THE mini-tasker

DECUS
RT-11 SIG NEWSLETTER

AUGUST 1983 VOLUME 9-3

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

This is the first

Subscription Service issue

of the

RT-11 Mini-Tasker

Printed in U.S.A.

THE mini-tasker

DECUS
RT-11 SIG NEWSLETTER

AUGUST 1983 VOLUME 9 – 3

TABLE OF CONTENTS

FROM THE EDITOR

I am looking for 10 volunteers to convert the audio tapes recorded at the St. Louis Symposium RT-11 sessions into articles that will be printed in future issues of the "Minitasker". Each volunteer will only be responsible for converting the tape from one session. Hopefully, this will be an ongoing process after each symposium. If you are interested, please call me as soon as possible.

I had hoped to get this newsletter to the Decus office before July 1, 1983. However, I was not able to gather all the articles concerning St. Louis in time. Therefore, only those of you that have already paid the subscription fee will be receiving this issue. I have included a copy of the subscription form in this newsletter. If you know anyone that has not subscribed yet, please pass the form on to them.

Thankyou,

Ken Demers

HOW TO REACH THEM

JOHN T. RASTED JTR Associates 58 Rasted Lane Meriden, CT 06450 (203) 634-1632	RT-11 SIG Chairman
KEN DEMERS MS-44 United Tech. Research Ctr. Silver Lane East Hartford, CT 06108 (203) 727-7139 or 7240	Newsletter Editor DECNET Contact
RALSTON BARNARD Division 2565A Sandia Laboratories Albuquerque, NM 87185 (505) 844-5115	RT-11 Handout Editor
MARK BARTELT HSC Research Development Corp. 555 University Ave. Toronto, Ontario Canada M5G 1X8 (416) 597-1500 Ext 4588 or 598-5982	Structured Languages Contact
DOUG BOHRER First National Bank of Chicago One First National Plaza Chicago, IL 60670 (312) 732-8785	APL Contact

NICK BOURGEOIS / 9238
Sandia Laboratories
P.O.Box 5800
Albuquerque, NM 87185
(505) 844-8088

TSX Contact
MACRO Contact

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

TECO Contact
Product Planning
Contact

JAMES R. CUTLER
Software Results Corp.
1229 West Third Ave.
Columbus, OH 43212
(614) 421-2094

Standards Coordinator

JOSEPH LACHMAN
Lachman Associates, Inc.
645 Blackhawk Drive
Westmont, IL 60559
(312) 986-8840
(303) 451-3872

Tape Copy Coordinator

BILL LEROY
Information Resources, Inc.
470 E. Paces Ferry Rd. N.E.
P.O. Box 52661
Atlanta, GA 30355
(404) 231-3117

COBOL Contact
Wish List Contact

CARL LOWENSTEIN
Univ. of Calif. San Diego
Marine Physical Lab. of the
Scripps Inst. of Oceanography
San Diego, CA 92152
(619) 294-3678

RT-11 Hardware Contact

JACK PETERSON
Horizon Data Systems
1901 Wildflower Terrace
Richmond, VA 23233
(804) 740-9244

C Contact

SUSAN S. RASTED
Software Dynamics Inc.
1000 Yale Ave.
Wallingford, CT 06492
(203) 265-2226

FMS-11 Contact

NED RHODES
E-Systems
Melpar Division
7700 Arlington Boulevard
Falls Church, VA 22046
(703) 560-5000

Symposia Coordinator

TOM SHINAL General Scientific Corp. 1684 East Gude Drive Rockville, MD 20850 (301) 340-2773	RT DECUS Library Contact
ED STEVENS E M D A Inc. 111 South Hudson St. #8 Pasadena, CA 91101 (213) 795-5991	BASIC Contact
J. W. TIPPIE Kinetic Systems Inc. 11 Mary Knoll Drive Lockport, IL 60441 (815) 838-0005	CAMAC Contact
RON TRELLUE Division 7523 Sandia Laboratories Albuquerque, NM 87185 (505) 844-0955	FORTRAN Contact RT-11 LUG Contact

WHY A SUBSCRIPTION SERVICE ?

As a current member of DECUS, you have already experienced the benefits derived from the information shared through the Special Interest Group Newsletters.

It is necessary to begin a subscription service in order to aid the volunteer editors in publishing more timely and sophisticated newsletters and to keep these newsletters commercial-free. In the ever changing environment of the computer industry, newsletters such as these technical publications, written and edited by actual Digital equipment users, become more and more valuable and in fact necessary.

WHY JOIN NOW ?

By signing up for the subscription service now, you will be assured of continuing to receive the newsletters of your choice. You won't miss any information.

As in any new venture, the start-up and initial subscriptions are very important. By signing up now, you will be showing your support for this invaluable service supplied by your fellow users.

If you DON'T sign up NOW but choose to wait . . . you will STOP receiving the SIG newsletters as of JULY 1, 1983. After July 1, subscriptions will run on a twelve month basis beginning the month you do subscribe.

PARTICIPATE

Remember YOU, as a Digital equipment user, are a vital part in the continued excellence of the SIG Newsletters. Please continue or start now supplying articles and problem-solving answers to the newsletters of your choice.

DON'T DELAY – USE THE ATTACHED FORM AND TAKE ADVANTAGE OF THE NEW SERVICE TODAY! ! !

This form is for use by U.S. chapter members only.

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
- No purchase orders accepted

Check # ______
Bank # ______
Amount $ ______

Name ______ (First) ______ (Last) ______ DECUS Membership No. ______

Company/Affiliation ______

Mailing Address ______ Mail Stop ______

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

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/PC 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 $______

Please charge my order to the following:

☐ Mastercard ______ Exp. date ______

☐ Visa ______ Exp. date ______

I understand that neither DECUS nor Digital Equipment Corporation is responsible for any publication not published by a Special Interest 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. ______

3/83

Summer Institute of Linguistics

Affiliated With Min. of Education & Culture and University of the Philippines
P.O. Box 2270 MANILA — TEL. 78-00-61
Cable Via RCA, Linguista Manila
Telex - 27772 SIL PH

DAVID H. OHLSON
Director

In our computer installation here in the Philippines we use one DRV11 parallel port to service several I/O devices (two different printers and another computer). Three separate handlers have been written to interface these devices to the RT11 operating system.

When RT11 loads a handler using the LOAD command or the .FETCH programmed request, after the handler is loaded into memory, the interrupt vector(s) are set to the current location of the interrupt service routine within the handler. When the handler is removed from memory with the UNLOAD command or the .RELEAS programmed request, RT11 resets the interrupt vector(s) to zero.

This approach works fine as long as one handler uses each hardware port. However, problems develop when several handlers try to use the same hardware port, and one or more of the handlers are loaded into memory with the LOAD command. If a second handler is used while another one is LOADed, the vector locations are overwritten by the second handler. Thus, if an attempt is made to use the first handler, the system crashes. RT11 will not reload the first handler even though requested with a LOAD command until the handler is UNLOADed. The problem becomes even more crucial when running TSX-Plus since it permanently loads all handlers.

I recently discovered a simple way to solve this problem. The solution is to have the handler initiation section reset the vector locations each time the handler is entered with a .READ or .WRITE request. This can be accomplished with the following code (assuming two vectors are used and all handlers use the same priority):

```
MOV     PC,R4               ; Point R4 to vector table
ADD     #dd$VTB-.,R4        ;   ("dd" is the handler name)
MOV     (R4)+,R5            ; Point R5 to first vector
ADD     (R4),R4             ; Point R4 to interrupt service

MOV     R4,(R5)+            ; Set first vector
TST     (R5)+               ; Skip over priority
MOV     PC,R4               ; Point R4 to second vector
ADD     #dd$VTB+8.-.,R4     ;   table entry
ADD     (R4),R4             ; Point R4 to interrupt service
MOV     R4,(R5)             ; Set second vector
```

With this routine installed in each handler, any number of handlers can be loaded at the same time and share a single hardware port.

Sincerely yours,

Kenneth Zook
Manager of Computer Services

DSIR

DEPARTMENT OF SCIENTIFIC AND INDUSTRIAL RESEARCH

APPLIED MATHEMATICS DIVISION

P.O.Box 1335 Wellington New Zealand Telephone(04) 727 855 Telex 3276 RESEARCH

I submit the following for MINITASKER if you think it appropriate.

I have prepared a document which presents an exhaustive list of those references to device handlers and interrupts which are contained in the DEC documentation available for the RT-11 V4.0 operating system. It is known as:

SCIN 123 'A Documentation Directory for RT-11 Device Handlers and Interrupts'

and is available from me as below (SCIN stands for Scientific Computing Information Note).

The intention is to save handler writers from having to search through the documentation for an obscure reference to some aspect of handlers or interrupts which they know they have seen but cannot remember where.

Ray Brownrigg
Applied Mathematics Division
D. S. I. R.
P. O. Box 1335
Wellington
New Zealand

In an effort to build better lines of communication with the developers of RMS, I am trying to start an RMS wish list. It would be under the auspices of the Data Management Systems Sig because that seems to me to be the most logical one; however, I would like to think of it as a cross-sig endeavor. If members of your SIG use RMS, I would appreciate it if you would print this letter in your newsletter.

Too many sig wishes are responded to with the answer, "Well, that's really an RMS problem.", and from there, the wish disappears. I'm sure that that is as frustrating to the developers of RMS as it is to the users.

A session is planned for Las Vegas to discuss existing RMS wishes, and to accept new ideas and wishes. If any member of any sig would like to send me input, my address is below.

Thank you.

Chuck Evans
Director of Data Processing
Times Publishing Company
Times Square
Erie, Pennsylvania 16534

SYTEK, INC.
19 CHURCH ST., P.O. BOX 128 / BEREA, OHIO 44017
Phone (216) 243-1613

I would be interested to know if anyone is using RT-11/CTS-300 to receive data files (and write them to disk) from an electronic mail service. I have a requirement to retrieve information from CompuServ's InfoPlex electronic Mail Service using CTS-300.

Sincerely yours,

Michael Bibler
140580

USER RESPONSES

SIRA LTD · SOUTH HILL · CHISLEHURST · KENT · BR7 5EH · ENGLAND
Telephone 01-467 2636 Telex 896649

Sira
RESEARCH & DEVELOPMENT DIVISION
DIRECTOR: SIMON COLE MSc MIEE

I was interested to read your letter in the latest "Mini-Tasker" requesting help in emulating RT11 under RSX V3.2 .

Some time ago, long before RTEM was available, I had a similar problem in developing some RT11 software without inconveniencing our users by shutting down RSX.

My solution was to write an EMT instruction handler which can be built into an RSX task, and which supports most of RT11's system calls of the EMT 340, 374, and 375 variety. Since RT11 and RSX object files (but not libraries) appear to have the same format, it is often possible to build tasks even if you do not have the sources, although some minor modifications to some RT11 functions are required to avoid RSX and RT11 system conflicts (e.g. use of low task memory, and EMT 376/377 calls). Source modifications, or patches, may be required for these. The EMT handler also includes an FIS instruction emulator, using the FP-11, which could easily be modified to use software instead.

I am not familiar with MINITAB, and this solution may not be appropriate to your problem.

Also, I should emphasise that my simulator is not complete or fully debugged - I only implemented/tested those functions I needed, or were easy to do. However, it does cover a reasonable proportion of the RT11 (V3B, XM monitor) system facilities (file I/O and CSI handling all work) and you might find it a useful starting point if you receive no better offers.

I shall be pleased to send you the sources free of change if you supply me with a blank floppy disk, DECtape II cartridge, or magtape (I can only write at 800bpi on reels up to 18cm diameter). Alternatively, I could send you paper tape if preferred.

Yours sincerely,

Chris Doran

Chris Doran,
Computer Manager.

AVM SYSTEMS incorporated

5050 Mark IV Parkway
Fort Worth, Texas 76106
Telephone 817-625-2811
Telex 910-893-5087 AVM SYS FTW

Alfred K. Blackadar offered program DATCMD in the March Minitasker. I find it necessary to terminate the .PRINT programmed request output buffer with a <000> or <200> byte to prevent trash from being printed. I changed

"MSGB: .ASCII <15> <12>" to "MSGB: .ASCII <000>"

Also, I "ASSIGN DY1 DK," so the system diskette, on which DATCMD resides, is not "DK." Therefore, I change

"FILE: .RAD50 /DK DATCMDSAV/" to "FILE: .RAD50 /SY DATCMDSAV/"

Sincerely,

Mark Luckstead
Software Programmer
AVM SYSTMS INC.

TSX

TSXLIB -- A FORTRAN CALLABLE IMPLEMENTATION OF THE TSX-PLUS EMTS

N. A. Bourgeois, Jr.
Sandia National Laboratories
Albuquerque, NM

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. TSX-Plus also honors most RT-11 programmed requests. TSXLIB makes the TSX-Plus EMTs available to the FORTRAN programmer as a library of callable routines. The code in TSXLIB is all reentrant.

INTRODUCTION

The TSX-Plus library routines provide facilities to support detached jobs, file structured device mounting and dismounting, communication between running programs, program performance analysis, real time program execution, shared run time system, shared files, system status information, communication between running programs and a terminal, program control of the terminal, ODT activation mode, and several miscellaneous EMTs. The system status information routines require the use of TSX-Plus Version 2.2 [3]. Several of the real time support routines require TSX-Plus Version 3.0 [5]. All other routines require TSX-Plus Version 2.0 or later.

TSXLIB [1] is available from the DECUS/US Chapter Program Library. The kit includes the MACRO source modules for all groups of routines, a complete user's manual, a cross reference chart, an indirect command file to build the library, and the implemented library. The cross reference chart lists for each routine the page number in the user's manual and the page number in the "TSX-Plus Reference Manual" [2].

The standard FORTRAN subroutine calling sequence shown below may be used to access all of the routines in TSXLIB.

CALL SUBRTN (ARG1, ... ,ARGn)

Those routines that return only one value are also callable as FORTRAN functions. This is as follows:

IRET = RTNAME (ARG1, ... ,ARGn)

The "TSX-Plus Reference Manual" describes how the EMTs implemented in TSXLIB are accessed from a MACRO program. However, the FORTRAN/MACRO interface described in the "RT-11 Programmer's Reference Manual" [4] may also be used to access the routines in the library. The TSXLIB (as well as SYSLIB) routines are also available to the "C" programmer via the DECUS "C" [6] library function, "call(...)".

DETACHED JOB SUPPORT

Table 1 lists the routines that provide detached job support from within an executing program.

ISTDJ	Get the status of a detached job.
KLDTJB	Kill a detached job.
STDTJB	Start a detached job.

Detached Job Support.
Table 1.

DEVICE MOUNTING AND DISMOUNTING

It is possible to mount and dismount a file structured device for directory caching from within a running program. The routines listed in Table 2 provide these capabilities.

DISMNT	Dismount a file structured device.
MOUNT	Mount a file structured device.

Device Mounting and Dismounting.
Table 2.

INTERPROGRAM MESSAGE COMMUNICATION

TSX-Plus provides an optional facility that allows running programs to communicate with each other. Table 3 lists the routines that support this interprogram message communication.

MSGSND	Send a message to another job.
RCVMSG	Try to receive a message from another job.
RCVMSW	Wait for a message from another job.

Interprogram Message Communication.
Table 3.

Messages are transferred between programs by using named message channels. A message channel accepts a message from a sending program, stores the message in a queue associated with the channel, and delivers the message to a receiving program on its request for a message on the channel. Message channels are separate from I/O channels.

Each active message channel has associated with it an ASCII character name that is used by both the sending and receiving programs to identify the channel. The names associated with the channels are defined dynamically by the running programs. A message channel is active when messages are being held in the queue associated with the channel or if a program is waiting for a message from the channel. When message channels become inactive they are returned to a free pool and may then be reused.

Once a message is queued on a channel, that message will remain in the queue until some program receives it. A program's exiting to the keyboard monitor does not remove any pending messages. This allows one program to leave a message for another program that will be run at a later time.

PERFORMANCE ANALYSIS SUPPORT

For many applications the keyboard monitor level performance analysis control provided by TSX-Plus is adequate. However, in cases such as analyzing the performance of an overlayed program it is necessary to have control over the performance analysis feature from the running program. The routines listed in Table 4 provide just this capability.

INITPA	Initialize for a performance analysis.
ISPPA	Stop a performance analysis.
ISTPA	Start a performance analysis.
TERMPA	Terminate from a performance analysis.

Performance Analysis Support.
Table 4.

REAL TIME PROGRAM SUPPORT

The real time program support provided by TSX-Plus allows multiple real time programs to be run concurrently with normal time sharing operations. The basic functions provided by this facility are listed in Table 5.

A program must have operator privilege to use any of the real time routines. The real time facilities are available to both normal jobs controlled by time sharing lines and to detached jobs. Detached jobs started by time sharing users have operator privilege only if the user starting them does.

A basic facility required by many real time programs is the ability to access the I/O page which contains the peripheral device registers. A normal time sharing job does not have this access. It is instead mapped to a simulated RMON. This allows programs that directly access offsets into RMON to run correctly.

CNVAPA	Convert a virtual address to a physical address.
IBICIO	Bit clear a value into the I/O page.
IBISIO	Bit set a value into the I/O page.
ICNINT	Connect an interrupt vector to a completion routine.
IPEKIO	Peek at a value in the I/O page.
IPOKIO	Poke a value into the I/O page.
IRLINT	Release an interrupt vector connection.
IUNLKM	Unlock a job from memory.
LKANMY	Lock a job into any memory.
LKLOMY	Lock a job into low memory.
MPIOPS	Map the I/O page into the program space.
MPRMPS	Map the simulated RMON into the program space.
TKCTL	Take exclusive control of the system.
RLCTL	Relinquish exclusive control of the system.
STPRLV	Set the user mode processor priority level.

Real Time Program Support.
Table 5.

A real time program can access the I/O page in one of two ways: it can map the I/O page into the program's space; or it can leave the simulated RMON mapped into the program's space and perform peek, poke, bit set, and bit clear operations into the I/O page. It is much more efficient to directly access the device registers by mapping the I/O page into the program's space than to use the routines to perform each individual access. However, this technique will not work if the program must also directly access offsets into RMON. The correct way to access offsets into RMON is with the SYSLIB routine, ISPY, which will work even if the I/O page is mapped into the program's space.

The TSX-Plus real time support facility allows a program to connect a real time interrupt to a completion routine. If this is done, the completion routine is executed each time the specified interrupt occurs. It is possible for several interrupt vectors to be connected to the same completion routine in a job but it is illegal for more than one job to connect to the same interrupt vector.

An execution priority may be specified for each completion routine. This is not the same as the hardware selected priority of the interrupt. All completion routines are synchronized with the job and run at hardware priority level zero. The completion routine priority is used to schedule the completion routines for execution. The available priority levels are zero through seven. The execution of a real time completion routine for one job will be interrupted and suspended if an interrupt occurs that causes a higher priority completion routine for another job to be queued for execution. However, a completion routine for a given job will never be interrupted to run another completion routine for the same job even if a higher priority completion routine is pending.

Completion routine priorities one through seven are real time priorities. They are higher than the priorities given to time sharing jobs and will always preempt the time sharing jobs. Completion routine priority zero is not a real time priority but rather a very high normal priority. Such zero priority completion routines are time sliced in the normal fashion. If a completion routine enters a wait state it relinquishes its real time priority. Jobs that have real time, interrupt driven completion routines need not be locked in memory.

In time critical, real time applications where a program must respond to an interrupt with minimum delay, it may be necessary for the job to lock itself in memory to avoid the time consumed in program swapping. This facility should be used with caution since if a number of large programs are locked in memory there may not be enough space left to run other programs.

A running program may gain exclusive access to the system to perform some time-critical task. The program may then relinquish this exclusive access when it is not needed.

SHARED RUN TIME SYSTEM SUPPORT

TSX-Plus provides a facility that allows shared run time systems or data areas to be mapped into the address spaces of multiple time sharing jobs. Table 6 lists the routines that support this feature.

IASRNT	Associate/disassociate a shared run time system with a job.
MAPRNT	Map a shared run time system into a job's region.

Shared Run Time System Support.
Table 6.

Memory space can be conserved by having several jobs access a common copy of a run time system rather than having to allocate space within each job. Shared run time systems are never swapped out of memory. When a job is associated with a run time system, a portion of the job's virtual memory is mapped so as to allow access to the run time system.

SHARED FILE SUPPORT

Table 7 lists the routines that offer access to the shared file record locking facility provided by TSX-Plus. This is useful in situations where programs being run from several terminals wish to update a common file. Through the record locking facility a program may gain exclusive access to one or more blocks in a shared file by locking those blocks. Other users attempting to lock the same blocks will be denied access until the first user releases the locked blocks.

ICKWTS	Check for writes to a shared file.
IDCLSF	Declare a file to be shared.
ISVST	Save the status of a shared file.
IUALBK	Unlock all locked blocks.
IUSPBK	Unlock a specific block.
LKBLK	Try to lock a block.
LKBLKW	Wait for a block to lock.

The recommended procedure for updating a shared file being accessed by several users is as follows:

1. Open the file.
2. Declare the file to be shared.
3. Lock all blocks which contain the desired record.
4. Read the locked blocks into memory.
5. Update the record.
6. Write the updated blocks to the file.
7. Unlock the blocks.
8. Repeat steps three through seven as required.
9. Close the file.

SYSTEM STATUS INFORMATION

Information typical of that returned by the SYSTAT keyboard command is made available to a running program by the routines listed in Table 8.

ICONTM	Determine the connect time for a job.
ICPUTM	Get the CPU time used by a job.
IEXSTS	Get a job's execution status.
ILNSTS	Check the status of a line.
IPGNAM	Get the name of the program being run by a job.
IPPNUM	Get the project-programmer number for a job.
MEMPOS	Determine the position of a job in memory.
MEMUSE	Determine the amount of memory used by a job.

System Status Information.
Table 8.

TERMINAL COMMUNICATION SUPPORT

The routines that allow a running program to communicate with a terminal are listed in Table 9.

TRmIN	Accept a string of characters from the terminal.
TRMMSG	Send a message to another terminal.
TRMOUT	Send a string of characters to the terminal.

Terminal Communications Support.
Table 9.

TERMINAL CONTROL SUPPORT

The several terminal control support routines are listed in Table 10.

BRKCTL	Establish break sentinal control.
HIEFOF	Turn off the high efficiency terminal mode.
HIEFON	Turn on the high efficiency terminal mode.
IACTCH	Check for pending activation characters.
ITRERR	Check for terminal input errors.
ITRTYP	Determine the terminal type.
TIMOUT	Set the terminal read time out value.

Terminal Control Support.
Table 10.

MISCELLANEOUS EMT SUPPORT

Table 11 lists the routines that support the several miscellaneous EMTs provided by TSX-Plus.

ISPBLK	Determine the number of free blocks in the spool file.
ISPY	Return values from within the simulated RMON (SYSLIB routine).
ITSLIC	Determine the TSX-Plus license number.
ITSLIN	Determine the TSX-Plus line number.
MEMSET	Set the memory allocation.

Miscellaneous EMT Support.
Table 11.

ODT ACTIVATION MODE SUPPORT

ODT activation mode may be turned on and off from within a running program. Table 12 lists the routines that support this feature. In this mode TSX-Plus considers all characters to be activation characters except the digits, the comma, the dollar sign and the semicolon.

RSRODT	Reset normal activation mode.
SETODT	Set ODT activation mode.

ODT Activation Mode Support.
Table 12.

REFERENCES

1. N. A. Bourgeois, Jr., "TSXLIB -- A Library Implementation of the TSX-Plus Programmed Requests", 11-490, DECUS/US Chapter Program Library, Marlboro, MA, October 1982.

2. S & H Computer Systems, Inc., "TSX-Plus Reference Manual", Nashville, TN, December 1980.

3. S & H Computer Systems, Inc., "TSX-Plus Version 2.2 System Release Notes", Nashville, TN.

4. Digital Equipment Corporation, "RT-11 Programmer's Reference Manual", AA-H378A-TC, Maynard, MA, March 1980.

5. S & H Computer Systems, Inc., "TSX-Plus Version 3.0 System Release Notes", Nashville, TN.

6. Robert B. Denny, et al, "C Language System for RT-11", 11-513, DECUS/US Chapter Program Library, Marlboro, MA, January 1982.

DECUS LIBRARY

The attched is the latest revised and new submissions to the DECUS library. Please publish them in the next "MiniTasker". Future abstracts should be submitted by the SIG's new librarian, Tom Shinal. Tom's address is:

General Scientific Corporation
1684 E. Gude Dr.
Rockville, MD 20850
(301) 340-2773

Sincerely,

Nick

N. A. Bourgeois, Jr.

11-SP-48 (new) by N. A. Bourgeois, Jr., Sandia National Laboratories, Albuquerque, NM.

Symposium Tape from the RT-11 SIG, Fall 1982, Anaheim.

11-600 (new) by David M. Goodman, Veterans Administration Hospital, Bedford, MA.

This is a collection of programs which allow the user of a VT125 graphics terminal to generate hardcopy of REGIS graphics on an EPSON MX-80 serial matrix printer. A subset of the programs is usable under RT-11 V4.0, with the full set usable under TSX-Plus V2.0. Two hardcopy formats are available: small (4.5" x 8") and large (8" x 11").

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

This TEXT System is a collection of software tools, which in combination with DECUS RUNOFF, form a comprehensive text processing system for technical manuscript preparation for output on a NEC Spinwriter with the Technical Math/Times Roman type thimble. The system is capable of super/subscripting and including portions of text from other files.

11-609 (new) by Robert Walraven, University of California, Davis, CA.

This is a two part package that runs under RT-11 and TSX-Plus to support communications and source file transfer between an RT-11 environment and VMS. The first part is a FORTRAN program to support the communications protocol. The second part is an RT-11 handler to support the modem. The handler uses XON/XOFF protocol with VMS.

11-611 (new) by Mark Pyatetsky, Fermi National Accelerator Laboratory, Batavia, IL.

This is an RT-11 device handler for the DR11-W which implements a communication protocol between two PDP-11 computers. Multiple logical links are possible at the same time. The handler is capable of recovering from time-outs and hardware problems on the other end.

11-612 (new) by Mark Pyatetsky, Fermi National Accelerator Laboratory, Batavia, IL.

This is a collection of routines which interface with the DR11-W handler (11-611). These routines implement no-wait, wait, and completion routine I/O. Multiple messages may be awaited and the error status of each message is saved until a status routine is called.

11-614 (new) by Alan V. Reinhart, Aydin Monitor Systems, Fort Washington, PA.

ADT provides identical capabilities to RT-11 as DEC's standard RSX-11 ODT, with a number of extras. There are no systax changes and the new features follow the RSX-11 ODT format. The biggest feature is that ADT is not linked with the application program, but loaded as a handler.

11-615 (new) by M. Russell Bakke, Trans-Lux Corporation, Norwalk, CT.

This program allows handling of 8" single density CP/M format floppy disks under RT-11 and TSX-Plus. The program makes use of a menu to allow viewing or printing the directory, copying files to or from the CP/M disk, erasing files, initializing the disk, and etc. "This program occasionally crashes TSX-Plus."

11-616 (new) by B. A. Harper, Blain, Bremner and Williams Pty. Ltd., Milton, Australia.

SPECS is a menu controlling program for use under RT-11 which interprets commands given in user created ASCII menu files, presents various options to the user and acts on the selections made. SPECS allows moving between various menus, chaining to other programs, sending commands to KMON, scheduling foreground or system jobs and communicating between jobs.

FORTRAN/RT TUTORIAL - EIS, FIS and FPU

Ron Trellue
Sandia National Laboratories
Albuquerque, New Mexico

David E. Ellis, Major, USAF
Air Force Weapons Laboratory
Kirtland AFB, New Mexico

ABSTRACT

This tutorial is a discussion of NHD, EIS, FIS, and FPU. The hardware available to implement these additional arithmetic instruction sets is presented. Building the Fortran IV compiler and library to take maximum advantage of the hardware is discussed. Examples of threaded and inline code are given to show the output from the various types of Fortran compilers. The results of running five programs built for the different arithmetic options illustrate the advantages of additional arithmetic hardware. The tests were run on a LSI 11/2, a LSI 11/23 with both the KEF11-AA chip and the FPF11 processor, a PDP 11/34, a PDP 11/45. a PDP 11/44, and a VAX 11/730.

INTRODUCTION

One of my functions at Sandia Laboratories is to provide RT-11 software to new users at Sandia. In addition to giving out RT-11, I give each user all the object modules needed to build the Fortran IV compiler and library. Each user is asked what kind of compiler and Fortran library they would like to have. Many people are unsure of what hardware they have, so they are not sure how they want their Fortran system built.

The purpose of this tutorial is to help clarify some of the options available in building a Fortran system, and in particular to discuss the meaning of EIS, FIS, and FPU. The following subjects will be covered:

- o What floating point hardware is available ?
- o What are the EIS, FIS, and FPU instructions ?
- o Why does anyone care ?
- o How do you get the software to take advantage of the hardware ?
- o How do you build the proper compiler and library ?
- o What code is generated by the various types of compilers ?
- o How does the type of code effect program execution times ?

HARDWARE/SOFTWARE TEAMWORK

The discussion of EIS, FIS, and FPU centers around the way the computer handles arithmetic operations such as add, subtract, multiply, and divide. Optional hardware can be purchased to implement certain arithmetic operations in hardware rather than in software. Different hardware is available for various types of processors. The software should be built to take the maximum advantage of the type of hardware present on the machine.

Every machine has one or more of the following types of hardware:

- o NHD -- No hardware
- o EAE -- Extended Arithmetic Element
- o EIS -- Extended Instruction Set
- o FIS -- Floating Instruction Set
- o FPU -- Floating Point Unit

There are several ways to determine the type of arithmetic hardware on your machine. One way is to look at your purchase order to see what you bought. Another is to look in your cabinet and see what type of board is there. A hard way is the trial and error method. RT-11 has a more sophisticated way to find out, just type

```
.SHOW CONFIGURATION
```

TYPES OF HARDWARE AVAILABLE

Some machines such as the LSI 11/23 and the PDP 11/44 come with EIS standard with the processors. Some products such as the KEF11-AA chip and the FPF11 board must be bought separately. Not all types of hardware are available for each type of processor. Table 1 shows the hardware options which are available for a variety of processors. EAE is only available on a small number of older machines, and will not be discussed in this paper.

==

Table 1. Arithmetic Hardware Options

EAE - Extended Arithmetic Element

KE11A,B	-	11/10, 11/15, 11/35, 11/40

EIS - Extended Instruction Set

KE11-E	-	11/35, 11/40
KEV11	-	LSI-11, 11/03
Standard	-	LSI 11/23, 11/34, 11/44, 11/60, 11/70

FIS - Floating Instruction Set

KE11-F	-	11/35, 11/40
KEV11	-	LSI-11, 11/03

FPU - Floating Point Unit

KEF11-AA	-	LSI-11/23 Microcode option
FPF-11	-	LSI 11/23
FP11	-	11/34, 11/44, 11/60, 11/70

EIS, FIS, AND FPU INSTRUCTIONS

The EIS, FIS, and FPU instructions extend the normal PDP-11 instruction set by providing additional instructions for certain arithmetic operations. These instructions are implemented in the hardware indicated by Table 1. The trick is to get the software to generate these additional instructions so that the hardware can take advantage of them. One way to make use of them is to write your own assembly language program and use them in it. Another is to get the high level language processors to generate these instructions. In the Fortran world, this means having the Fortran compiler generate the proper code, and building the Fortran library with the pre-compiled OTS routines which utilize the appropriate arithmetic instructions.

WHAT IS EIS ?

EIS, the Extended Instruction Set provides only four additional instructions.

MUL -- Fixed point multiply (32 bit)

DIV -- Fixed point divide (32 bit)

ASH -- Arithmetic shift (16 bit)

ASHC - Arithmetic shift combined (32 bit)

These instructions only effect integer operations, and do not effect either single or double precision arithmetic. The EIS instructions are standard on most of the newer processors.

WHAT IS FIS ?

The FIS instructions are only available on a small number of older systems, and are not available on most of the newer machines which support a floating point processor. The FIS instructions are:

FADD - Floating add

FSUB - Floating subtract

FMUL - Floating multiply

FDIV - Floating divide

The FIS instructions only work on single precision arithmetic. The results of our time tests will illustrate the advantage of having FIS when doing a large amount of single precision arithmetic.

WHAT IS FPU ?

FPU is a floating point unit which provides arithmetic hardware to implement 46 floating point instructions. They are:

ABSF	LDCDF	MULF	STCDI
ABSD	LDCFD	MULD	STCDL
ADDD	LDCIF	NEGF	STEXP
ADDF	LDCID	NEGD	STFPS
CFCC	LDCLF	SETF	STF
	LDCLD	SETD	STD
CLRF	LDEXP	SETI	STST
CLRD	LDFPS	SETL	
CMPF	LDF	STCFD	SUBF
CMPD	LDD	STCDF	SUBD
DIVF	MODF	STCFI	TSTF
DIVD	MODD	STCFL	TSTD

These instructions are implemented on a PDP-11 with a FP11 floating point processor. On an LSI 11/23 they can be implemented in microcode on the KEF11-AA chip, or on the FPF-11 processor.

WHO CARES ?

The timing results given later in this paper will indicate the benefits which can be attained by taking advantage of the arithmetic hardware. In certain cases the increase in speed is substantial. Investing in floating point hardware can be expensive, but depending on the types of calculations to be done, floating point hardware can greatly speed up program execution.

THREADED CODE

The Fortran IV compiler can be built to produce either inline or threaded code. Threaded code is hardware independent. This means that the compiler produces the same code no matter what hardware is available on the machine. Threaded code consists of calls to routines in the Fortran Object Time System (OTS). The threaded code calls thread their way through the OTS routines which were built for a particular type of hardware. The Fortran library containing the OTS can be built for any one of the five types of arithmetic hardware - NHD, EAE, EIS, FIS, or FPU. A threaded code object file could be linked with any one of the five types of OTS libraries which can be built. Threaded code is usually used with no hardware (NHD) or floating point (FPU) libraries. The Fortran debugger, FDT, only works when linked with threaded code object modules.

INLINE CODE

The inline code compiler generates actual assembly language instructions for each Fortran statement, with fewer calls to library routines. The Fortran IV compiler can only produce inline code for EAE, EIS, and FIS. This means that if you have no arithmetic hardware, (NHD), then you must build a threaded code compiler. If you have a floating point unit, you can build a threaded code compiler, or if you have both EIS and FPU, you can build a compiler which produces inline EIS code, to link with an FPU library. The Fortran IV V2.5 compiler can not produce inline FPU code. Shame on DEC ! The Fortran 77 compiler available under RSX-11M is capable of producing inline FPU code. Some of the timing tests presented later in this paper show the advantages of having inline FPU code.

BUILDING THE COMPILER

The Fortran compiler is built from the Fortran pieces which are distributed as a set of object modules. The input and output devices are assigned, and the FORGEN program run.

```
.ASSIGN DL1 INP

.ASSIGN DL0 OUP

.RUN INP:FORGEN
```

The FORGEN program asks questions about how the compiler should be built. The first question concerns the number of lines allowed per listing page. This depends on the type of line printer you have. The next two questions on the maximum size of a formatted ASCII record and the number of channels open at a time can both be overidden at compile time, so taking the defaults is normally adequate. Finally comes the question about the type of code that the compiler should produce. The choices are threaded, inline EAE, inline EIS, or inline FIS. You can build a threaded code only compiler, an inline only compiler, or a compiler which will do threaded or inline. If you wish to build a compiler that can do both, the way you answer the questions will determine the default setting.

At the end of the question session, a command file, FORBLD.COM is created and can be initiated to build the compiler. The file FORBLD.COM only has one command in it. It will start up one of the following three indirect command files to link together a compiler which will be threaded, inline, or both depending on which command file is run.

F4LTHR.COM - Threaded code only compiler, about 130 blocks in size

F4LINL.COM - Inline code only compiler, about 180 blocks in size

F4LINK.COM - Both inline and threaded compiler, about 210 blocks

How do the answers to the questions get incorporated into the building of the compiler ? They are saved in a file on OUP: called DEFLTS.OBJ. The file DEFLTS.OBJ is then linked with the appropriate Fortran pieces as specified by one of the three F4L*.COM files to produce a tailored compiler. I like to build a compiler which does both inline EIS by default, and threaded code when asked. Most applications run faster with inline code. Threaded code is handy when a program gets too big, since threaded code is usually smaller than inline code. Threaded code is mandatory when using the Fortran debugger, FDT.

BUILDING THE FORTRAN LIBRARY

The second piece of software required for a Fortran development environment is the Fortran library or OTS. The Fortran OTS can be added to the System Subroutine Library SYSLIB.OBJ, or can be put in a separate file with any name you choose. If the Fortran library is to be added to SYSLIB.OBJ, then SYSLIB.OBJ must exist on the output device. Again device assignments must be made, and the OTS build program can be run.

```
.ASSIGN DL1 INP

.ASSIGN DLO OUP

.RUN INP:OTSGEN
```

The program will ask a few questions concerning the building of the OTS. One of the important questions is what type of code should the library be built for - NHD, EAE, EIS, FIS, or FPU. This is where you get to specify how you want the library to be built. A command file is created which can be initiated by typing "OUP:OTSBLD.

The OTSBLD command file uses the librarian to build the Fortran library from the Fortran pieces. The way you answer the questions determines which modules are put into the library. When the compiler is built, the FORGEN program must be run to create the file DEFLTS.OBJ. It is quite easy to build the library yourself by selecting the pieces you want, and putting them into the library. A typical build might look like this:

```
.R LIBR
*OUP:SYSLIB=SYSLIB,INP:FPU,INP:OTSCOM/G/C
*INP:VIRNP,INP:V2NS
*//
Global? $ERRS
Global? $ERRTB
Global? $OVRH
Global?
*↑C
```

In building the Fortran OTS, the following modules can be used.

```
 This module is mandatory.
   OTSCOM.OBJ - Routines everyone needs

 Pick one of these, depending on hardware.
   NHD.OBJ - No hardware, software does it.
   EAE.OBJ - Routines with EAE instructions
   EIS.OBJ - Routines with EIS instructions
   FIS.OBJ - Routines with FIS instructions
   FPU.OBJ - Routines with FPU instructions

 Pick one of these types of virtual array
 support.
   NOVIR.OBJ   - No virtual array support
                 needed
   VIRNP.OBJ   - Virtual array support under
                 SJ or FB needed
   VIRP.OBJ    - Virtual array support under
                 XM is needed

 Pick one of these if threaded code will be
 used.
   V2NS.OBJ    - No array subcript checking,
                 threaded code only
   V2S.OBJ     - Do array subcript checking,
                 threaded code only

 Optional features.
   UNI.OBJ     - Provides SIMRT, standalone
                 Fortran support if needed
   UIOBYT.OBJ  - Provides byte resolution in
                 direct access files
```

The size of the library will depend on the number of modules selected, and on the type of arithmetic support requested. For example an NHD no hardware library will produce the largest library file since all the arithmetic operations must be performed in software rather than hardware. The FPU library will be the smallest since the routines rely on the hardware to do the work.

THE TEST ROUTINES

In order to see if the different code options - NHD, EIS, FIS, and FPU, really make any difference in the execution time of a program, several programs were compiled under the various options and the run times recorded. Five programs were used.

```
CCA1.FOR   -  Single precision calculations
CCA2.FOR   -  Same as CCA1, except double
              precision
CCA5.FOR   -  Fortran library function test
CCA9.FOR   -  Binomial expansion
CCA10.FOR  -  IF test
```

Programs CCA1 and CCA2 do a variety of floating point operations such as addition, subtraction, multiplication, and division as well as array element addressing and operations with functions. Parts of the programs are as follows.

```
      PROGRAM CCA1
C-----------------------------------------
      STIME=SECNDS(0.0)
          .
          .
11        DO 18 I = 1,N1,1
             X1 = (X1+X2+X3-X4)*T
             X2 = (X1+X2-X3+X4)*T
             X3 = (X1-X2+X3+X4)*T
             X4 = (-X1+X2+X3+X4)*T
18        CONTINUE
```

```
         .
21       DO 28 I = 1,N2,1
         E1(1) =(E1(1)+E1(2)+E1(3)-E1(4))*T
         E1(2) =(E1(1)+E1(2)-E1(3)+E1(4))*T
         E1(3) =(E1(1)-E1(2)+E1(3)+E1(4))*T
         E1(4) =(E1(1)+E1(2)+E1(3)+E1(4))*T
28       CONTINUE
         .
         .
61       DO 68 I = 1,N6,1
            J = J*(K-J)*(L-K)
            K = L*K-(L-J)*K
            L = (L-K)*(K+J)
            E1(L-1) = J+K+L
            E1(K-1) = J*K*L
68       CONTINUE
         .
         .
71       DO 78 I = 1,N7,1
         X=T*ATAN(T2*SIN(X)*COS(X)/COS(X+Y))
         Y=T*ATAN(T2*SIN(Y)*COS(Y)/COS(X+Y))
78       CONTINUE
         .
         .
118      X = SQRT(EXP(ALOG(X)/T1))
         .
         .
      TTIME=SECNDS(STIME)
```

CCA2 is identical to CCA1 except that the variables are declared double precision. CCA5 is the Fortran library function test.

```
      PROGRAM CCA5
C---------------------------------------
      STIME=SECNDS(0.0)
      N = 26400
      C = 1.0
      DO 1 I=1,N
      D = C
      E = C
      A = SQRT(C)
      B = SIN(D)
      C = COS(E)
      D = SQRT(A)
      E = EXP(B)
      A = SIN(C)
      B = SQRT(D)
      C = COS(E)
      D = ATAN(A)
      E = SQRT(B)
      A = COS(C)
      B = EXP(D)
      C = SQRT(E)
      D = SIN(A)
      E = COS(B)
      C = SQRT(C)
      B = COS(D)
      A = EXP(E)
      D = ATAN(A)
 1    CONTINUE
      A = A + B + D + E
C---------------------------------------
        TTIME=SECNDS(STIME)
```

CCA9 is a binomial expansion.

```
      PROGRAM CCA9
C---------------------------------------
      STIME=SECNDS(0.0)
        .
        .
      DO 600 M=1,IA
      BC=0.0
      BB=1.0
      DO 500 N=1,IE
      DO 400 K=IB,IC,ID
      AK=K
      L=N+1
      MM=L/2
      COMB(1)=1.0
      COMB(L)=1.0
      DO 120 I=2,MM
      AI=I-1
      L=L-1
      AN=L
      COMB(I)=AN/AI*COMB(I-1)
  120 COMB(L)=COMB(I)
      IF(N+1-MM*2)150,151,150
  150 COMB(L-1)=(AN-1.0)/(AI+1.0)*COMB(L)
  151 L=N+1
      Q=AK/AA
      P=1.0-Q
      BI=Q**N
      BII=BI
      DO 160 I=2,L
      BI=BI*COMB(I)/COMB(I-1)*P/Q
  160 BII=BII+BI
  400 BB=BB*BII
  500 BC=BC+BII
  600 CONTINUE
C---------------------------------------
      TTIME=SECNDS(STIME)
```

CCA10 is the IF check routine.

```
      PROGRAM CCA10
C---------------------------------------
      STIME=SECNDS(0.0)
        .
        .
      DO 300 L=1,N
      DO 300 M=1,N
      GO TO 200
  100 IF(I)101,999,999
  101 IF(J)999,102,999
  102 IF(K)999,999,103
  103 IF(I)104,999,999
  104 IF(J)999,105,999
  105 IF(K)999,999,106
  106 IF(I)107,999,999
  107 IF(J)999,108,999
  108 IF(K)999,999,109
  109 IF(I)110,999,999
  110 IF(J)999,111,999
  111 IF(K)999,999,112
  112 CONTINUE
      GO TO 300
  200 IF(I)201,999,999
  202 IF(K)999,999,203
  201 IF(J)999,202,999
  203 IF(I)204,999,999
  205 IF(K)999,999,206
  204 IF(J)999,205,999
  206 IF(I)207,999,999
```

program gets too big, since threaded code   Pick one of these types of virtual array

```
  208 IF(K)999,999,209
  207 IF(J)999,208,999
  209 IF(I)210,999,999
  211 IF(K)999,999,212
  210 IF(J)999,211,999
  212 CONTINUE
      GO TO 100
  999 CONTINUE
  300 CONTINUE
C-----------------------------------
      TTIME=SECNDS(STIME)
```

GENERATED CODE

The Fortran compiler can list the generated code which it produces. A statement was selected from program CCA9 and the generated code for threaded, inline EIS, inline FIS, and inline FPU was listed. The statement chosen was:

```
  150  COMB(L-1)=(AN-1.0)/(AI+1.0)*COMB(L)
```

Example 1 shows the code generated by a Fortran IV threaded code compiler. The left column gives the relative octal byte address of each threaded code call, and the next column gives the name of a threaded code routine from the Fortran library executed, followed by any pertinent parameters. A discussion of threaded code is available on an earlier RT-11 SIG DECUS tape.

Example 1. Threaded Code

```
000404  SAF$MM  $DATA+#004000 $DATA+#177770
000412  MOF$MS  $DATA+#004012
000416  SUF$IS  #040200
000422  MOF$IS  #040200
000426  ADF$MS  $DATA+#004006
000432  DIF$SS
000434  SVF$MM  $DATA+#004000 $DATA+#177774
000442  MUF$SS
000444  MOF$SA
```

Example 2 shows the code generated for the same statement from an inline compiler built for EIS code. Inline code contains the actual MACRO-11 instructions for each statement. This particular statement required more code to be generated by the inline compiler than did the threaded version. Since no integer operations are performed, none of the four EIS instructions appear in the generated code for this particular example. Both types of code require library routines to be called, so the total number of instructions required to execute this statement is not obvious. Inline code is normally larger than threaded code, and usually faster. The inline code in Example 2 shows that the arithmetic operations are done by jumping to subroutines (JSR PC,$ADDF). These subroutines such as $ADDF are taken from the Fortran OTS and can contain EIS instructions, or FPU instructions depending on which arithmetic module was selected when the library was built. The more instructions which can be implemented in hardware, the faster and smaller the routine will be.

Example 2. EIS Code

```
000542     MOV    AN+2,-(SP)
000546     MOV    AN,-(SP)
000552     CLR    -(SP)
000554     MOV    #40200,-(SP)
000560     JSR    PC, $SUBF
000564     CLR    -(SP)
000566     MOV    #40200,-(SP)
000572     MOV    AI+2,-(SP)
000576     MOV    AI,-(SP)
000602     JSR    PC,$ADDF
000606     JSR    PC,$DIVF
000612     MOV    L,R0
000616     ASL    R0
000620     ASL    R0
000622     MOV    COMB-2(R0),-(SP)
000626     MOV    COMB-4(R0),-(SP)
000632     JSR    PC,$MULF
000636     MOV    L,R0
000642     ASL    R0
000644     ASL    R0
000646     MOV    (SP)+,COMB-10(R0)
000652     MOV    (SP)+,COMB-6(R0)
```

Example 3 shows the code generated by an FIS inline compiler. FIS code required the same number of statements to be generated. Notice that the single precision arithmetic operations implemented in EIS code by a subroutine call, such as JSR PC,$ADDF, have been replaced with simple FIS instructions like FADD. This is the advantage of FIS code where the FADD instruction is executed immediately by the hardware unit.

Example 3. FIS Code

```
000536     MOV    AN+2,-(SP)
000542     MOV    AN,-(SP)
000546     CLR    -(SP)
000550     MOV    #40200,-(SP)
000554     FSUB   SP
000556     CLR    -(SP)
000560     MOV    #40200,-(SP)
000564     MOV    AI+2,-(SP)
000570     MOV    AI,-(SP)
000574     FADD   SP
000576     FDIV   SP
000600     MOV    L,R0
000604     ASL    R0
000606     ASL    R0
000610     MOV    COMB-2(R0),-(SP)
000614     MOV    COMB-4(R0),-(SP)
000620     FMUL   SP
000622     MOV    L,R0
000626     ASL    R0
000630     ASL    R0
000632     MOV    (SP)+,COMB-10(R0)
000636     MOV    (SP)+,COMB-6(R0)
```

Example 4 is the generated code from the RSX-11M Fortran 77 compiler. It shows the benefits of inline FPU code. The code is small, and no subroutine or threaded library routine calls are necessary.

Example 4. FPU Code

```
000472    LDF     #40200,F0
000476    LDF     AN,F1
000502    SUBF    F0,F1
000504    ADDF    AI,F0
000510    DIVF    F0,F1
000512    MOV     R2,R0
000514    ASL     R0
000516    ASL     R0
000520    MULF    COMB-4(R0),F1
000524    STF     F1,COMB-10(R0)
```

GROUND RULES

Each of the five test programs were built a variety of ways to illustrate the execution speed of the different types of code. The first set of tests were all run on the same PDP 11/45. Since the 11/45 has EIS and a floating point processor, tests could be run with NHD threaded code, inline EIS code and library, threaded code and the FPU library, and inline EIS code linked with the FPU library. We call this option EPU. The 11/45 does not support FIS, so another machine was used to demonstrate FIS timing.

The programs were compiled under the Fortran IV/RT V2.5 compiler patched through Autopatch F and run under RT-11 V4.0 patched through Autopatch F. The routines were all compiled without linenumbers to increase execution times. Linenumbers are of great value when a program aborts, but add overhead in time and size to a working program.

The same programs were also run on the PDP 11/45 under the TSX Plus system available from S&H Computer Systems, Inc. TSX Plus provides a multi-user RT-11 environment. The routines were then re-built using Fortran IV V2.5 under RSX-11M V4.0. This was done to get a comparision of RT-11 versus RSX-11M run times. It is interesting to note that the Fortran IV compiler under RT-11 and the Fortran IV compiler under RSX-11M produced identical code. The routines were also built and run using Fortran 77 under RSX-11M. This was done to see if Fortran 77 is indeed faster than Fortran IV since it does produce inline FPU code. The routines were re-compiled and run on a VAX 11/730 under VMS for comparision with RT-11 and RSX.

The routines were compiled with no linenumbers, the test programs were the only job running, and no print spoolers, system jobs, other users, or additional processing was taking place. None of the programs did any I/O during the timing period.

The programs were also run on an LSI-11/2 to illustrate the difference between the NHD, EIS and FIS options. Two different LSI-11/23s were used to show the relative speed of the FPF-11 floating point unit compared to the KEF11-AA microcode FPU chip. The programs were also run on an 11/34 and an 11/44 with floating point processors, just for fun.

Table 2 shows the three character names used to identify the different options for the timing test results.

===

Table 2. Naming Convention

NHD	-	Threaded code compiler, NHD library
EIS	-	Inline EIS compiler, EIS library
FIS	-	Inline FIS compiler, FIS library
FPU	-	Threaded code compiler, FPU library
EPU	-	Inline EIS compiler, FPU library
F4P	-	RSX-11M F77 compiler with /F4P
F77	-	RSX-11M F77 compiler and library

SINGLE AND DOUBLE PRECISION RESULTS

Tables 3 and 4 give the run times of programs CCA1 and CCA2 built for the various options. CCA1 and CCA2 are identical, except that CCA1 has single precision variables, and CCA2 has all double precision variables. CCA1 and CCA2 do a diverse set of operations such as adds, multiplies, array referencing, trigonometric functions, and exponentiation as shown by the program sample given earlier. The other programs are much more specialized. Since none of the programs do any I/O, such as reading data from a file, they may not seem representative of the average user program. Their purpose, however, is to illustrate the different types of code which primarily effect floating point arithmetic operations.

The CCA1 and CCA2 run times indicate that the more arithmetic hardware available the better. For single precision, the floating point processor reduced run time from 5.8 minutes to 1.4 minutes over using no hardware, and from 4.5 minutes to 1.4 minutes over EIS. On double precision, CCA2, the reduction was even more dramatic from 26 minutes NHD to 18.6 with EIS, to 2.0 minutes with EPU. The EPU option, EIS inline code linked with an FPU library, seemed to produce the best results. It takes advantage of the fastest inline code available to Fortran IV, plus the FPU library. Inline code, however, often produces a larger .SAV image. For applications which do a large number of double precision calculations, buy a floating point processor !

Table 3. CCA1 Times on PDP 11/45

Single Precision
(minutes)

	RT	TSX	RSX
NHD	5.82	6.37	6.96
EIS	4.47	4.85	5.30
FPU	1.62	1.75	1.88
EPU	1.42	1.53	1.64
F4P	-	-	.69
F77	-	-	.69

Table 4. CCA2 Times on PDP 11/45

Double Precision
(minutes)

	RT	TSX	RSX
NHD	25.98	28.32	30.94
EIS	18.65	20.28	22.15
FPU	2.18	2.37	2.50
EPU	2.00	2.17	2.30
F4P	-	-	.90
F77	-	-	.90

RT-11 VERSUS TSX PLUS AND RSX-11M

Tables 3 and 4 also show that both TSX Plus and RSX impose some overhead on program execution. This increase in run time should be expected for a multi-tasking system since overhead must be incurred to see if anyone else is ready to run. The advantage of a multi-tasking system is that while one job is waiting for I/O, another job can be executing so that the total run time for the two jobs can be less than running them consecutively.

Table 5 shows the size of the .SAV and .TSK files from program CCA1 for both RT-11 and RSX-11M. In both systems the inline code versions, CCA1.EIS and CCA1.EPU, produce slighlty more code but run a little faster than the equivalent threaded code version, CCA1.FPU.

While TSX Plus can execute the same .SAV files as RT-11, the RSX .TSK files were appreciably larger than the RT-11 image. The more sophisticated, relocatable nature of a multi-tasking system such as RSX caused the .TSK files to be almost twice as large as the .SAV image. Remember that the compiler produced identical code, but the library routines under RSX must be smarter to run in that kind of environment.

Table 5. .SAV Versus .TSK File Sizes

	RT-11		RSX-11M	
	.SAV (blks)	Time (min)	.TSK (blks)	Time (min)
CCA1.NHD	28	5.82	43	6.96
CCA1.EIS	28	4.47	44	5.30
CCA1.FPU	25	1.62	41	1.88
CCA1.EPU	26	1.42	42	1.64
CCA1.F4P	-	-	44	.69
CCA1.F77	-	-	44	.69

FORTRAN 77

Notice that the F4P and F77 execution times under RSX-11M are twice as fast as the best that the RT-11 Fortran IV compiler can do. This is the advantage of the F77 compiler which produces highly optimized, inline FPU code. Table 5 shows that the size of the .TSK files increased slightly over the equivalent .TSK file produced by the Fortran IV compiler, but the F77 versions ran much faster.

TRIG FUNCTIONS, BINOMIAL, AND IF TEST

Tables 6, 7, and 8 show the results of the single precision trig function test (CCA5), the binomial expansion (CCA9), and the IF test (CCA10). Table 6 shows that Fortran IV has a good implementation of trigonometric functions taking full advantage of the floating point hardware. Table 7 shows several interesting results. For some strange reason CCA9 blew up under TSX Plus even though the same exact .SAV file ran fine under RT-11. SPR time for S&H Computer Systems. CCA9 also aborted under RSX when the floating point hardware was NOT used. SPR time again. CCA9 is also a somewhat rare example where threaded/FPU code is actually faster than inline EIS/FPU code. It just depends on what the program is doing. Sometimes it is best to time the program both ways, and use whichever is most effective.

CCA10, the IF test shown in Table 8, illustrates that EIS can be much faster than threaded code. It also shows that having a floating point processor does not help when the program does not do any floating point arithmetic. In this case EIS is as fast as EPU.

Table 6. CCA5 Times on PDP 11/45

Functions
(minutes)

	RT	TSX	RSX
NHD	17.67	19.33	21.13
EIS	13.20	14.37	15.65
FPU	1.93	2.07	2.14
EPU	1.67	1.75	1.81
F4P	-	-	1.87
F77	-	-	1.87

Table 7. CCA9 Times on PDP 11/45

Binomial
(minutes)

	RT	TSX	RSX
NHD	16.03	17.58	?
EIS	12.70	13.58	?
FPU	3.58	?	4.08
EPU	4.48	?	5.17
F4P	-	-	1.59
F77	-	-	1.59

? means that the program aborted

Table 8. CCA10 Times on PDP 11/45

IF test
(minutes)

	RT	TSX	RSX
NHD	5.35	5.90	6.37
EIS	1.52	1.67	1.81
FPU	5.35	5.90	6.37
EPU	1.52	1.67	1.81
F4P	-	-	1.87
F77	-	-	1.87

EIS VERSUS FIS

Tables 9 and 10 show the results of running CCA1 and CCA2 on an LSI 11/2 which has both EIS and FIS. For single precision arithmetic, the four FIS instructions decreased execution time from 15 minutes with EIS to 4.8 minutes with FIS. On double precision, Table 10, the FIS instructions offered no speed increase over EIS. This is due to the fact that the FIS instructions are only implemented for single precision arithmetic. Tables 11, 12, and 13 also point out that the FIS option can be advantageous when single precision arithmetic is performed.

DIFFERENT MACHINES

Tables 9 through 13 detail the execution times of the same programs run on an LSI 11/2 with both EIS and FIS, an 11/23 with EIS and an FPF11 floating point processor, an 11/34, 11/45, and an 11/44 all with an FP11. The 11/23 appears to be about twice as fast as the 11/2 for every test. One exception is double precision. Since the 11/2 does not support a floating point processor, it cannot take advantage of any double precision arithmetic. It is interesting to note that an 11/2 with FIS can execute single precision arithmetic almost as fast as an 11/23 with the KEF11-AA microcode floating point chip - for CCA1 it took 4.8 minutes for the 11/2 versus 4.1 minutes for the 11/23 (Tables 9 and 14). Many people have been dismayed when they upgraded their 11/2 with FIS to an 11/23, and the single precision run times were only minimally decreased.

The execution times of the various machines show their relative power. No big surprises. The tests were run on the 11/23 and 11/34 because they are two of the more common systems currently in use, and relate to a large community of users. The test results show that the 11/44 was twice the speed of the 11/34 on the NHD and EIS tests, and almost that much faster on the floating point tests.

Table 9. CCA1 - Various Processors

Single Precision

	11/2	11/23	11/34	11/45	11/44
NHD	18.47	8.12	7.93	5.82	3.65
EIS	15.43	7.13	5.63	4.47	2.75
FIS	4.80	-	-	-	-
FPU	-	2.07	1.78	1.62	0.98
EPU	-	1.87	1.58	1.42	0.93

Table 10. CCA2 - Various Processors

Double Precision

	11/2	11/23	11/34	11/45	11/44
NHD	83.07	35.25	34.05	25.98	15.47
EIS	57.00	26.42	22.90	18.65	10.60
FIS	57.00	-	-	-	-
FPU	-	2.72	2.38	2.18	1.42
EPU	-	2.55	2.20	2.00	1.30

Table 11. CCA5 - Various Processors

Functions

	11/2	11/23	11/34	11/45	11/44
NHD	60.03	25.07	25.48	17.67	11.90
EIS	49.63	21.67	17.40	13.20	8.63
FIS	14.42	-	-	-	-
FPU	-	2.90	2.73	1.93	1.88
EPU	-	2.55	2.43	1.67	1.73

Table 12. CCA9 - Various Processors

Binomial

	11/2	11/23	11/34	11/45	11/44
NHD	52.10	22.57	22.77	16.03	10.70
EIS	47.32	21.50	15.95	12.70	8.65
FIS	13.83	-	-	-	-
FPU	-	4.28	3.73	3.58	2.15
EPU	-	5.65	4.75	4.48	2.75

Table 13. CCA10 - Various Processors

IF test

	11/2	11/23	11/34	11/45	11/44
NHD	12.82	6.03	5.67	5.35	2.20
EIS	4.35	1.93	1.85	1.52	0.77
FIS	4.35	-	-	-	-
FPU	-	6.03	5.67	5.35	2.20
EPU	-	1.93	1.85	1.52	0.77

KEF11-AA VERSUS FPF11

Table 14 points out the difference between the two floating point options available on the LSI 11/23. The KEF11-AA chip implements the floating point instructions in microcode. From Tables 9 and 14 it can be seen that the microcode chip can decrease processing time for CCA1 from 7 minutes to 4, and for CCA2 from 26 minutes to 7. That alone is a substantial savings. The FPF11 floating point processor implements the floating instruction set in hardware, and cuts the execution time of the KEF11-AA in half. Notice that the FPF11 did a much better job of executing the trig functions of CCA5.

Table 14. KEF11-AA versus FPF11

LSI 11/23 Floating Point

Program	FPF11	KEF11-AA
CCA1.FPU	2.07	4.07
CCA1.EPU	1.87	3.85
CCA2.FPU	2.72	7.32
CCA2.EPU	2.55	7.17
CCA5.FPU	2.90	11.70
CCA5.EPU	2.55	11.33
CCA9.FPU	4.28	9.07
CCA9.EPU	5.65	10.45
CCA10.FPU	6.03	6.00
CCA10.EPU	1.93	1.95

VAX 11/730

The test routines were run on a VAX 11/730. The 730 is an attractive system for migration for the RT-11 user. Table 15 shows the execution times of two different 11/730 systems, one with the FP730 floating point processor and one without. These times are listed along with the comparible times from the RSX-11M Fortran 77 run times on the 11/45 and 11/44. VAX times are more accurately compared with the RSX times rather than the RT-11 times since VMS supports Fortran 77. Fortran 77 has been shown to be substantially faster than Fortran IV running on the same machine (Tables 3 through 8). The 11/730 appears to be about the same speed as the 11/44, and has the advantages of the VMS operating system.

Table 15. VAX 11/730 versus 11/45, 11/44

Program	11/45	11/44	11/730	11/730
	RSX/F77		no FP	FP730
CCA1	0.69	0.56	1.76	0.56
CCA2	0.90	0.74	3.51	0.89
CCA5	1.87	1.82	5.91	1.37
CCA9	1.59	1.30	6.73	0.99
CCA10	1.87	0.87	1.90	1.90

CONCLUSIONS

EIS - the Extended Instruction Set, FIS - the Floating Instruction Set, and FPU - the Floating Point Unit are hardware/software solutions to faster execution of integer, single precision, and double precision arithmetic operations. EIS is standard on most new CPUs, while FIS is only available on some older model processors. For integer operations, EIS is very useful. EIS inline code linked with a floating point library (EPU) can be quite effective in the RT-11 environment. Inline FPU code has been shown to have great potential in decreasing run times, but is not currently available to the RT-11 Fortran IV community.

Double precision arithmetic on all types of machines will take four times as long to execute as the same program doing single precision, when the operations must be performed in software. From Table 9 CCA1 took 8.12 minutes on the 11/23 for NHD and 35.25 minutes for the double precision version, CCA2. For EIS the difference was 7.13 minutes for CCA1 and 26.42 minutes for CCA2. In contrast, when the floating point operations could be performed directly by the hardware (EPU), the times only changed from 1.87 minutes for CCA1 to 2.55 minutes for CCA2.

On a LSI 11/2, FIS can greatly speed up single precision operations by doing the arithmetic in hardware rather than software. This is reason the LSI 11/2 can perform single precision arithmetic almost as fast as a LSI 11/23 with the KEF11-AA chip which implements the 46 floating point instructions in microcode.

A floating point processor can improve the execution of double precision arithmetic by an order of magnitude over having to perform the same operations in software (Table 10 FPU and EPU versus NHD and EIS).

The software must be tailored to take maximum advantage of the hardware present on a given machine. The Fortran compiler can be built to generate the additional instructions implemented by the various types of hardware. Which type of hardware/software is best for a particular application depends on what type of calculations are to be done, and what types of hardware is available.

In each application it is important to know what types of operations are to be executed, what type of hardware is available, and how to build the software to take maximum advantage of the hardware.

RTEM-11: The RT Emulator

Don Bridgewater
Digital Equipment Corporation
Maynard, MA

John Rasted, Session Chairperson
JTR Associates
Meriden, CT

Reported by Gavin Perry, DECUS Scribe Service

The RT development group has produced a simulated RT machine that can be run as a task on RSX-11M or M-PLUS or under VMS on a VAX. The product is called RTEM-11. It emulates the RT-11 FB stand alone environment to provide an RT-11 development system in a multiuser environment. This product is not meant to be used as an application environment. It has severe limitations on its I/O capabilities.

RTEM provides RT-11 FB development and runtime environments. It runs the same .SAV files as stand alone RT. It allows several RT-11 developers to use an RSX-11M system simultaneously. Free standing RT-11 systems should be used for running applications. RTEM provides maximum compatibility by utilizing RT-11 monitor code. Two types of RT system devices are provided for.

An RT-11 format "virtual disk" file on a FILES-11 volume
A foreign mounted RT-11 format volume (RK05, RX01...)

DCL (Digital Command Language) and indirect command files are supported. Most RT-11 distributed utilities are supported. There are three new utilities. A Files Interchange Program (FIP) allows transfer between RT-11 files and host FILES-11 files (instead of using FILEX). A Jack Of All Trades (JOAT) utility supplies some new functionality required in the RTEM environment. An RSX like Indirect command line processor (IND) was included in RTEM (it is also available in RT-11 Version 5). The following layered products are supported: BASIC11/RT-11, FORTRAN IV/RT-11, single user DIBOL, APL/RT-11, and FMS/RT-11. The following RT-11 layered products are not supported:

DECnet/RT, MU/BASIC, 2780/3780 RT, Time Shared DIBOL, Lab/VT11/LV11 Packages, Labs/Apps.

If the virtual system device option is used the system is divided into two parts. A shared portion is called LB:[1,1]RT11SH.SYS. It is a read-only file with a read/write mark zone (for SWAP, BAM, IND, Handlers) This portion may contain up to 8 directory segments. File storage is variable size. Here reside RTEM-11 Monitor, Handlers, and Utilities. Unused entries are padded as protected "NON-ENTITIES" which occupy directory space, are protected, have zero length, do not appear in directory listings and do not appear as an <EMPTY> to .ENTER or Squeeze. The shared portion is always squeezed with zero free blocks.

The private portion of the virtual system device is in SY:[ppn]RT11PR.SYS and is read/write enabled. It contains up to 31 directory segments of which 1-8 are copies from the shared portion. The default creation size of file storage is 494 blocks (this may be changed by the system manager). Unused file storage space need not be allocated. The Mark Area (copied from the shared file) contains SWAP.SYS, BAM.SYS, IND.SAV and some handlers that use SET commands.

The non-system virtual devices (VS: for some reason) allows up to eight devices exclusive of the system device. JOAT initiates access to the VS0: to VS7: devices. It is intended to be used for users with several virtual system devices for accessing other users' virtual system devices. It treats a virtual system device as a data server by mapping only the private part of a VSD as read/write.

The RT-11 clock routine is never entered. RT-11 MRKT/CMKT are converted to the RSX equivalents. DATE/TIME are decoupled from clock ticks and are thus obtained from the host. DAY/MONTH roll over is thus provided by RSX automatically. There is no time loss during checkpointing.

Restricted Programmed Requests

.CMKT	does not return number of ticks remaining (always 0)
.CNTXSW	must not reference the I/O page
.DEVICE	must not reference the I/O page
.DRxxx	handler requests
.MTxxx	can only reference the console terminal
.MTSET	hardware characteristics can not be set
.PROTECT	not useful since program can't access I/O devices
.SPFUN	only if function is available through the host driver
.UNPROTECT	same as .PROTECT

The following programmed requests may not be used: .CTIMIO, .SDTTM, .TIMIO and any XM requests.

The LP: is the only supported printer handler. It is a special device that processes .ENTER, .WRITE and .CLOSE. The .ENTER command opens a FILES-11 file, .WRITE writes to the file and .CLOSE spools the file to the printer.

FIP runs under RTEM-11 to transfer files between RT-11 and FILES-11 volumes. It performs formatted ASCII, FORTRAN carriage control and Image Mode transformations. It does not allow file deletion operations, directory listings or volume initialization operations.

JACK-OF-ALL-TRADES attaches and releases devices from host files and performs some general utility functions. JOAT passes files specifications for which FILES-11 files will be used as a virtual device. It can list which files correspond to which virtual devices. It can extend the VSD files when more space is needed and create virtual files. In addition to files handling JOAT passes command lines to the host system and is used to terminate RTEM-11 to return control to the host system.

You maw not use the followins utilities: ERROUT, FORMAT, MDUP, ODT (VDT maw be used instead). This is because these utilities access the I/O pase which is unavailable. Some utilities have restricted use. BOOT option (/O) in DUP is not available. In FILEX the TOPS-10 option (/T) maw not be used. The /V:n[m]] option in LINK is also out. RESORC confisuration options (/H and /Z) can not be used.

What RTEM-11 does not do. RT-11 user prosrams cannot access FILES-11 files. (You have to copw the file in with FIP.) There is no access to the I/O pase. RTEM doesn't support downloadins of stand-alone RT-11 swstems. The followins devices are not supported: Cassette (TU60), Card Reader, Papertape Reader/Punch, VT11/VS60 displaw, PDT handler (PD), Error Lossins Handler, RF11 Fixed head Disk. The lawered products that are unsupported were listed in the Lawered Products section above.

In answer to questions the followins points were broušht out. KMON, USR etc look normal since the code is the same as in RT-11. RTEM does have floatins point support but it is slow. One user found two busš in

RTEM: 1. It won't work on RK03 devices (this maw be fixed in the next version?). 2. It doesn't like "$" in labels in the IND labels. (This was fixed in the released version as the same problem occurred in RT-11 V5). The SJ monitor is not supported but there was sisnificant interest in the audience for it. There is no direct access to anw real devices. While this product was meant to provide a development environment it appears that a lot of the interest was in usins it for applications. RTEM is a lawered product for the VAX as well as for PDP-11s but a prerequisite is the ownership of an RT-11 license. The emulatins task provides up to 60k bwtes of space for the RT-11 swstem part but maw be set bw the swstem manaser to be as small as 24k(words?). Several users had trouble usins the installation suide and settins it to run but once it was up thew seemed quite happw with its performance.

RTEM-11 is product number QJ13 (plus media codes); the Users Guide is numbered AA-N435A-TK and the Installation Guide is AA-N436A-TK.

RT-11 Version 5 Overview

Jim Williams
Disital Equipment Corporation
Mawnard, MA

John Rasted, Session Chairperson
JTR Associates
Meriden, CT

Reported bw Gavin Perrw, DECUS Scribe Service

This article will explain what wou need to know about RT-11 V5 availabilitw and how to set it, what's new in version 5, what's the same as version 4.0, and what's different from version 4.0. At the end of the article is a potpourri of questions and answers from the session that maw help in evaluatins when and if certain unavailable features maw appear.

Version 5 for RT-11 has been released as of April 28, 1983. Supported users should have received their copies by the end of May. If you have RT-11 but are unsupported just buy the H kit, you do not need a new license. New users will need to buy the license, documentation, and support as needed. There are several options for types of licenses for the various CPUs, general licenses for OEMs etc. The A kit is no longer available but all the components (unbundled) that were in the kit are available.

Version 5 will be compatible with version 4.0, and it will be easier to install and use. V5 rounds out existing functionality without any large changes. V5 will support new DIGITAL CPUs and devices. There are some enhancements to the XM system.

Version 5 is to be customer installed. This savings on installation is reflected in the lower price of RT. A new update service replaces Autopatch. Modular replacement will be provided instead of patches for ease of maintenance. All corrections will be cumulative. New features may be distributed by this mechanism. The distribution will be remastered at each

update so that new purchasers will get a version ready to install without patching beforehand. Self-maintenance service will be available, including telephone service, machine readable patches, and the above Update service. Full DIGITAL support service is also available including on-site assistance and specialist support if required. The software dispatch review will still be available for all levels of support and may be purchased by unsupported sites.

The PDT-150 is supported but the PDT-130 is not. The T-11 is supported. UDA/MSC disks are supported under RT-11, but there is a maximum of 8x64k blocks on the drive, and the maximum file size is 64k blocks since a large device is treated as up to 8 smaller devices with a maximum of 64k blocks each. Under XM there are the following enhancements: Q-Bus 22-bit addressing is supported, up to 8 128kw jobs may be run at once, and the executive is more efficient. It can now run virtual jobs and privileged foreground jobs with virtual overlays.

A handler has been added to allow the editing of command lines with a subset of the KED keypad editor commands. It is even possible to save frequently used command lines and reissue the command with a few key strokes. V5 supports logical disks with the MOUNT and DISMOUNT LDn:dd:filename.DSK command. Logical assignments are maintained across bootstraps and SQUEEZEs. This handler is integrated with the system and has features that will avoid those nasty problems that can occur with the DECUS XD: or AR: and other layered subdevice handlers. When the file containing a device is moved the handler will know about it. A Virtual Memory disk handler is included which will support 22-bit addressing. This allows the user to partition a section of memory to look like a very fast disk. The VM device can even be the system disk under SJ or FB; it is not currently bootable under XM.

User programs can be used as KMON commands in two ways.

1. CCL can trap commands that fall through DCL and it will run a program named SY:commandname.SAV. If the command line has parameters after the command these will be passed into the program. If the parameters look like DCL format (e.g. separated by spaces) then the order will be switched so watch out. DEC considers this a feature not a bug.

2. If the CCL can't find SY:file.SAV the UCL option may be included at SYSGEN time. At this point a file called UCL.SAV will be run and the whole command line will be passed to a user written command parser/interpreter.

New SYSGEN options include:

1. Direct question for including high speed ring buffer.
2. Set DZ11 up to 9600 baud (versus 300 for V4).
3. Remote console option.
4. An answer file is created during SYSGEN like in RSX and thus previous responses can easily be changed. V5 is supplied with an answer file with default answers; thus making a few minor changes will not be difficult. Previously created answer files can be saved, making it easy to document your configuration.
5. One can SET EXIT NOSWAP to keep handlers from swapping out upon completion of a job,again improving performance of TU and floppy devices.
6. IND control/command file structure can be included to allow parameters and conditionals in a command file. (discussed below)

Error logging support for SJ has been added. One can optionally select no logging of successful I/O to improve performance of floppy and TU58 devices.

Write protect support has been implemented in the software for DX and DY devices. NOWRITE can not be used for the system device since it needs to write the modified driver back to SY:. Variable retry count can be implemented for the following devices: DX DY DD DL DM RK.

New Programmed Requests include the following:

.FPROT set/reset file protect status
.SFDAT set file date
.ABTIO abort outstanding I/O for this job/channel for SJ
.FETCH available for XM (needn't have all handlers loaded)
.PEEK to look into the monitor
.POKE to change/crash the monitor
.PVAL to modify monitor data
.GETLIN can be set to take input only from the terminal

and not from command files.

The IND processor is a superset of the RSX IND processor. It allows string substitution, file operations, conditional execution, etc. With SET KMON [NO]IND this feature can be switched in or out.

Improved utilities features include:

PIP /MULTIVOLUME copies from a large volume to several small. volumes. Incremental restore is possible, as well as complete restoration of the large volume. /VERIFY checks data after copy. /[NO]PROTECT default is now to leave protection as found in the original. /SINCE /BEFORE work to move a subset of files based on date. /DATE to move files created on a specific date.

DIR /[NO]PROTECT list [un]protected files

DUP can optionally retain or replace the bad blocks information.

LINK global cross reference map, virtual overlays allowed with privileged FG job, routines allowed in multiple overlays (without being pulled to root).

MACRO completes assembly if CREF fails, error messages are compatible with the documentation.

FILEX /WAIT option.

SLP has checksums for source patches.

BINCOM allows wildcard comparisons and device to device.

HELP pages help text output to video terminals (if SET TT SCOPE).

QUEMAN /LOG option, /SINCE:dd:mmm:yy, /DATE:dd:mmm:yy

QUEUE protects file while despooling, file by file output to RT-11 file structured devices.

FORMAT formatting allowed while FG active.

A new utility, BUP, allows backing up of large files to multiple volumes of smaller media. Unsupported new utilities will include: TTYSET set terminal characteristics on multiterminal systems, MTYSET for auto-baud terminals (multiterminal systems), and RTMON system activity monitor and display (system job support only). SPEED is still available and BLET, the single line editor, allows substitution of a string for a character. E.G. could be used to stand for FORTRAN to easily compile FORTRAN programs with MYPROG. The underscore in front of the character will prevent it from being substituted for in the command line.

The System Users Guide has been split into two volumes. (What was chapter 4, the Keyboard commands, is now a separate volume from the Utilities Manual.) The customer installation guides are configuration specific which will simplify customer installation. A Mini-Reference replaces the Pocket Guide; it won't fit in your pocket but it contains enough information to be a useful daily reference volume that won't take up the whole desk.

Changes from Version 4 include the following notes:

PIP retains protection status unless switch is called.

COPY for .SYS files does not require /SYS if there are no wildcards in the input string.

DELETE defaults to NOCONFIRM if there are no wild cards in the input file specification.

SRUN defaults to .REL instead of .SYS extension.

QUEUE work file is now SY:QUFILE.WRK.

User written handlers will work if they were written using the .DRDEF call otherwise they won't work. ...CM0 and ...CM1 are different. Their internals were undocumented anyway and shouldn't be called. There is a way around this. If you used them and if you can't figure it out, contact Jim Williams.

Questions and Answers Sessions:

Q. Can character matching across case be done?

A. Yes, that is now the default. ENABLE/DISABLE LCM will change whether it is case sensitive or not.

Q. Can a bootable logical disk be moved to a different device and still be booted?

A. Perhaps not. Submit an SPR.

Q. Will compiled BASIC or FORTRAN-77 become available?

A. It's possible (but you will have to really want it).

Q. Does the Backup and Restore Utility have to do a bad blocks scan automatically before copying?

A. Yes, we use DEC disks

Q. Why can't we use 22-bit addressing on a 11/23?

A. There is no way to tell if the device is a REV A or a REV C device. REV A's would crash. If you know you have a REV C you can patch around this restriction.

Q. Will the backup utility work on version 4?

A. No. Besides you have to get version 5 to get the utility so use it.

SYMPOSIUM TAPE INFORMATION

Here is the annotated directory for the RT-11 SIG tape from the St. Louis DECUS. We were unable to build it on-site, so I did it back home.

Please publish the directory in the newsletter. I have sent a tape to Joe Lachman, who is the head of the tree, and he will distribute it from there down the branches. Please mention in the newsletter that people who want copies of the tape should contact their local LUG librarians, not me or Joe directly.

Thanks,

Rally Barnard
Sandia Labs
Albuquerque, NM 87185
(505) 844-5115

```
*        Spring, 83 DECUS Symposium RT-11 Tape                     *
*                                                                  *
*                        St. Louis                                 *
*                                                                  *
*                   Annotated Directory                            *
*                                                                  *
********************************************************************
*                                                                  *
*                         IMPORTANT                                *
*                                                                  *
*        Read the file, README.1ST, first.                         *
*                                                                  *
********************************************************************
```


David Stagg
Dept of Pharmacology
Yale University Medical School
333 Cedar Street
New Haven
Ct 06510
203-436-2151

These files are taken from the Fall, 81, RT-11 tape.

This is an implementation of the virtual-device driver described by Crapuchettes on page 639 of the DECUS proceedings Fall 1980.

```
README.1ST     7  05-Apr-83  SIG tape copy instructions
XD    .SYS     2  05-May-81  SJ/FB driver
XDATCH.SAV    21  17-Apr-81  XDATCH utility
DUP   .SXD    41  21-Sep-81  DUP V4.0K patched for XD
XD    .DOC    10  16-Nov-81  Description document
XDX   .SYS     2  05-Apr-81  XM driver
DUP   .COM     1  16-Mar-82  Patch file for DUP.SAV

XD    .DEV    55  06-Apr-83  VIRTUAL DEVICE containing
                              the following files:
   XDATCH.FOR     9  13-Nov-81  XDATCH utility source
   GETFIL.MAC    10  05-Nov-81  XDATCH subroutine
   FTRAN .MAC     3  17-Apr-81  XDATCH subroutine
   TRIMS .MAC     1  17-Apr-81  XDATCH subroutine
   ASLOOK.MAC     6  17-Apr-81  XDATCH subroutine
   XD    .MAC     5  08-May-81  Virtual-device driver source
   MMGT  .MAC     1  08-Dec-82  XDATCH subroutine
   XD    .COM     1  01-May-80  Virtual-device build file
   XDATCH.COM     1  17-Apr-81  XDATCH utility command file
   DUP   .COM     1  16-Mar-82  Patch file for DUP.SAV
   XD    .RNO     8  16-Nov-81  Description RUNOFF source
    11 Files, 46 Blocks
```

****

DIR1.DEV - Annotated tape directories, part 1.
DIR2.DEV - Annotated tape directories, part 2.

N. A. Bourgeois, Jr.
Sandia National Laboratories
Division 9238
P. O. Box 5800
Albuquerque, NM 87185
(505) 844-8088

R. W. Barnard
Sandia National Laboratories
Division 2565A
P. O. Box 5800
Albuquerque, NM 87185
(505) 844-5115

Annotated directories of the DECUS Symposia RT-11 tapes from the Spring of 78 through the Spring of 83 (this symposium).

S78	.DIR	29	14-Apr-82	F78	.DIR	27	21-Dec-81
S79	.DIR	37	21-Dec-81	F79	.DIR	92	21-Dec-81
S80	.DIR	41	21-Dec-81	F80	.DIR	102	21-Dec-81
S81	.DIR	158	21-Dec-81				

7 Files, 486 Blocks

F81	.DIR	55	21-Dec-81	S82	.DIR	28	08-Dec-82
F82	.DIR	18	05-Apr-83	S83	.DIR	44	01-Jun-83

4 Files, 145 Blocks

****

INFOTS.DEV - Initialize FORTRAN's OTS.

N. A. Bourgeois, Jr.
Sandia National Laboratories
Division 9238
P. O. Box 5800
Albuquerque, NM 87185
(505) 844-8088

This submission shows how to initialize FORTRAN's OTS from a program or subroutine written in MACRO-11.

INFOTS - Demonstrates FORTRAN's OTS initialization

NAB - Compares the executable and map files generated.

NABFOR - A do-nothing program that was used to study FORTRAN's OTS initialization process

NABMAC - A MACRO-11 program that generates executable code identical to that generated from NAB.FOR.

INFOTS.COM	1	08-Feb-83	NABFOR.FOR	1	07-Feb-83
INFOTS.DOC	5	09-Feb-83	NABFOR.LST	1	09-Feb-83
INFOTS.LST	15	09-Feb-83	NABFOR.MAP	4	09-Feb-83
INFOTS.MAC	8	08-Feb-83	NABFOR.OBJ	5	09-Feb-83
INFOTS.MAP	5	09-Feb-83	NABFOR.SAV	10	09-Feb-83
INFOTS.OBJ	2	09-Feb-83	NABMAC.LST	196	09-Feb-83
INFOTS.RNO	5	07-Feb-83	NABMAC.MAC	99	08-Feb-83
INFOTS.SAV	15	09-Feb-83	NABMAC.MAP	4	09-Feb-83
JUNK .FOR	1	03-Feb-83	NABMAC.OBJ	15	09-Feb-83
JUNK .LST	1	09-Feb-83	NABMAC.SAV	10	09-Feb-83
JUNK .OBJ	5	09-Feb-83	PRINT .COM	1	09-Feb-83
NAB .COM	2	08-Feb-83			

23 Files, 411 Blocks

****

TSXLIB.DEV - TSX system services library paper.

N. A. Bourgeois, Jr.
Sandia National Laboratories
Division 9238
P. O. Box 5800
Albuquerque, NM 87185
(505) 844-8088

This is a copy of the paper presented at DECUS, including vu-graphs.

ABSTRA.DOC	2	16-Dec-82	VUG007.RNO	1	03-Mar-83
ABSTRA.RNO	3	16-Dec-82	VUG008.DOC	1	03-Mar-83
SHNOTE.DOC	6	15-Dec-82	VUG008.RNO	2	03-Mar-83
SHNOTE.RNO	6	15-Dec-82	VUG009.DOC	2	03-Mar-83
TSXLIB.DOC	42	02-Mar-83	VUG009.RNO	3	03-Mar-83
TSXLIB.RNO	37	02-Mar-83	VUG010.DOC	1	03-Mar-83
VUG000.RNO	1	03-Mar-83	VUG010.RNO	1	03-Mar-83
VUG001.DOC	1	03-Mar-83	VUG011.DOC	1	03-Mar-83
VUG001.RNO	1	03-Mar-83	VUG011.RNO	2	03-Mar-83
VUG002.DOC	2	03-Mar-83	VUG012.DOC	2	03-Mar-83
VUG002.RNO	2	03-Mar-83	VUG012.RNO	2	03-Mar-83
VUG003.DOC	1	03-Mar-83	VUG013.DOC	1	03-Mar-83
VUG003.RNO	1	03-Mar-83	VUG013.RNO	1	03-Mar-83
VUG004.DOC	1	03-Mar-83	VUG014.DOC	2	03-Mar-83
VUG004.RNO	1	03-Mar-83	VUG014.RNO	2	03-Mar-83
VUG005.DOC	1	03-Mar-83	VUG015.DOC	2	03-Mar-83
VUG005.RNO	1	03-Mar-83	VUG015.RNO	2	03-Mar-83
VUG006.DOC	1	03-Mar-83	VUG016.DOC	1	03-Mar-83
VUG006.RNO	1	03-Mar-83	VUG016.RNO	1	03-Mar-83
VUG007.DOC	1	03-Mar-83			

39 Files, 142 Blocks

****

BENCHM.DEV - Fortran Benchmark Routines.

Ron Trellue
Sandia National Laboratories
Division 7523
P. O. Box 5800
Albuquerque, NM 87185
(505) 844-0955

The files in this submittal are the benchmark routines used for the Fortran/RT Tutorial - EIS, FIS, and FPU, which was given at this Symposium. Other test routines which were not discussed are also included.

APHORI.DAT	35	17-Jun-82	CCA8 .COM	1	22-Jul-82
BENBYS.FOR	6	18-Jun-82	CCA8 .FOR	6	30-Jul-82
BENBYS.PAS	7	22-Jun-82	CCA9 .COM	1	07-Sep-82
BENBYX.PAS	7	22-Jun-82	CCA9 .FOR	5	02-Sep-82
BENCH .FOR	14	07-Jan-80	INPUT .FOR	2	02-Sep-82
BENVIR.FOR	14	08-Jan-80	JACOBI.FOR	24	30-Jul-82
CCA1 .COM	1	07-Sep-82	LITTL .COM	1	30-Jul-82
CCA1 .FOR	16	02-Sep-82	LITTL .FOR	2	30-Jul-82
CCA1X .FOR	36	15-Jul-82	NSC2 .FOR	6	24-Jul-78
CCA10 .COM	1	07-Sep-82	RAV3 .FOR	2	24-Jul-78
CCA10 .FOR	6	02-Sep-82	RAV5 .FOR	2	24-Jul-78
CCA11 .COM	1	30-Jul-82	RCCALL.COM	2	12-Apr-83
CCA11 .FOR	10	30-Jul-82	RCCA1 .COM	1	16-Nov-82
CCA2 .COM	1	07-Sep-82	RCCA10.COM	1	13-Sep-82
CCA2 .FOR	16	07-Sep-82	RCCA2 .COM	1	07-Sep-82
CCA3 .COM	1	22-Jul-82	RCCA5 .COM	1	13-Sep-82
CCA3 .FOR	8	22-Jul-82	RCCA9 .COM	1	13-Sep-82
CCA4 .COM	1	22-Jul-82	README.CCA	3	17-May-83
CCA4 .FOR	7	22-Jul-82	SCRIP .COM	1	30-Jul-82
CCA5 .COM	1	07-Sep-82	SCRIP .FOR	5	30-Jul-82

CCA5	.FOR	5	02-Sep-82	SCRIP2.FOR		4	24-Jul-78
CCA6	.COM	1	22-Jul-82	SCRIP3.FOR		4	24-Jul-78
CCA6	.FOR	5	22-Jul-82	SP1111.FOR		112	24-Jul-78
CCA7	.COM	1	22-Jul-82	TIME	.FOR	4	24-Jul-78
CCA7	.FOR	5	22-Jul-82				

49 Files, 398 Blocks

****

DSKLIB.DEV - Librarian for multiple disks.

Joel Berez
Infocom
64 Jacqueline Rd.
Pittsburgh, PA 15217
(617) 492-1031

DSKLIB, DLB*, UA - A disk librarian program to maintain a master catalog of multiple disk directories.

AR, ARCDEF - A sub-device handler.

Details of these programs are found in the file ANOTAT.DIR.

ANOTAT.DIR		6	21-May-83	DSKLIB.DOC		124	08-May-83
AR	.DOC	12	25-Nov-81	DSKLIB.EOB		60	18-May-83
AR	.MAC	7	27-Nov-81	DSKLIB.ESV		28	18-May-83
ARCDEF.MAC		4	27-Nov-81	DSKLIB.NOB		62	18-May-83
ARCDEF.SAV		3	27-May-81	DSKLIB.NSV		29	18-May-83
DLBCOM.MAC		1	08-May-83	UA	.MAC	1	08-May-83
DLBCOM.SAV		1	15-Mar-79	UA	.SAV	1	18-May-83
DLBOPT.MAC		4	26-Dec-81				

15 Files, 343 Blocks

****

GETRSX.DEV - Transfer FILES-11 files to RT-11.

Mark Bartelt
HSC Research Development Corporation
555 University Avenue
Toronto, ON M5G1X8
(416) 597-1500, X4588

GETRSX, CLOSE, CSTAT, CVTR50, EMT*, GETCHN, LOOKUP, READW, R50FIL, STR*, TYPES, UNXSYS - This submission contains an RT-11 program for reading ODS-1 (RSX) disks. The file GETRSX.DOC contains the UNIX manual pages for `getrsx'.

Bryan Kattwinkel
Kattwinkel Komputing
520 Palm Springs Blvd, #502
Indian Harbour Beach, FL 32937
(305) 773-3518

README.1ST, RTRSX, RSXGET, RSX*, DOPEN - An RT-11 program to read files and directories from an RSX-11 (ODS1) filesystem.

CLOSE .C	1	12-Oct-82	README.1ST	2	21-May-83
CSTAT .C	1	13-Oct-82	READW .C	1	12-Oct-82
CSTAT .H	1	13-Oct-82	RSXGET.C	38	07-May-82
CVTR50.C	6	22-Oct-82	RSXGET.DOC	6	07-May-82
DOPEN .MAC	2	06-May-83	RSXSUB.C	27	19-May-83
EMT .MAC	5	13-Oct-82	RTRSX .C	20	21-May-83
EMTDEF.H	3	13-Oct-82	RTRSX .DOC	7	19-May-83
GETCHN.C	1	14-Oct-82	RTRSX .SAV	29	21-May-83
GETRSX.COM	1	19-May-83	R50FIL.H	1	13-Oct-82
GETRSX.DOC	8	19-May-83	STRCAT.C	1	19-May-83
GETRSX.SAV	31	19-May-83	STRCHR.C	1	19-May-83
GETRSX.SRC	45	20-May-83	TYPES .H	1	13-Oct-82
GETRSX.TXT	8	20-May-83	UNXSYS.C	3	09-Nov-82
LOOKUP.C	1	22-Oct-82			

27 Files, 251 Blocks

****

22BIT.DEV - Handlers for 22-bit systems.

R. L. Hicksted
Micro Technology Inc.
1620 Miraloma Ave.
Placentia, CA 92670
(213) 632-7580

MXV22, RTDY, README.MTI - Modification to DY driver for 22-bit addressing for use with MTI MXV22 controller.

Dale J. Travis
Saturn Systems
6875 Washington Ave. S.
Minneapolis, MN 55345

DY*, DM* - Patch for the DY handler to support 22-bit addressing of the LSI-11/23 using the Micro Technology MXV-22 floppy disk controller with TSX+ V3 and later. A patch for the DM handler to support 22-bit addressing of the LSI-11/23 using the Emulex SC02/C disk controller with TSX+ V3 and later.

VM - The DECUS VM handler modified so it won't clear all of memory when it sizes itself.

MODEM7, TSX*, CIO - A program running under RT-11 that would be compatible with the MODEM7 from the CP/M world.

CIO .MAC	2	19-Dec-82	MODEM7.SAV	25	19-Jan-83
CIO .OBJ	1	19-Dec-82	MXV22 .DOC	283	01-May-83
DM .DOC	2	18-May-83	README.MTI	2	23-May-83
DM22V4.SLP	3	04-Mar-83	RTDY .COM	3	25-Mar-83
DY .DOC	2	18-May-83	TSXIS .MAC	1	12-Mar-83
DY22V4.COM	1	17-May-83	TSXIS .OBJ	1	12-Mar-83
DY22V4.SLP	2	17-May-83	TSXRT .MAC	1	12-Jan-83

MODEM7.C	53	12-Mar-83	TSXRT .OBJ	1	12-Jan-83
MODEM7.COM	1	06-May-83	VM .DOC	1	14-May-83
MODEM7.DOC	5	14-May-83	VM .MAC	21	14-May-83

20 Files, 411 Blocks

****

APL1.DEV - APL-11, Version 1 for RT-11.
APL2.DEV

Doug Bohrer
First National Bank of Chicago
One First National Plaza, Ste 0005 (1-22)
Chicago, IL 60670
(312) 732-2666

APLO* - The various APL compilers available, depending on the hardware configuration.

*.APL - APL utilities for calculating, printing, and regression analysis.

CHAREX, FSALEN - FORTRAN utilities for converting and manipulating files for APL format.

Details of these programs are found in the file APLUTL.DOC on disk APL2.

APL00 .SAV	100	31-May-83	APL03 .SAV	95	31-May-83
APL01 .SAV	100	31-May-83	APL04 .SAV	95	31-May-83
APL02 .SAV	95	31-may-83			

5 Files, 485 Blocks

APLUTL.DOC	42	31-May-83	FSALEN.FOR	7	31-May-83
APL06 .SAV	83	31-May-83	FSALEN.HLP	4	31-May-83
APL07 .SAV	83	31-May-83	FSALEN.SAV	21	31-May-83
CHAREX.FOR	8	31-May-83	REGRES.APL	7	31-May-83
CHAREX.HLP	4	31-May-83	UTLCAL.APL	10	31-May-83
CHAREX.SAV	27	31-May-83	UTLPRT.APL	4	31-May-83
FSALEN.C	4	31-May-83			

13 Files, 304 Blocks

****

UTIL1.DEV - System and Application Utilities.

R. W. Barnard
Sandia National Laboratories
Division 2565A
P. O. Box 5800
Albuquerque, NM 87185
(505) 844-5115

PARSE - A flexible filespec parser for FORTRAN programs.

SWCNSE- A console switcher routine which allows runtime setting of communications line speed (for interfaces such as DL11V-E or DLV-11F).

R. S. Frazer
Sandia National Laboratories
Division 7523
P. O. Box 5800
Albuquerque, NM 87185
(505) 844-7863

SETOPS - A program for checking which options in a handler are available, and which are or are not set.

RTCOM - Another communications package for making an RT system a smart terminal.

LX8BIT - A version of the LS handler with enhanced settable options and the ability to pass 8-bit data.

M. Alvin Levy
Levy & Adams
75 Marietta St. NW, Ste 400
Atlanta, GA 30303
(404) 521-2406

FACE, LIST, SCRIBE, README.LVY - These are programs that print a source file. FACE provides the ability to interactively set the top and bottom margins as well as the number of lines to be printed on a page. LIST is a program that will print any given part or all of a file. SCRIBE is a program that will print all of one or more files.

TCU*, README.TCU - Programs to read or write system date/time to Digital Pathways TCU-50D clock.

Thomas J. Shinal
General Scientific Corp
1683 E. Gude Dr.
Rockville MD 20850
(301) 340-2773

GETRNO, GETGLO, HEADER, OUTRNO, STRING - This program extracts flagged comment fields from a source file and sends them to a RUNOFF source file.

SHUT, DELAY* - TSX shutdown program.

DELAY1.MAC	1	22-Oct-82	PARSE .DOC	3	29-Apr-83
DELAY1.OBJ	1	22-Oct-82	PARSE .FOR	5	08-Apr-83
DELAY1.SAV	2	22-Oct-82	PARSE .FTN	8	08-Apr-83
DELAY5.MAC	1	22-Oct-82	PARSE .OBJ	16	17-May-83
DELAY5.OBJ	1	22-Oct-82	README.LIS	6	19-May-83
DELAY5.SAV	2	22-Oct-82	README.TCU	2	21-May-83
EXAMPL.PAR	3	08-Apr-83	RTCOM .DOC	18	03-Feb-83
FACE .PAS	4	14-May-83	RTCOM .SAV	16	07-Oct-82
FACE .SAV	14	17-May-83	RTSET .FOR	17	18-Feb-83

GETGLO.PAS	1	10-Sep-82	RTSET .SAV	30	18-Feb-83
GETRNO.COM	1	10-Sep-82	SCRIBE.PAS	10	17-May-83
GETRNO.DOC	17	10-Sep-82	SCRIBE.SAV	19	17-May-83
GETRNO.MAP	4	01-Dec-82	SETOPS.DOC	1	17-May-83
GETRNO.OBJ	10	01-Dec-82	SETOPS.FOR	15	02-Mar-83
GETRNO.PAS	21	02-Dec-82	SETOPS.SAV	35	02-Mar-83
GETRNO.RNO	15	01-Dec-82	SHUT .COM	2	25-Oct-82
GETRNO.SAV	25	01-Dec-82	SHUT .DOC	1	04-May-83
GETRNO.SRC	1	10-Sep-82	STRING.DEC	2	10-Sep-82
HEADER.RNO	1	10-Sep-82	STRING.OBJ	7	01-Dec-82
LIST .PAS	19	17-May-83	SWCNSE.DOC	2	29-Apr-83
LIST .SAV	21	17-May-83	SWCNSE.MAC	5	08-Apr-83
LX8BIT.DOC	4	02-Mar-82	SWCNSE.SAV	2	28-Oct-81
LX8BIT.SAV	2	02-Mar-82	TCUGET.MAC	7	21-May-83
OUTRNO.OBJ	13	01-Dec-82	TCUPUT.MAC	6	21-May-83
OUTRNO.PAS	13	02-Dec-82			

49 Files, 432 Blocks

****

UTIL2.DEV - More utilities.

Joel Berez
Infocom
64 Jacqueline Rd.
Pittsburgh, PA 15217
(617) 492-1031

DDT - (Dynamic Debugging Technique) allows symbolic debugging of programs using the symbol table file optionally produced by LINK.

VIEW, V - View is a small utility program that allows quickly paging through an ASCII file on a VT100.

BYE, BYEBLD - Bye prints a saying or definition.

COPY, MOUNT - A TECO macro to backup a large disk to multiple smaller disks.

Details of these programs are found in the file ANOTAT.DIR.

ANOTAT.DIR	6	21-May-83	DDT .DOC	13	14-Sep-82
BYE .IDX	4	17-May-81	DDT .MAC	130	16-Oct-82
BYE .LNS	123	17-May-81	DDT .OBJ	26	08-May-83
BYE .SAV	3	19-Aug-82	MOUNT .SAV	2	30-Nov-78
BYEBLD.SAV	4	17-May-81	V .COM	1	08-May-83
COPY .DOC	5	07-Mar-83	VIEW .MAC	36	08-Jul-81
COPY .TEC	2	30-Nov-78	VIEW .SAV	6	08-May-83

14 Files, 361 Blocks

****

UTIL3.DEV - FORTRAN and C utilities.

Doug Bohrer
First National Bank of Chicago
One First National Plaza, Ste 0005 (1-22)
Chicago, IL 60670
(312) 732-2666

FIXLEN - A prosram to convert a FSA file to a XSA file.

MATCH, MATCHB - Prosrams for FSA file comparison. Matched records are written to the match file, those that don't are written to error output.

MERGEF - Mersins two previously-sorted FSA files.

CRREAD - Shared file readins and writins.

Details of these prosrams are found in the file FORUTL.DOC.

CRREAD.FOR	12	31-May-83	MATCH .SAV	28	31-May-83
FIXLEN.C	2	31-May-83	MATCHB.FOR	22	31-May-83
FIXLEN.FOR	10	31-May-83	MATCHB.HLP	7	31-May-83
FIXLEN.HLP	3	31-May-83	MATCHB.SAV	29	31-May-83
FIXLEN.SAV	29	31-May-83	MERGEF.C	11	31-May-83
FORUTL.DOC	21	31-May-83	MERGEF.HLP	11	31-May-83
MATCH .FOR	20	31-May-83	MERGEF.SAV	36	31-May-83
MATCH .HLP	7	31-May-83			

15 Files, 248 Blocks

****

VTUTL1.DEV - Variety of Utility Files for the VT-100.
VTUTL2.DEV

Stephen Cribbs
Atomic Energy of Canada, Ltd.
Pinawa, Manitoba R0E1L0
(204) 653-2311

MJCLOK - Foresround Job which displays calendar and clock on a VT-100.

TXTWRT - A text formattins prosram to set displays on a VT-100 screen.

TCFL - A VT-100 screen formattins packase.

ENC .FOR	15	11-May-83	PTXTW .MAC	2	11-May-83
EXMPL1.INP	1	11-May-83	README.1ST	6	18-May-83
EXMPL2.INP	1	11-May-83	TCFL .BFO	7	11-May-83
IDSPTC.FOR	30	11-May-83	TCFLIO.MAC	23	11-May-83
IDSPTC.MAC	7	11-May-83	TCFLO .MAC	27	11-May-83
IPARSE.FOR	6	11-May-83	TXTC .COM	1	11-May-83
KMP2WU.MAC	2	11-May-83	TXTL .COM	1	11-May-83
MATCH .FOR	4	11-May-83	TXTLO .COM	1	11-May-83
MJCLOK.DOC	41	12-May-83	TXTWRT.FOR	25	11-May-83
MJCLOK.MAC	37	12-May-83	TXTWRT.HLP	51	11-May-83
MJCLOK.REL	6	12-May-83	TXTWRT.OVL	53	11-May-83
NEST .FOR	6	11-May-83	TXTWRT.SAV	51	11-May-83

24 Files, 404 Blocks

CURP .FOR	2	19-May-83		SLVPLX.SAV	32	19-May-83
CURP .SAV	16	19-May-83		SLVPLY.FOR	32	19-May-83
GTTRM .FOR	1	19-May-83		SLVPLY.SAV	36	19-May-83
KTCC .FOR	6	19-May-83		TCFL .BFO	7	19-May-83
KTCC .SAV	19	19-May-83		TCFL .CSL	10	19-May-83
KTRMID.FOR	2	19-May-83		TCFL .DOC	98	20-May-83
KTRMID.SAV	14	19-May-83		TCFL .MTT	12	19-May-83
PTCFL .MAC	2	19-May-83		TCFLIO.MAC	24	19-May-83
README.2ND	4	19-May-83		TCFLO .MAC	27	19-May-83
SLVPLM.FOR	32	19-May-83		TCFLU1.MAC	8	19-May-83
SLVPLM.SAV	40	19-May-83		TCFLU2.MAC	21	19-May-83
SLVPLX.FOR	30	19-May-83		TID .NTE	2	19-May-83

24 Files, 477 Blocks

****

MTUTL1.DEV - Magnetic tape utilities.
MTUTL2.DEV

Doug Bohrer
First National Bank of Chicago
One First National Plaza, Ste 0005 (1-22)
Chicago, IL 60670
(312) 732-2666

DUMPMT, SPDUMP, EBCDIC, EXTMT - Dump the contents of a magnetic tape to a terminal.

BCK*, RET*, BACKUP - A high-blocking-density magnetic tape backup system.

TAPER, TAPXXX - A program to create a Fortran Sequential ASCII data file from a magnetic tape file.

Details of these programs are found in the file FORUTL.DOC on disk MTUTL2.

BACKUP.HLP	23	31-May-83		DUMPMT.HLP	16	31-May-83
BCKDIR.C	7	31-May-83		DUMPMT.SAV	52	31-May-83
BCKDIR.SAV	53	31-May-83		RETRN .C	3	31-May-83
BCKTPE.C	19	31-May-83		RETRN .SAV	30	31-May-83
BCKTPE.SAV	75	31-May-83		RETTPE.C	11	31-May-83
BCKUP .C	8	31-May-83		RETTPE.SAV	53	31-May-83
BCKUP .SAV	32	31-May-83		SPDUMP.FOR	21	31-May-83
DUMPMT.FOR	15	31-May-83		SPDUMP.SAV	53	31-May-83

16 Files, 471 Blocks

EBCDIC.DAT	2	31-May-83		TAPER .FOR	22	31-May-83
EXTMT .DOC	4	31-May-83		TAPER .HLP	16	31-May-83
EXTMT .MAC	26	31-May-83		TAPER .SAV	53	31-May-83
EXTMT .OBJ	3	31-May-83		TAPXXX.FOR	27	31-May-83
FORUTL.DOC	21	31-May-83		TAPXXX.SAV	54	31-May-83

10 Files, 228 Blocks

****

NOTE: The files listed for the various entries are all the ones associated with the entries. RT11 explicit (*) and implicit (no extension) wildcarding has been used.

R. W. Barnard 1-June-1983.

RT-11 SIG

FUTURES SURVEY

?? RT-11/Pro350 ?? ?? RT-11/VAX ??

As new hardware is announced by DEC, there are always questions concernint the use and support of RT-11 on that hardware. In an effort to define the RT-11 SIG's role in interacting with DEC on these questions, there will be a workshop session at the 1983 Fall DECUS U.S. Symposium in Las Vegas.

Although not necessarily restricted to the topics described here, the workshop will focus on two futures issues - RT-11 on the Professional PC, and RT-11 on 32-bit computers.

To help pre-load ammunition for this workshop (which may become a permanent symposium feature) the RT-11 SIG members are asked to answer the following survey questions.

****

1. Do you want/need RT-11 on the Professional 300 series computers? __Y __N

2. Would you be satisfied if RT-11 were offered only on the Pro 350? __Y __N

3. What features of the present Pro 325/350 would you want to see included or supported on a PRO/RT-11 operating system? (Note: I don't give you a menu. Go find out for yourself what the Pro's features are. If you're going to ask for RT-11 on the Pro, know what you're asking for.)

   ____
   ____
   ____
   ____

4. Because of the restrictive architecture of the Professional PC, some features of RT-11 may not be possible. What RT-11 features would you be willing to sacrifice in order to get it on the Pro 325/350?

   ___ SJ Monitor
   ___ Single Line Editor (SL)
   ___ VM Handler
   ___ Logical Disk Subsetting (LD)
   ___ Multi-terminal support
   ___ User written device handlers
   ___ User Command Linkage (UCL)
   ___ System Job Support
   ___ Device Timeout Support
   ___ Error Logging
   ___ BATCH Support
   ___ .SPCPS support (look it up!)
   ___ Asynchronous Terminal Status
   ___ .FETCH support in XM
   ___ Other ______________
   ___ Other ______________

5. It would be foolish to believe that DEC will not eventually make much smaller VAX-like 32-bit computers. What "VRT-11" features and enhancements would you expect to see in a single-user/multi-job operating system on a 32-bit workstation computer?

   ____
   ____
   ____
   ____

Please send responses to: Crow4ell, Ltd.*
145 Andanada
Los Alamos, NM 87544

Your DECUS Membership No.

____

* but not very

Place
Stamp
Here

Crowell, Ltd.*
145 Andanada
Los Alamos, NM 87544

RT-11 SIG Survey