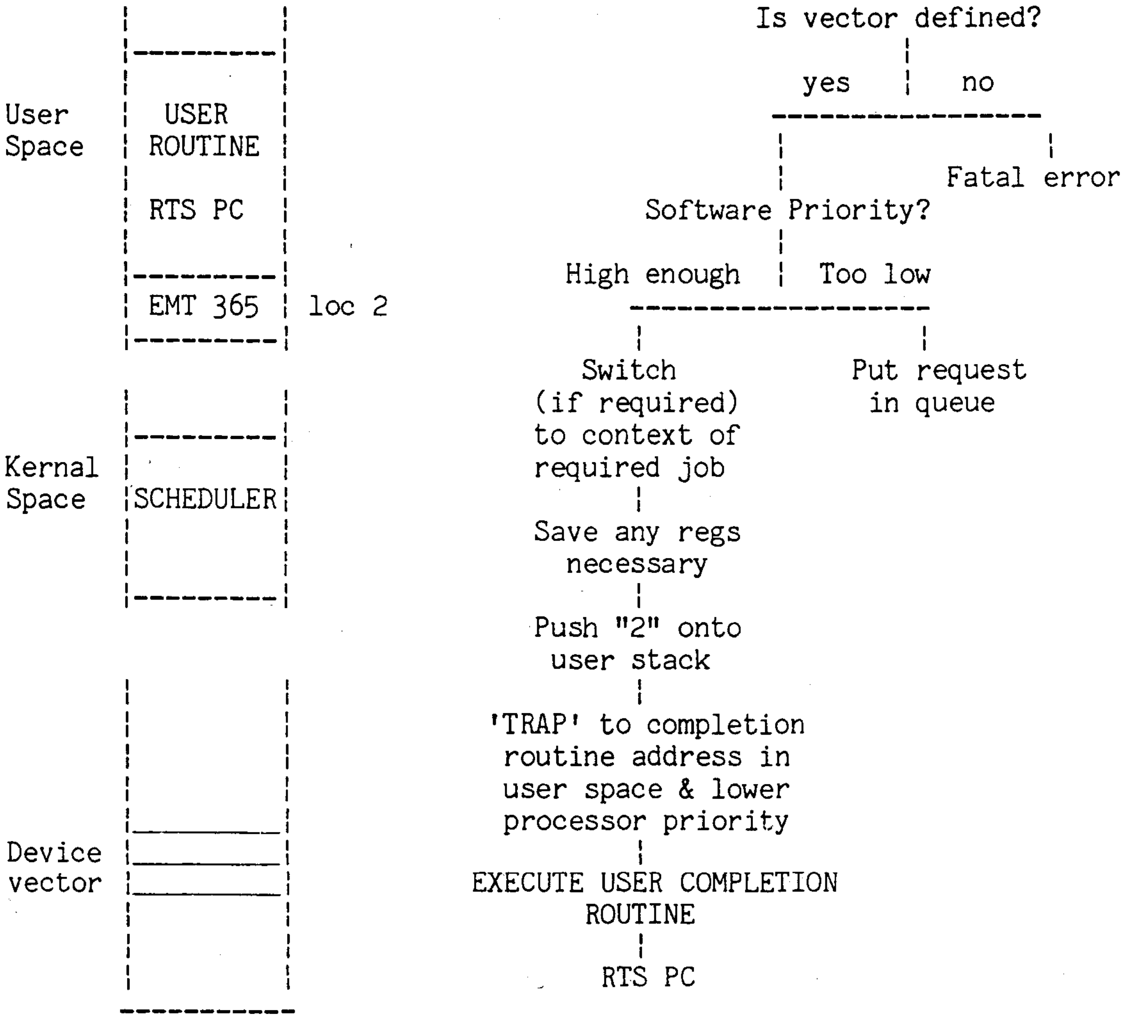
2. Checks that there is enough space in the despatch tables for the routine address.

3. Checks that no other job is connected to this interrupt.

4. Puts "EMT 365" instruction into location '2' of user space (this is to enable user to exit from completion routine to TSX-Plus - on entry '2' is pushed onto stack)

5. Enters vector and job information (ie completion routine address, job number, etc.) into TSX-Plus despatch tables.

When an interrupt occurs.....



Interrupt service routines (ISRs) written for RT-11 cannot be used in a TSX-Plus application without slight modification. This is because there are several fundamental differences in the way the routines are called. These are:

1. Under RT-11 it is usual to put the address of the ISR directly in the device interrupt vector. This is because the user program and monitor are in the same address space. Under TSX-Plus the user must call a special EMT to set up the ISR address.

2. In RT-11 interrupt service routines, it is recommended that the first instruction be a ".INTEN" request - this informs RT-11 that the ISR is in operation, saves registers and changes processor priority. With TSX-Plus completion routines, the .INTEN request is neither necessary nor supported - TSX-Plus always knows when a user interrupt occurs.

Problems with Interrupt Service Routines:

1. If the processor is not in the context of the ISR job when the interrupt occurs, then because of the context switch necessary, there can be a long delay before the routine is entered - this can be 2mS if the job is in memory and 100 mS or more if job is on disk. This can result in loss of data items.

2. With every hardware interrupt there is an associated TSX-Plus overhead. This is the time taken by TSX-Plus to discover the routine entry address, change the processor priority, save any registers used by the real-time job, then 'trap' to the start of the service routine.

To prevent disk swapping delays, it is recommended always to have the job LOCKED into memory. This prevents the job being swapped to disk while waiting for an interrupt. TSX-Plus provides a special EMT for this purpose.

The upshot of all this is that in general, TSX-Plus real-time completion routines cannot handle interrupts at much more than 100 per second. If interrupt rates greater than this are required, then some other approach should be adopted.

3.3 Device Handlers

The use of an RT-11 device handler gives the fastest possible response to interrupts since there is no context switch overhead. Device handlers can be used for purposes other than straight block-orientated transfers to and from peripherals. The handler provides a window on TSX-Plus itself, and by clever use of SET options and .SPFUNs, user programs can do almost anything.

Conventional device handlers always call TSX-Plus (or RT-11) on every interrupt via "$INTEN" (this is implicit in the .DRAST macro). This performs register saving, state checking, and processor priority lowering. On non-DMA devices this can be quite an overhead due to sheer volume of interrupts (the TU58 is a classic case). If the service routine is short, it is possible to do away with the call to $INTEN in order to save time. If this is done, then the service routine must be exited with an "RTI" and not an "RTS". For long service routines, this is a dangerous practice. Leaving the processor executing at a high priority may "lock out" other interrupts - including those from user terminals.

4. Summary

In summary, we may compare the use of programmed I/O, interrupt service routines and device handlers for different classes of application.

APPLICATION	TSX-Plus Version Number		
	<2.0	2.0,2.1	>2.1
Sampling as fast as possible in bursts	Handler	Handler	Prog I/O (job locked)
Constant high-speed interrupts	Handler	Handler	Handler
Bursts of high-speed samples (up to 30 KHz) say lasting 1 sec every 10	Handler	Handler	Prog I/O or Handler
Constant sampling at rates of less than 100 Hz	Handler	ISR or Handler	ISR or Handler

From the table, it can be seen that handlers solve most real-time applications except the "flat-out" situation. However, it should be born in mind that handlers are usually more difficult to write and debug and that TSX-Plus must be re-sysgenned to incorporate a new handler. Also, in a time-sharing environment, it is much more important that the handler itself be "correct".

Real-time support is a compromise in any time-sharing system. The eventual best solution will always be a trade-off between speed of operation, programming difficulty, and effect on other time-shared users.

```
                                    ; APPENDIX I - PROG TO DISPLAY CHARS RECEIVED FROM A SERIAL LINE
                                    ; USING TSX+ REAL-TIME SUPPORT - COPYRIGHT (C) JPY ASSOCIATES LTD!
                                    ;
                                    ; PROGRAM 1 - PROGRAMMED I/O METHOD
                                    ;
        176520                              CSR=176520
                                    ;
                                            .MCALL  .TTYOUT,.DEVICE,.EXIT,.PRINT
                                    ;
000000  012700  000072'             GO:     MOV     £REMAP,R0                   ;MAP IN THE I/O PAGE
000004  104375                              EMT     375
000006  103004                              BCC     REMOK                       ;ERROR?
000010                                      .PRINT  £REMERR                     ;YES - ERROR MESS
000016                                      .EXIT                               ;BYE
000020                              REMOK:  .DEVICE £AREA,£TIDY                 ;TIDY-UP ON EXIT
000040  012737  000001  176520              MOV     £1,@£CSR                    ;ENABLE READ
000046  105737  176520              HANG:   TSTB    @£CSR                       ;READY?
000052  100375                              BPL     HANG                        ;NO - HANG HERE
000054  013700  176522                      MOV     @£CSR+2,R0                  ;GET CHAR INTO R0
000060                                      .TTYOUT                             ;TYPE ON TERMINAL
000064  005237  176520                      INC     @£CSR                       ;RE-ENABLE READ
000070  000766                              BR      HANG                        ;WAIT FOR NEXT
```

```
                                               ;
                                               ; EMT BLOCKS ETC
                                               ;
000072     005     140                         REMAP:  .BYTE   5,140                    ;REMAP EMT BLOCK
000074                                         AREA:   .BLKW   3                        ;.DEVICE EMT BLOCK
000102  176520                                 TIDY:   .WORD   CSR                      ;TURN OFF INTERRUPTS ON EXI
000104  000000  000000                                 .WORD   0,0                      ;BY CLEARING CSR
                                               ;
                                               ; ERROR MESSAGE
                                               ;
                 .NLIST  BIN
000110  REMERR:  .ASCIZ  /TSX MAPPING ERROR - BYE/
                 .END    GO

                                              ; APPENDIX II - PROG TO DISPLAY CHARS RECEIVED FROM A SERIAL LINE
                                              ; USING TSX+ REAL-TIME SUPPORT - COPYRIGHT (C) JPY ASSOCIATES LTD!
                                              ;
                                              ; PROGRAM 2 - INTERRUPT SERVICE ROUTINE METHOD
                                              ;
        000320                                          VECTOR=320
        176520                                          CSR=176520
                                              ;
                                                        .MCALL  .TTYOUT,.DEVICE,.EXIT,.PRINT,.SPND
                                              ;
000000  012700  000112'                       GO:       MOV     £REMAP,R0               ;MAP IN THE I/O PAGE
000004  104375                                          EMT     375
000006  103004                                          BCC     REMOK                   ;ERROR?
000010                                                  .PRINT  £REMERR                 ;YES - ERROR MESS
000016                                                  .EXIT                           ;BYE
000020                                        REMOK:    .DEVICE £AREA,£TIDY             ;TIDY-UP ON EXIT
000040  012700  000114'                                 MOV     £CNCT,R0                ;CONNECT UP SERVICE ROUTIN
000044  104375                                          EMT     375
000046  103004                                          BCC     CNCTOK                  ;ERROR?
000050                                                  .PRINT  £CONERR                 ;YES - ERROR MESS
000056                                                  .EXIT                           ;BYE
000060  052737  000101  176520                CNCTOK:   BIS     £101,@£CSR              ;ENABLE INTERRUPTS
000066                                                  .SPND
                                              ;
                                              ; INTERRUPT SERVICE ROUTINE
                                              ;
000074  013700  176522                        SERVE:    MOV     @£CSR+2,R0              ;GET CHAR INTO R0
000100                                                  .TTYOUT                         ;TYPE IT ON TERMINAL
000104  005237  176520                                  INC     @£CSR                   ;RE-ENABLE READ
000110  000207                                          RTS     PC                      ;BACK TO TSX
                                              ;
                                              ; EMT BLOCKS ETC
                                              ;
000112     005     140                        REMAP:    .BYTE   5,140                   ;REMAP EMT BLOCK
000114     011     140                        CNCT:     .BYTE   11,140                  ;CONNECT EMT BLOCK
000116  000320  000074' 000007                          .WORD   VECTOR,SERVE,7
000124                                        AREA:     .BLKW   3                       ;.DEVICE EMT BLOCK
000132  176520                                TIDY:     .WORD   CSR                     ;TURN OFF INTERRUPTS ON EX
000134  000000  000000                                  .WORD   0,0                     ;BY CLEARING CSR
                                              ;
                .NLIST BIN                    ; ERROR MESSAGES
000140  REMERR: .ASCIZ  /TSX MAPPING ERROR - BYE/
000170  CONERR: .ASCIZ  /TSX ISR CONECTION ERROR - BYE/
                .END    GO
```

New & Revised DECUS Library Submissions

11-299 (rev) by Michael N. Levine, Naval Weapons Center, China Lake, CA.

INDEX is a cross referencing program that does for FORTRAN what CREF does for MACRO. A source program passed through INDEX will be checked for all of its variable name and label usage. The results will then be listed in alphabetical order, listing all the variable names and labels used in the program, the lines on which they were used, and how they were used. This revision fixed bugs and expanded the option switches.

11-472 (rev) by N. A. Bourgeois, Jr., Sandia National Laboratories, Albuquerque, NM.

KB is an RT-11 V4.0 device independent terminal handler. It can be used in either the foreground or the background (but not both simultaneously) to read and write to a DL11/DLV11 controlled terminal. Conditional code is included for use with extended memory addressing. Eleven set options are also included to permit configuration of the handler. The revision fixed a bug in the abort code.

11-595 (new) by Keith W. Hadley, University of Toronto, Ontario, Canada.

HP.SYS is a Hewlett-Packard 7470A plotter handler to be used in an RS-232 environment with XON-XOFF handshaking protocol. It is a read/write handler useful for communicating with the plotter. Since many laboratory devices use XON-XOFF, this handler might be easily modified for such devices.

11-596 (new) by Keith W. Hadley, University of Toronto, Ontario, Canada.

LG.SYS is a serial line printer handler designed specifically for the CENTRONICS 739 graphic printer. This handler takes into account many of the 739's features that the LS.SYS handler does not.

11-599 (new) by Karl Dunn, SCI Systems, Inc., Hazel Green, AL.

This is a set of routines that derive an optimal path through a set of coplanar points. They return a sequence of integers that represent ordinal point identifiers, the points having been given as a sequence of coordinates.

11-603 (new) by Dan Dill, Boston University, Boston, MA.

This TEXT System is a collection of software tools, which in combination with DECUS RUNOFF (11-530), form a comprehensive text processing system for technical manuscript preparation with NEC Spinwriters and the Technical Math / Times Roman type thimble.

11-605 (new) by John Crowell, Crow4ell, Ltd., Los Alamos, NM.

This package contains TECO command files which will perform sufficient modification of RT-11 V4.0 uncommented source files (as in the binary distribution kit) to enable generation of an RT-11 system (SJ of FB) called FART-11 to run on the SBC-11/21 FALCON microcomputer. There is also a patch for the file, SYSGEN.CND, which will include the FALCON options in the sysgen process. If you must have FART-11, this is the recommended procedure.

11-606 (new) by G. Laurent and S. Rozenberg, INFI, Chaville, France.

SPAL-11, for Structured Programming using Assembly Language, is a set of macros, which when incorporated in the default macro library, SYSMAC.SML, provides the ability to write well constructed, clear, and maintainable programs. It makes sophisticated use of the MACRO-11 assembler and where possible the code generated has been optimized.

11-608 (new) Ron Tenny, G. W. Tenny Company, Inc., Scottsville, NY.

This is an electronic bulletin board message mailing facility written in DIBOL. The board facility allows mail to be sent to user and group accounts, and to a printer. All mail is password protected and clearance coded. This facility requires the use of a single key ISAM file.

06-Apr-83/NABourgeois

DECUS

DIGITAL EQUIPMENT COMPUTER USERS SOCIETY
ONE IRON WAY, MR02-1/C11
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, MR02-1/C11
Marlboro, Massachusetts 01752 USA

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