

Contributions to the newsletter should be sent to:

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

Other communications can be sent to:

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

or

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

Table of Contents

User Input	
Terminal I/O Routines - - - - -	2
Portran Virtual Array Support - - - - -	4
User Requests	
FB Problem on the 11/23 - - - - -	10
Past Symposium Information	
1981 European Decus Wishlist - - - - -	11
Symposium Tape Copy	
Medium Guidelines - - - - -	14
Decus Library	
Plotting the HP9872s on the IEEE 488 Bus - - - - -	14
Algeb - Language for Algebra - - - - -	15
TSXLIB - for RT-11 Fortran - - - - -	15
NEC Spinwriter Support - - - - -	16
System for Technical Manuscript Preparation - - - - -	16
SPR's	
RT-11 with SIMRT Problem - - - - -	21
MU-Basic/RT-11 V2.00s problem - - - - -	21
Closing Multiple Logical Units Problem - - - - -	24

Copyright © 1981 Digital Equipment Corporation
All Rights Reserved

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.

USER INPUT

```

;
; QUERY.MAC
;
; Subroutines for displaying a question on the terminal and reading/
; displaying the answer. Answers may be in an INDIRECT COMMANDFILE
; as the .GTIN macro is used for input from terminal.
;
; Input Parameters:
; -----
; QUEST; Byte string, is displayed question.
; NADEF; Max. permitted nr. of characters, values or
;         ~ default value (depends on entry point).
; MODE; -1 no action, immediate return.
;         0 display question, read answer
;         =answer mode.
;         1 the same as for 0, but mode is changed to -1
;         =repeat mode.
;         2 display question and previous given answer.
;         =display mode.
;
; OUTPUT(INput in display mode):
; -----
; GLBANS; answer (given previous answer in display mode)
;
; FORTRAN CALL'S:
; *****
;
; QUEST , GLBANS , NADEF ,MODE
;
; CALL QA('Question',Byte array ,Max. nr. of char. ,MODE)
; CALL QL(' ' ' ',Logical variable,Default value , ' )
; CALL QF(' ' ' ',Real variable , ' , ' )
; CALL QI(' ' ' ',Integer variable , ' , ' )
; CALL QM(' ' ' ',Integer array ,Wanted nr. of val., ' )
; CALL QV(' ' ' ',Real array , ' , ' , ' , ' )
;
; CALL QO('Strings to be Printed')
; CALL BELL
;
; Default answer values can be in the argument list in the case of a single
; value request or by setting the array in the multiple value request( not
; for QA: answer is left blank).
;
; Multiple input of numbers on 1 line must be separated by comma or blank.
; Any number format on input is acceptable and conversion is made to the
; format needed. Multiple input of same number by *-operator(see also
; FRFMT earlier published in the MINITASKER, FRFMT is written as a sub-
; routine and the + sign preceding a number is accepted also).
;
; Multiple concatenated blanks in answer are reduced to a single one.
;
; On error condition(too much or less numbers, integer value out of range,
; QL only accepts YES or NO, etc.) bell rings and question is again
; displayed.

```

```

; Repeat mode(MODE=-1) is usefull when a list of succeeding questions is
; programmed. If on execution one wants to correct a given answer without
; asain answering the following ones, just JUMP to the question with MODE=-1.
;
; Available as mainly MACRO-11 coded source. A version using floatings point
; processor for number conversion is also available. However a complete in
; FORTRAN coded source is also available but generates as much as 4* more
; code as the MACRO version.
;
;
; Fortran example:
; *****
;
;       BYTE NAME(15),SWTC
;       DIMENSION REGPAR(5),IPAR(4)
;C
;       MODE=0
;C
;C Array defaults(or use DATA statement):
;C
;       REGPAR(1)=1.1
;       REGPAR(2)=2.
;       REGPAR(3)=3.3
;       REGPAR(4)=4.
;       REGPAR(5)=5.
;       IPAR(1)=1
;       IPAR(2)=2
;       IPAR(3)=3
;       IPAR(4)=40
;C
;       CALL Q0('-----')
;10      CALL QA('1. Name of Person?          ',NAME,15,MODE)
;20      CALL QI('2. Registration duration in sec.? [100]:',IDUR,100,MODE)
;30      CALL QL('3. New Person?              [No]:',SWTC,.FALSE.,MODE)
;40      CALL QV('4. Give registr. param. [1.1,2.,3.3,4,5]:',REGPAR,5,MODE)
;50      CALL QM('5. Give sample param.      [1,2,3,40]:',IPAR,4,MODE)
;       CALL Q0('-----')
;       CALL QI('Correct above answer? If so give nr. 1-5:',IANS,0,0)
;       IF (IANS.LE.0.OR.IANS.GT.5) GOTO 100
;       MODE=1
;       GOTO (10,20,30,40,50)IANS
;100     CALL REGIS(NAME,IDUR,SWTC,REGPAR,IPAR)
;       END
;

```

H. T. M. Haenen
 Dept. Clin. Neurology AZG
 P.O. Box 30.001
 9700 RB GRONINGEN
 Holland

Enclosed is a description of a method which enables two FORTRAN programs to use VIRTUAL arrays concurrently in the background and foreground without using the XM monitor. It might be of use to others who, like we GAMMA 11 users, are restricted to the foreground/background monitor. The method has been tested in several Australian hospitals for over 18 months, and has so far proven reliable. You might like to include it in a future issue of "Minitasker". If anyone has any questions about the method I will be happy to try to answer them.

SIMULTANEOUS USE OF EXTENDED MEMORY BY BG AND FG WITH FB MONITOR

R. Fulton
 Dept of Nuclear Medicine
 Royal Prince Alfred Hospital
 SYDNEY. N.S.W. 2050.

The patches described here permit background FORTRAN V2.5 programs to make use of extended memory, concurrently with MACRO-11 or FORTRAN foreground extended memory programs. Previously, as far as could be determined, background and foreground jobs could only utilize extended memory concurrently with the XM monitor and PLAS support.

A FORTRAN VIRTUAL program will normally use extended memory starting at 28K words. These patches alter the base address in extended memory of a background FORTRAN program's VIRTUAL data, and bypass the trap message "Error 64 - Virtual Array Initialisation Failure".

The only decision to be made by the programmer is at which address in extended memory allocation of the background program's VIRTUAL data should begin. This address should be high enough in extended memory to leave enough space below it for the foreground program's needs.

Two ways of implementing the patches are:

Method 1: Patch the FORTRAN library. Before installing the patch in the FORTRAN library, a decision should be made as to the highest extended memory address which will ever be used by any foreground program when a FORTRAN background program using extended memory is co-resident. The base address of the background program's VIRTUAL DATA can then be chosen as any 32 word boundary above that.

Method 2: Identical to the first, except that the modules extracted from the FORTRAN library for patching are not reinserted. Then, if the module names are included explicitly in the linker command string prior to FORLIB, they will be linked in preference and the program will relocate its VIRTUAL data upwards. However, should a background FORTRAN program require all of extended memory, the module names can be omitted, thus linking the unpatched modules from the FORTRAN library and causing VIRTUAL data to begin at 28K. This method also removes the need for separate FORTRAN libraries, one patched and the other unpatched, for linking with background and foreground FORTRAN extended memory programs.

The Patches

The value substituted in all patches but the first will depend on the foreground's memory requirements, and the amount of extended memory your system has. It fixes the base address at which FORTRAN begins storing VIRTUAL arrays, which must be above the highest address in extended memory ever used by any foreground job. For example, at RPAH we calculate that the foreground will not use extended memory above 80k words. Therefore, we patch FORTRAN to begin its virtual arrays at the next 32 word boundary above that, 500000 (octal), or the 5000 th (octal) 32 word boundary in memory. Please note that if an error is made, and the VIRTUAL arrays start too low, they will clash with the foreground's data. This will not be reported by the system, but both programs will give erroneous results.

When you have determined the (octal) address at which the VIRTUAL arrays are to start, cross out the last 2 zeroes. The result is the value to be inserted in the patches. The command files which build the package are currently set to fix the base address of VIRTUAL arrays at a default of octal 500000. To depart from this, read the section "Choosing different VIRTUAL array base address".

The first patch aborts the trap message "Error 64 - Virtual array initialization failure" which is ordinarily needed to signify an attempt by two concurrent jobs to access extended memory.

Having been careful to ensure that the two jobs will use separate parts of extended memory, the trap is not needed.

Choosing different VIRTUAL array base address

An examination of the command file FORPAT.COM shows that some of the patches insert the value 5000, and others patch consecutive bytes with 0 and 12 respectively. These are two forms of the same patch. In the latter case, the two bytes comprising the octal pattern 5000 are not in the same word.

The high byte of 5000 is 12, and the low byte is 0. To select a different base address for VIRTUAL arrays, for example 620000 (octal), substitute the high and low bytes of 6200 (14 and 200) for 12 and 0 respectively in FORPAT.COM.

```
!
! PATCHES FORTRAN LIBRARY MODULES %VRINT AND %VIRN%
!
! AT PRESENT PLACES VIRTUAL ARRAYS BASE ADDRESS
! AT 500000 (OCTAL)-:
! IF THIS DOES NOT SUIT, EDIT THIS COMMAND FILE CHANGING ALL
! OCCURRENCES OF 5000 TO DESIRED VALUE
!
! SEE DOCUMENTATION FOR VALUE TO BE INSERTED
! (DEPENDS ON YOUR MEMORY REQUIREMENTS)
!
```

```
RU LIBR
SRC:VRINT=SRC:FORLIB/E
%VRINT
```

```
!Integer*2
SRC:VIRIN=SRC:FORLIB/E
%VGETI
```

```
!Logical*1 (BYTE)
SRC:VIRLN=SRC:FORLIB/E
%VGETL
```

```
!Real*4
SRC:VIRFN=SRC:FORLIB/E
%VGETF
```

```
!Complex
SRC:VIRQN=SRC:FORLIB/E
%VGETQ
```

```
!Double precision
SRC:VIRDN=FORLIB/E
%VGETD
```

C

```
RU SIFF
SRC:VRINT.OBJ/A
```

```
336
103400
~Y
SRC:VIRIN.OBJ/A
```

```
550
5000
~Z
403
0
12
~Y
SRC:VIRQN.OBJ/A
```

```
550
5000
~Z
403
0
12
~Y
SRC:VIRDN.OBJ/A
```

```
576
5000
~Z
403
0
12
~Y
SRC:VIRLN.OBJ/A
```

```
562
5000
~Z
403
0
12
~Y
SRC:VIRFN.OBJ/A
```

```
560
5000
~Z
403
0
12
~Y
~C
RU LIBR
SRC:FORLIB=SRC:FORLIB,VRINT/R,VIRIN/R,VIRLN/R,VIRFN/R
SRC:FORLIB=SRC:FORLIB,VIRDN/R,VIRQN/R
~C
```

VIRTUAL ARRAY INTERFACE BETWEEN FORTRAN AND MACRO-11

Two macros have been written which enable integer VIRTUAL arrays set up within a FORTRAN program to be accessed directly from a MACRO-11 subroutine. These macros employ routines \$VGETI and \$VSTOI from the FORTRAN library which read from and write to an integer location in VIRTUAL memory respectively. These subroutines can be called from within MACRO-11 if their names are declared as global, and the program is linked with the FORTRAN library. Similar macros could be written for VIRTUAL floating point arrays using \$VGETF and \$VSTOF. These four subroutines will naturally behave identically in FORTRAN and MACRO-11, so regardless of whether the FORTRAN library has been patched or not, a reference to a particular array element from MACRO-11 will always be successful using these macros.

VGETI

Returns Ith or (I,J)th element of an integer VIRTUAL array declared in a FORTRAN program to specified destination in a MACRO-11 subroutine.

Macro-11 Call: VGETI I, OFFST, DEST [,J, IDIM]

where: I - the first index of a 2-D array element, or sole index of a 1-D array element. (Note: processing of an entire 2-D array is faster if it is treated as 1-D).

OFFST-- always 2, 4, 6, 10 etc.
is the R5 offset to the chosen VIRTUAL array

DEST-- is the entity to receive the value. (see restrictions below)

J-- (optional) second index of a 2-D array element

IDIM-- (optional) first dimension in 2-D array declaration.

Restrictions: The above arguments can take any form, ie, addresses of any mode, symbolic variables, absolute memory locations, or octal constants except :-
(1) I must never be R1
(2) DEST must never be R0, R1 or R2.

Example:

```
FORTRAN
INTEGER A,B
VIRTUAL A(5,6), B(32,32), C(12)
TEMP = 0.32
```

```
CALL MACSUB(TEMP,C,A,B,32)
END
```

```
MACRO SUBROUTINE
MACSUB:: VGETI #12,4,R4 : GET C(10) IN R4
        VGETI #3,6,R3,#2,#5 : GET A(3,2) IN R3
        :
        ;PROCESS MATRIX B, TREATING IT
        ;AS 1-D FOR SPEED
```

```

;
MOV @12(R5), BDIM ; GET DIMENSION
MOV BDIM, R3
MUL BDIM, R3 ; DIMENSION SQUARED
1$: VGETI R3, 10, R4 ; PROCESS ELEMENTS
.
.
SOB R3, 1$
```

VSTOI

Writes a value from a MACRO-11 subroutine to the Ith or (I,J)th element of an integer VIRTUAL array declared in a FORTRAN program.

MACRO-11 Call:

VSTOI I, OFFST, VAL [,J, IDIM]

where: I-- first index of a 2-D array element, or sole index of a 1-D array element. (Note: processing of a 2-D array is faster if it is treated as 1-D).

OFFST-- always 2, 4, 6, 10, etc.
The R5 offset to the chosen VIRTUAL array.

VAL-- the value to be stored in the array element.

J-- (optional argument) second index of a 2-D array

IDIM-- (optional argument) first dimension in 2-D array declaration.

Restrictions: The above arguments can take any form, ie, addresses of any mode, symbolic variables, absolute memory locations, or octal constants except :-
(1) I must never be R1.

Example:

FORTRAN	MACRO SUBROUTINE
INTEGER ACT	; PUT CONTENTS OF R3 IN JARR(2,6)
VIRTUAL IARR(20),JARR(10,10)	;
	MACSUB:: VSTOI #2,6,R3,#6,#12
	; PUT VALUE ON TOP OF STACK
	; IN IARR(12)
	;
CALL MACSUB(IARR,ACT,JARR)	
END	VSTOI #14,2,(SP)
.MACRO VGETI I, INDX, DEST, J, IDIM	
MOV RO,-(SP)	; SAVE REGISTERS
MOV R1,-(SP)	
MOV R2,-(SP)	
.NARG NUM	; GET NO OF ARGUMENTS

```

; WERE THERE 5 ARGUMENTS?
; YES: ASSEMBLE THESE INSTRUCTIONS
; NO: ASSEMBLE ONLY THIS INSTRUCTION
; ALWAYS ASSEMBLE FROM HERE ON
; PUSH ADDRESS OF ELEMENT WITHIN ARRAY
; PUSH ARRAY ADDRESS
; GET THE VALUE OF THE ELEMENT ON STACK
; POP IT OFF STACK
; RESTORE REGISTERS

```

.ENDM VGETI

.MACRO VSTORI I, INDX, VAL, J, IDIM

```

; SAVE REGISTERS
; GET NO OF ARGUMENTS
; WERE THERE 5 ARGUMENTS?
; YES: ASSEMBLE THESE INSTRUCTIONS
; NO: ASSEMBLE ONLY THIS INSTRUCTION
; ALWAYS ASSEMBLE FROM HERE ON
; PUSH ADDRESS WITHIN ARRAY OF DESTINATION ELEMENT
; PUSH VALUE TO BE STORED
; PUSH START ADDRESS (VIRTUAL) OF ARRAY
; ACTIVATE KT-11 AND DO STORE
; RESTORE REGISTERS

```

Installing the macros in a user-created macro library.

Instead of defining the above macros in every subroutine which uses them they can be installed in a macro library, the name of which must be supplied to the assembler.

- 1) Create an ascii file with extension .MAC which contains the macro definitions, eg FILE.MAC.
- 2)

```
.R LIBR
*USRMAC=FILE/M
* C
This creates the macro library USRMAC.MAC.
```

When assembling your macro subroutine use the following command :-

```
.R MACRO
*dev:filnam,dev:filnam = dev:filnam,dev:macro library name/M
* C
```

```

eg: .R MACRO
*MACSUB,MACSUB = MACSUB,USRMAC/M
* C

```

Don't forget to call the macros with an .MCALL statement in the macro source code :-

```

eg: .MCALL VGETI, VSTORI
.GLOBL $VSTOI, $VGETI

```

Preservation of KT-11 Registers on Context Switches

RT-11 automatically saves the values of the general purpose registers and processor status information whenever there is a switch between background and foreground. However, the KT-11 status register and PAR/PIR registers are not preserved. Unless arrangements are made to preserve the KT-11 status register and the PAR or PAR's used by the programs, erroneous memory access will occur.

Under RT-11, FORTRAN loads PAR 0 just before every extended memory read or write. If the background has set up PAR 0 and is interrupted by the foreground before it has executed the memory access, an erroneous fetch or store will occur when the background resumes because PAR 0 will contain the value used by the foreground.

Fortunately, DEC provides the .CNTXSW programmed request which permits specified locations in memory to be preserved on every context switch. To eliminate context switching problems, it is only necessary to call a short macro subroutine at the start of both background and foreground programs. The subroutine below which preserves PAR 0 and the KT-11 status register has proven satisfactory.

	.GLOBL	CNTXSW
	.MCALL	.CNTXSW
CNTXSW:	MOV	#LIST, RO
	.CNTXSW	,\$SWAPLS
	RTS	PC
SWAPLS:	.WORD	177640
	.WORD	177572
	.WORD	0
LIST:	.BYTE	0,33
	.WORD	0
	.END	

USER REQUESTS

I am having trouble with the FB monitor on an LSI 11-23. I can't get a foreground program to terminate normally & retain files it creates. Is there a solution to this problem?

David Rothbard
Geology Dept.
Univ. of Missouri
Columbia, Mo. 65211

PAST SYMPOSIUM INFORMATION

Enclosed please find a synopsis of the wish-list items we received at the 1981 European DECUS in Hamburg for submission to the mini-tasker. I thought it might make an interesting article. Our "official" responses are included.

Submissions from the RT-11 Development Group have been conspicuously absent from the pages of the newsletter for some time, a situation I'd like to see change.

See you in L.A.?

Regards,



Les Parent

Project Leader,
RT-11 Development

Wish list from European Symposia 1981

The following is a synopsis of the wishlist items presented at the 1981 European DECUS:

1. Logical disk subsetting.

DEC: Look for this in V5.0

2. Have a 3rd logical device (LB: - RSXish) which would contain MACRO, LINK... When a DCL command was used the third device would be scanned for the file.

DEC: No plans to implement any other "permanent" logical names

3. LookUp file on SY: if .LOOKUP fails on DK:

DEC: A popular wish, but no plans as yet.

4. Ability to handle complex relocation for REL files.

DEC: No plans to extend LINK's functionality in this area.

5. FPP-11 support.

11.

DEC: Already there! The FPP-11 is FPP compatible.

6. ROMable version of RT-11.

DEC: There is - it's called MRRT.

7. Better debusser than ODT/VDT. Example: Symbolic debusser. They really do not like the current debussers.

DEC: Possibly in V5.0, probably unsupported.

8. MRRT upgraded to use logical devices as planned in Version 5.

DEC: No plans to extend MRRT functionality beyond RT-11 V4.0

9. Have a programmed request that reads a directory only.

DEC: Unnecessary - lookup device non-filestructured and start reading at block 6!

10. SJ contain the same features as FB (ea. synchronous completion routines, console SET options, .DEVICE, .WAIT, etc.)

DEC: Looking at making SJ a conditional assembly of FB, but no plans yet - affects SJ users who are already "size" sensitive.

11. User specified alternative to CTRL/C

DEC: No plans.

12. .GTLINE programmed request.

DEC: It was looked at, but no plans at this time.

13. User defined DCL commands

DEC: Look for this in V5.0

14. UIC's

DEC: Never! RT-11 is NOT a multi-user system.

15. Protecting a file from being read.

DEC: Ditto!

12.

16. Ability to create and read multi-directories. Another UIC request.

DEC: Ditto, Ditto!!

17. Allow access of logical disks through passwords

DEC: Ditto, Ditto, Ditto!!!

18. SET DEVICE INOIWRITE

DEC: Available as Feature of Logical Disk Subsetting in V5.0

19. .SPCPS programmed request in SJ.

DEC: We're looking into it - possibly in V5.0

20. SET TT console options such as speed.

DEC: Again, we're looking into it, possibly for V5.0

21. CCL (with file specifications)

DEC: Once again, possibly in V5.0

22. Calendar clock support

DEC: Not likely, since we don't make such a device. Perhaps a "surprise" contribution to the RT-11 SIG Library!?

23. Make utilities virtual jobs.

DEC: No plans.

24. Virtual overlays in SJ.

DEC: No plans - use VM handler (which we plan to ship with V5.0)

25. Document a way to use the same CSR for LS and boot console

DEC: Messy stuff!

26. Way to display all registers in ODT/VDT.

DEC: No plans, but may ship an unsupported symbolic debugger with V5.0

27. Make sure that "USR escape" feature is not changed.

DEC: You're not supposed to know about that! (It won't)

28. Shared resident libraries.

DEC: We've looked long and hard at this, for a "clean" implementation (not a "handler" with a "directory") and have decided the tradeoffs aren't worth it.

30. FILES-11 support in FILEX.

DEC: No plans.

SYMPOSIUM TAPE COPY

If you plan to submit a program to the RT-11 SIG Tape at Los Angeles, please bring it on a single or double density floppy diskette.

DECUS LIBRARY

PLOTTING ON THE HP 9872S THE IEEE-488 BUS.

Further to my earlier letter regarding the handler PL.MAC for a 'listen only' device on the IEEE-488 bus I have now carried out some modifications to DECUS *11-427 which significantly improve the plotting facilities using the HP 9872S plotter.

These routines have been modified and extended to include:

1. Plotting via a disc file. The file may then be plotted via a spooler
2. Rotation of plots to fit A4 size plots across the roll paper.
3. Control of paper advance and the cutter.
4. Pen speed control.
5. A routine to allow symbols to be drawn at data points.

I have also patched QUEUE and QUEMAN to provide a second queue package which allows plot files to be kept separate from the main queue. This avoids plot files holding up the printing queue, and also includes the plotter (PL) and the PLT extension as the default device and file types. Apart from these changes, and a renamed.TMP file the package operates in exactly the same way as QUEUE and QUEMAN.

Although these modifications have greatly improved the plotting facilities available, the actual plotting software could still do with some improvements. I would be pleased to hear from anyone who would like details of the plotting facilities mentioned above, or who has available better software for the HP 9872S plotter.

I hope to get these modifications documented and submitted to DECUS in the near future.

Any correspondence should be sent to:-

J. Docherty
Ministry of Works and Development
Central Laboratories
P O Box 30845
LOWER HUTT
New Zealand.

New & Revised DECUS Library Submissions

11-475 (new) by David Ford, ANSCO Information System,
Ltd., Quebec, Canada.

ALGEB is a language designed to satisfy the needs that arise in doing computational Algebra and Number Theory. It is a block structured recursive language in the ALGOL-PASCAL family. ALGEB features matrix and vector manipulations and operations with integers of essentially unlimited magnitude. Floating point operations are not included.

TSXLIB

Like RT-11, TSX-Plus offers the MACRO programmer a number of system services via programmed requests or EMTs. RT-11 makes its system services available to the FORTRAN programmer through the system subroutine library, SYSLIB. TSXLIB makes the TSX-Plus EMTs available to the FORTRAN programmer as a library of callable routines. The package includes the MACRO source modules for all the routines, a user's manual, a cross reference chart, an indirect command file to build the library and the implemented library.

Earlier versions of TSXLIB appeared on the Fall 80 and Spring 81 DECUS Symposia RT-11 Tapes. TSXLIB now contains 49 routines supported under TSX-Plus V2.0. The current TSXLIB release, 81h04a, fixes some bugs. About half of the routines have been tested and the rest have been reviewed. The cross reference chart indicates which routines have been tested and which reviewed.

I am submitting the TSXLIB package to the DECUS Program Library and plan to put it on the Fall 81 DECUS Symposium RT-11 Tape. Anyone desiring a copy of this release sooner than mentioned above should contact me.

N. A. Bourgeois, Jr. 1738
Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185
(505) 844-8088

The enclosed contribution to the Mini-Tasker describes a collection of software tools that combine with the NEC Spinwriters 5510/20 and 5515/25, with the Technical Math / Times Roman thimble, to create an effective technical manuscript preparation facility for the RUNOFF user. Publication in the Mini-Tasker seems an ideal way to "get the word out" about the capabilities of this collection of tools. The package is being submitted to the DECUS library.

Sincerely,



Dan Dill
Professor of Chemistry

A System for Technical Manuscript Preparation

Dan Dill
Chemistry Department
Boston University
Boston, Massachusetts 02215
617/353-4277

Traditional preparation of technical manuscripts is generally a slow, tedious, error-prone process. Secretaries are not uniformly familiar with special notations, and anyhow are hampered by reading handwritten text. Equations are difficult to type and once typed difficult to alter. Finally, because of the effort involved, one hesitates to make changes once a manuscript has been typed. These problems are largely eliminated by using a word-processing system with technical document capabilities. The facilities contained in the UNIX system (NROFF and TROFF) are best examples.

Unfortunately, most of the writers in the world do not also have a UNIX terminal or similar capability close at hand. However, the pieces (both software and hardware) now exist so that a modest and inexpensive but quite effective system for the preparation of technical manuscripts can be put together on PDP 11s. These pieces are the following:

- o A VT52 or VT100 video terminal and a keypad editor such as K52 or KED. (Non-screen editors can be used, but are obviously much less friendly.)
- o A NEC Spinwriter 5510/20 or 5515/25 with the Technical Math / Times Roman (TM/TR) thimble.
- o The DECUS program RUNOFF, for document formatting.
- o A RUNOFF preprocessor, such as the program TEXT described below, to facilitate use of the super/subscripting, alternate, and composite character set features of the Spinwriter.
- o A macro preprocessor, such as MP, available as part of the DECUS C System, to allow local or system-wide definition of sequences of TEXT and RUNOFF commands as single commands, and to allow selective inclusion of text.

- o A collection of command files to automate the processing through MP, then TEXT, and finally RUNOFF, to give final output on the Spinwriter.

The introduction of KED/K52 as part of RT-11 Version 4 and (optionally) for RSX-11M Version 3.2 make text entry so easy that RUNOFF becomes a viable alternative to a secretary. The introduction of the Spinwriter with the TM/TR thimble means that manuscripts of considerable technical complexity

can be produced in a single pass. The only missing piece is a program such as TEXT to extend the full capabilities of the Spinwriter to the RUNOFF user.

The Spinwriter print element can be positioned in 1/120-inch horizontal increments and 1/48-inch vertical increments, and so can construct precise composite symbols. The NEC TM/TR thimble has 127 (!) symbols, including many of the commonly used greek letters and special graphics. These capabilities of the Spinwriter are invoked by various, often lengthy, sequences of control characters. TEXT was created, as a preprocessor for RUNOFF, to spare the RUNOFF user the need to generate these sequences manually. At the same time, RUNOFF was modified to pass these non-printing command sequences raw to the output without affecting RUNOFF justification and fill computations. The resulting TEXT/RUNOFF combination provides a considerable (output-device-specific) extension to the RUNOFF command set.

1.0 What TEXT does

The TEXT commands are

↑ = superscript following character (or characters inside {...}),
| = subscript following character(s),
\ = print alternate/composite character(s).

The basic idea of TEXT is illustrated by the TEXT command string

↑1\p↑-|u.

This generates a string which, when processed by RUNOFF, will cause the Spinwriter to print

1_{π_u}

There are bells and whistles, but this is the essential capability of TEXT.

2.0 Pulling out all of the stops

To illustrate TEXT/RUNOFF in action, there is on the next page a listing of a sample TEXT input file. (The example is taken from a manuscript by P. M. Dittman, Dan Dill, and J. L. Dehmer, entitled "Shape-resonance induced non-Franck-Condon effects in the valence-shell photoionization of O₂," that was prepared on the Spinwriter using TEXT/RUNOFF and submitted for publication to the Journal of Chemical Physics in March of 1981. The equations have been wrapped to successive lines to fit on the page. Tab characters are indicated by <TAB> and _ escapes TEXT commands. The resulting RUNOFF output on the Spinwriter is given on the following page.

Study of the *.TXT file and the resulting Spinwriter output illustrate many of the features of the TEXT/RUNOFF combination.

2.1 Example TEXT input file *.TXT. Output is RUNOFF input (*.RNO)

```
.nhd;ps 58,255;lm 5;.rm 80;ts 14 77;.f;.j;.sp 1
Treatment of vibrational motion within the context of the adiabatic nuclei
approximation24 is accomplished as follows: In place of the fixed-R
dipole amplitudes given in Eq. (36) of Ref. 19,25 we define the
vibrational-state-dependent amplitudes
.tp 4;.rm 255;.nf;.nj;.b 2
<TAB>D+{↑((-)v|fv|i)}|{Lm|{g}} = <v|f_|D+{↑((-)|{Lm|{g}})}_|v|i>
= \IdR X|f(R) D+{↑((-)|{Lm|{g}})} X|i(R) .<TAB>(1)
.b 2;.f;.j;.rm 80
The vibrationally resolved integrated cross section is then computed
as↑{↑{19}}
.tp 4;.rm 255;.nf;.nj;.b 2
<TAB>\s+{↑{v|fv|i}} = (4\p+{↑{2}/3})\ah\w \S|{↑{↑{Lm|{g}}}}
_|<v|f_|D+{↑((-)|{Lm|{g}})}_|v|i>_↑{↑{2}} .<TAB>(2)
.b 2;.f;.j;.rm 80
where h\w is the photon energy, and the vibrationally unresolved
integrated cross section is given by
.tp 4;.rm 255;.nf;.nj;.b 2
<TAB><\s> = \S|{↑{↑{v|f}}}\s+{↑{v|fv|i}} = <v|i_|\s(R)_|v|i> .<TAB>(3)
.b 2;.j;.f;.rm 80
where \s(R) is the fixed-R integrated cross section given by Eq. (41) of
Ref. 19.26 To obtain the vibrationally unresolved asymmetry
parameter, we begin with the general expression for the vibrationally
resolved differential cross section, which, in accordance with Yang's
theorem,27 is given by
.tp 4;.rm 255;.nf;.nj;.b 2
<TAB>d\s+{↑{v|fv|i}}/d\W = (\s+{↑{v|fv|i}})/4\p
[1 + \b+{↑{v|fv|i}}] P|{↑{2}}(\cos \T) .<TAB>(4)
.b 2;.f;.j;.rm 80
Here, P|{↑{2}}(\cos \T) is the second Legendre polynomial, \T is the ejection
angle relative to the polarization direction of the light, and \b is
obtained, in analogy with Eq. (2), by substituting Eq. (1) for the fixed-R
dipole amplitudes in Eq. (40) of Ref. 19.25 Finally, using the
expression
.tp 4;.rm 255;.nf;.nj;.b 2
<TAB><d\s/d\W> = (<\s>/4\p) [1 + <\b>] P|{↑{2}}(\cos \T)
= \S|{↑{↑{v|f}}}\d\s+{↑{v|fv|i}}/d\W<TAB>(5)
.b 2;.f;.j;.rm 80
for the vibrationally unresolved differential cross section, we can
express
the vibrationally unresolved asymmetry parameter as
.tp 4;.rm 255;.nf;.nj;.b 2
<TAB><\b> = \S|{↑{↑{v|f}}}\s+{↑{v|fv|i}} \b+{↑{v|fv|i}} / <\s>
= <v|i_|\s(R)\b(R)_|v|i> / <v|i_|\s(R)_|v|i><TAB>(6)
.b 2;.f;.j;.rm 80
where \b(R) is the fixed-R asymmetry parameter obtained from Eq. (40) of
Ref. 19 using the fixed-R dipole amplitudes, Eq. (36) of Ref. 19.
```

2.2 Spinwriter RUNOFF output file from file *.TXT

Treatment of vibrational motion within the context of the adiabatic nuclei approximation²⁴ is accomplished as follows: In place of the fixed-R dipole amplitudes given in Eq. (36) of Ref. 19,²⁵ we define the vibrational-state-dependent amplitudes

$$D_{Lm_f}^{(-)vf_i} = \langle v_f | D_{Lm_f}^{(-)} | v_i \rangle = \int dR X_f(R) D_{Lm_f}^{(-)} X_i(R) . \quad (1)$$

The vibrationally resolved integrated cross section is then computed as¹⁹

$$\sigma^{vf_i} = (4\pi^2/3) \alpha \hbar \omega \sum_{Lm_f} |\langle v_f | D_{Lm_f}^{(-)} | v_i \rangle|^2 , \quad (2)$$

where $\hbar \omega$ is the photon energy, and the vibrationally unresolved integrated cross section is given by

$$\langle \sigma \rangle = \sum_{v_f} \sigma^{vf_i} = \langle v_i | \sigma(R) | v_i \rangle , \quad (3)$$

where $\sigma(R)$ is the fixed- R integrated cross section given by Eq. (41) of Ref. 19.²⁴ To obtain the vibrationally unresolved asymmetry parameter, we begin with the general expression for the vibrationally resolved differential cross section, which, in accordance with Yang's theorem,²⁶ is given by

$$d\sigma^{vf_i}/d\Omega = (\sigma^{vf_i}/4\pi) [1 + \beta^{vf_i} P_2(\cos \Theta)] . \quad (4)$$

Here, $P_2(\cos \Theta)$ is the second Legendre polynomial, Θ is the ejection angle relative to the polarization direction of the light, and β is obtained, in analogy with Eq. (2), by substituting Eq. (1) for the fixed- R dipole amplitudes in Eq. (40) of Ref. 19.²⁵ Finally, using the expression

$$\langle d\sigma/d\Omega \rangle = (\langle \sigma \rangle/4\pi) [1 + \langle \beta \rangle P_2(\cos \Theta)] = \sum_{v_f} d\sigma^{vf_i}/d\Omega \quad (5)$$

for the vibrationally unresolved differential cross section, we can express the vibrationally unresolved asymmetry parameter as

$$\langle \beta \rangle = \sum_{v_f} \sigma^{vf_i} \beta^{vf_i} / \langle \sigma \rangle = \langle v_i | \sigma(R) \beta(R) | v_i \rangle / \langle v_i | \sigma(R) | v_i \rangle \quad (6)$$

where $\beta(R)$ is the fixed- R asymmetry parameter obtained from Eq. (40) of Ref. 19 using the fixed- R dipole amplitudes, Eq. (36) of Ref. 19.

3.0 Summary of symbols

The collection of all of the TEXT symbols and the currently defined composite characters available on the TM/TR font is given below. It is easy to add new composite symbols.

Greek characters

\a = α
 \b = β
 \D = Δ
 \d = δ
 \e = ϵ
 \g = γ
 \h = η
 \l = λ
 \m = μ

Special symbols

\f = f
 \i = i
 \j = j
 \k = k
 \l = l
 \m = m
 \n = n
 \o = o
 \p = p
 \r = r
 \s = s
 \t = t
 \u = u
 \v = v
 \w = w
 \x = x
 \y = y
 \z = z

Indexicals

\0 = 0
 \1 = 1
 \2 = 2
 \3 = 3
 \4 = 4
 \5 = 5
 \6 = 6
 \7 = 7
 \8 = 8
 \9 = 9

Overstruck (composite) symbols

e\' = \acute{e}
 e\" = \grave{e}
 e\l = ℓ
 e\~ = \tilde{e}
 e\Y = \mathcal{E}
 A\% = \bar{A}
 O\ / = ϕ
 A\A = \hat{A}
 E\U = \bar{E}
 e\V = \mathcal{E}

Composite symbols

\+ = \pm
 \- = \mp
 \< = \leq
 \> = \geq
 \= = $=$
 \Z = \mathbb{Z}
 \X = \rightarrow
 A\|B = $A\|B$

Large composite symbols

\I = \int
 \S = \sum
 \[= $[$
 \] = $]$

4.0 OK. Where can I get one?

TEXT is written in MACRO 11, to run on PDP 11 computers under the operating systems RT-11 (Version 4) and RSX-11M (Version 3.2). There is a detailed manual describing the obvious and the less obvious features of TEXT, its operation, and examples of automated processing using the RT-11 command MUNG (and TECO) or using the RSX-11M indirect command processor task ...AT.. The package will be submitted to the DECUS PDP 11 library. For those in a hurry, it is available in the mean time on RX01/RX02 floppies from TEXT's author, me, (Al Franken ...) Dan Dill, Chemistry Department, Boston University, Boston, Massachusetts 02215, USA, telephone 617/353-4277. The modified RUNOFF will also be provided.

5.0 And so Virginia ...

We have been using this collection of tools for about seven months. Together they provide a critical mass that has markedly transformed into something almost pleasant the previously most unpleasant task of technical manuscript preparation. The cost above that of a typical RT-11 or RSX-11M installation is the price of a Spinwriter, about \$2,500. In view of the return, this seems a small additional price to pay, indeed.

SPR'S

PROBLEM:

FORTTRAN IV used standalone on RT-11 with SIMRT linked into FORLIB, will not accept lower case characters from the key board with the proper bit set in the JSW.

RESPONSE:

Thank you for bringing this problem to our attention. As you have noticed, the JSW should have the offset value of 44 rather than 54. Please make the source change to that line (JSW = 44), then macro the module as follows:

```
.R MACRO
*SIMRT=FRT,SIMRT
*~C
```

Rebuild the FORLIB with SIMRT installed. Back up SIMRT prior to making this source change.

PUBLICATION:

It is not our intention to publish this SPR.



UNIVERSITY OF VICTORIA

P.O. BOX 1700, VICTORIA, BRITISH COLUMBIA, CANADA V8W 2Y2
TELEPHONE (604) 477-6911, TELEX 049-7222

Computing and Systems Services

Users of MU-BASIC/RT-11 Version 2.00S may be interested in several bugs we have uncovered.

Several of these involve the internal counters which are supposed to keep track of the current number of open channels for each user. Under certain circumstances a counter may be decremented without having been previously incremented. The lack of balance between increments and decrements causes the counter to go negative, and this condition goes undetected until the byte value rolls over to a positive number of 177 octal. At this point the user gets a TOO MANY CHANNELS error, even though he may have no channels open at the time. This condition persists for that user partition even after the user logs off and another logs in, and can only be cleared by exiting MU-BASIC and re-loading it using the RT-11 RUN command.

Another problem arises when a user attempts to specify a two-digit unit number for a file I/O operation. Although MU-BASIC allows terminals to be specified in file I/O operations, an ILLEGAL FILE SPECIFICATION error results if more than one digit is specified for the unit number, thereby restricting these operations to ten terminals.

The enclosed patches seem to fix all the bugs encountered so far. They may be implemented using the following commands, and then relinking:

```
MACRO PAS1AA
MACRO PAS2EA
MACRO PAS2EB
```

Sincerely,

R. J. Tapp

R. J. Tapp
Academic Systems

```
RUN PAT
*MUBS1=MUBS1,PAS1AA
```

```
RUN PAT
*MUBS1X=MUBS1X,PAS1AA
```

```
RUN PAT
*MUBS2E=MUBS2E,PAS2EA
```

```
RUN PAT
*MUBS2E=MUBS2E,PAS2EB
```

: PAS1AA.MAC - PATCH FOR CHANNEL COUNT PROBLEMS IN MUBS1.OBJ

```
.TITLE MUBS1
.PSECT MUBS1
.ENABL GBL
```

```
==.+144
CALL PAS1A1
==.+606
CALL PAS1A2
==.+552
CALL PAS1A3
==.+336
JMP CLOSX1+4
==.+240
CLOSC6:
==.+20
CLOS7B:
==.+140
CLOSX1:
==.+1600
PURGE:
```

```
.PSECT PAS1AA
PAS1A1: INCB CNOC(R5)
CALL CLOS7B
RETURN
PAS1A2: INCB CNOC(R5)
CALL CLOSC6
RETURN
PAS1A3: INCB CNOC(R5)
CALL PURGE
RETURN
```

.END

; PAS2EA.MAC - PATCH FOR CHANNEL COUNT PROBLEMS IN MUBS2E.OBJ

```

        .TITLE  MUBS2E
        .PSECT  MUBS2E
        .ENABL  GBL

.=.+2532
OPNF1:  JMP      PAS2E1
.=.+1316
OPNF14: CALL      PAS2E2

        .PSECT  PAS2EA
PAS2E1: TSTB     CNOC(R5)
        BGE     1$
        JMP     OPNF1+76
1$:     MOV      #100000,R0
        JMP     OPNF1+4
PAS2E2: INCB     CNOC(R5)
        CALL    CLOSC7
        RETURN
        .END

```

; PAS2EB.MAC - PATCH FOR UNIT SPECIFICATION PROBLEM IN MUBS2E.OBJ

```

        .TITLE  MUBS2E
        .PSECT  MUBS2E
        .ENABL  GBL

.=.+174
MOV      SYASUNQ,2(SP)
.=.+54
FNDDEV:  TSTB     10(SP)
.=.+6
MOV      6(SP),2(SP)
.=.+1160
CALL     PAS2E3

        .PSECT  PAS2EB
PAS2E3:  SUB      #'0,R2          ; CONVERT FIRST DIGIT TO BINARY
        MOV      R2,-(SP)
        MOVB     3+6(SP),R2      ; GET SECOND DIGIT
        BEQ      1$              ; SKIP IF NOT THERE
        SUB      #'0,R2          ; CONVERT TO BINARY
        ASL      (SP)            ; MULT. FIRST DIGIT BY 2
        ADD      (SP),R2         ; ADD TO SECOND
        ASL      (SP)            ; AND AGAIN BY 4
        ASL      (SP)
        ADD      R2,(SP)         ; 2*A + 4*(2*A) = 2*A + 8*A = 10*A
1$:      MOV      (SP)+,R2
        RETURN

        .END

```

Ms. Sheila Hatchell, Supervisor
 SPR Administration
 BOX F
 Maynard, MA 01754

Re: SPR 11-39276

Dear Ms. Hatchell:

Thank you for responding to our SPR with a "fix". We have not tried it yet, but hope that it works; the problem has been quite a bother to us.

Could you please explain why this (and apparently other) patches are not published? I doubt very much that we are the only users opening and closing more than one file, so a number of people are going to waste a lot of time on this, even though a patch exists to fix the problem. The obvious question is "How many other bugs have been fixed and the general user community hasn't been told?" Perhaps I have misinterpreted the Publication note; the "fix" you sent does look like a software dispatch item.

Again, thank you for your assistance.

Best regards,

OSTERGAARD ASSOCIATES

Norman R. Dotti, PE

SPR NO. 11 - 39276

SOFTWARE:	SYSTEM	VERSION	PRODUCT	VERSION	COMPONENT
	RT-11	4.0	FORTRAN IV	V2.5	OTS

PROBLEM:

Fatal memory error occurs when closing more than one opened unit.

RESPONSE:

Thank you for bringing this problem to our attention. Enclosed is the necessary patch to alleviate this problem.

PUBLICATION:

It is not our intention to publish this SPR.

CORRECTION FOR UNIT CLOSING (PAT17)

PROBLEM:

The FORTRAN OTS does not correctly close a unit when more than one unit has been closed.

SOLUTION:

1. Type in the following MACRO file: PAT17.MAC

PAT17.MAC:

```
.TITLE $EOL
.IDENT /007/
.PSECT OTS$I
```

S=.

.=S+114

JMP PATIOX

RET:

.=S+262

```
PATIOX: ADD    #4,R0
MOV      -(R0),R4
MOV      -(R0),(SP)
SUB      #4,R0
JMP      RET
.END
```

2. Assemble the patch using MACRO-11

```
.R MACRO
*PAT17=PAT17
**C
```

3. Install the patch, using PAT, to the most recently patched OTSCOM.OBJ file:

NOTE: Make a copy of OTSCOM.OBJ before you patch it just in case something goes wrong.

```
.R PAT.SAV
*OTSCOM=OTSCOM/C:61441,PAT17/C:11647
```

4. Rebuild the OTS using the procedure described in the FORTRAN IV Installation Guide.

5. Test the patch by creating and compiling the following FORTRAN program.

NOTE: You must create TST.DAT and TST1.DAT in your work area before running the following FORTRAN program.

```
OPEN(UNIT=3,NAME='TST.DAT',TYPE='OLD')
OPEN(UNIT=2,NAME='TST1.DAT',TYPE='OLD')
WRITE(2,*) 'HELLO'
WRITE(3,*) 'BYE'
CLOSE(UNIT=3)
CLOSE(UNIT=2)
STOP
END
```

Which should execute without error when the patch has been successfully installed.



DIGITAL EQUIPMENT COMPUTER USERS SOCIETY
ONE IRON WAY, MR2-3/E55
MARLBORO, MASSACHUSETTS 01752
Forwarding and Return Postage Guaranteed

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, MR2-3
Marlboro, Massachusetts 01752 USA

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