THE mini-tasker

DECUS
RT-11 SIG NEWSLETTER

SEPTEMBER 1979 VOL. 5 NO. 4

Contributions to the newsletter should be sent to:

Ken Demers
MS-48
United Technologies Research Center
Silver Lane
East Hartford,Conn. 06108
203 727-7241

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
617 481-9511 Ext. 4141

FROM THE EDITOR

Please fill out the enclosed form on the last page of the newletter and return it to me. For those of you not planning to attend the December Symposium in San Diego, this is your only opportunity to give input on how our newsletter will be funded starting in fiscal year 1981.

As the questionnaire indicates, we are planning to make back issues of the newsletter availible for a small fee.

USER REQUESTS

Armstrong CORK COMPANY

AREA CODE 717 | TELEPHONE 397-0611
P.O. BOX 3511 LANCASTER, PA 17604

July 30, 1979

If anyone has a fast technique for running a statistical cross-correlation between two one-dimension vectors on a PDP11, please contact me.

J. W. Cluck
Engineering Department
Armstrong Cork Company
P. O. Box 3511
Lancaster, PA 17604
717-397-0611, Extension 7153

PAST SYMPOSIUM INFORMATION

The following people have volunteered to distribute the Symposium RT-11 magtape throughout Europe. The standard procedure is to send them a magtape and return postage. They will copy the symposium tape onto yours and return it to you. If you do not supply adequate return postage your magtape will be gladly accepted as a donation.

Contact:

Howard Schultens
Physiologisches Inst. 2
Humboldtallee 7
D-3400 Goettingen
W. Germany
Telex: 96703 Tel: 0551/395914

or

Ir. J. Loeve
Head Dept. CRW/ASV
Erasmus University Rotterdam
P.O. Box 1738
3000 Dr.
Rotterdam, Holland

CALIFORNIA INSTITUTE OF TECHNOLOGY

CHARLES C. LAURITSEN LABORATORY OF HIGH ENERGY PHYSICS
PASADENA, CALIFORNIA 91125

August 13, 1979

Ken Demers
MS-48
United Technologies Research Center
Silver Lane
East Hartford, Connecticut 06108

Dear Ken:

Enclosed is a reprint of a poster paper from the spring DECUS symposium, outlining how to implement a resident library under RT-11. I've had so many requests for copies that I've run out (twice).

Several people have suggested that I submit it to the Mini-Tasker, thereby enabling it to reach interested persons who weren't able to attend the DECUS meeting and have therefore not received copies of the proceedings. I'm also hoping it will reduce the amount of reprinting and mailing I'll have to do.*

The various files referred to in the paper (MAKELB.FOR, LBMAIN.MAC, and OTI replacement object modules for FORLIB) will (I hope) be included on the fall DECUS RT-11 SIG tape.

Sincerely yours,

Mark Bartelt

* You will find the poster paper in the back of this newsletter.

USER INPUT

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

Department of Physics

26-Jul-79

RE: Simple Bootstrap Loaders for RL01, RK05

Dear sir;

We have a PDP-11/10 with a 32 word diode ROM, type BM 792-YB, originally containing only the standard DEC bootstraps for RK05 disk and TU58 DECtape. After receiving our dual RL01, I devised the following bootstrap for drive Ø

(1) Unload drive Ø, wait for LOAD light

(2) Load drive Ø, wait for READY light

(3) Enter the following code
```
 772/ 12737  (MOV #14,@#DL$CSR)
 774/ 14
 776/ 174400
1000/ 777    (BR .)
```

(4) Start at location 772, the RUN light should flash and then go out as BR . is overlayed by the HALT in block 1. If RUN stays on, go back to step (1)

(5) Clear the (end-of-track) drive error by entering IN ORDER the following code
```
174404/ 13
174400/ 6
```

(6) Start execution at Ø, RT-11 should boot.

After an associate pointed out to me that the standard Digital BM 792-YB bootstrap of 32 words had six unused words, RL01 support was added. This was accomplished by changing location 173156 to 5007 (CLR PC) from 137 (JMP), thus increasing the number of unused words to 8.

These eight words were then used as the basis for the RLØ1 bootstrap on the BM 792-YB (see attached listing)

```
         173156/ 5007    (CLR PC)
START:   173160/ 12737
         173162/ 14
         173164/ 174400
         173166/ 12737
         173170/ 777
         173172/ 1032
         173174/ 12707
         173176/ 1032
```

Block 1 of the RLØ1 disk must also be patched as follows (this should present no problem for RT-11 users because the start of block 1 is apparently unused)

```
.R PATCH
*DL:
*1002/ 12700
*1004/ 174400
*1006/ 5710
*1010/ 100376
*1012/ 12737
*1014/ 13
*1016/ 174404
*1020/ 12710
*1022/ 6
*1024/ 105710
*1026/ 100376
*1030/ 5007
*1032/ 763
*E
.
```

Note that the monitor is <u>not</u> patched, hooking a new bootstrap on the disk will not alter this code. This means that the patch <u>NEED ONLY BE DONE ONCE</u> ! (for EACH disk)

Digital field service informed me of this trivial <u>4-word</u> bootstrap for RKØ5 drive Ø.

(1) Unload and RELOAD drive Ø

(2) Enter the following code (it does not matter where)

```
begin:  12737      (MOV  #5,@#RK$CSR)
        5
        177404
        1          (WAIT)
```

(3) Start at location 'begin' -- the READ light should flicker on drive Ø

(4) Press HALT, then start at location Ø.

(It is a good idea to write-lock the drive first!)

Sincerely,

T. Miles

T. Miles

```
.TITLE  DLKICK  V01.01

; TWO-PART BOOTSTRAP FOR RL01 DISK, for BM792-YB
;       T. MILES, DEPT. OF PHYSICS, UNIV. OF VICTORIA, CANADA, 25-JUL-79

; DISK REGISTERS

        DL$CSR=174400           ; RL01 CSR
        DL$DA=174404            ; RL01 ADDRESS REGISTER

; START OF DIODE BOOTSTRAP

        OFFSET=60               ; LENGTH OF (IMPROVED) DEC BM792-YB CODE
        ROM=173100+OFFSET       ; STANDARD LOCATION FOR 32 WORD DIODE R

        .ASECT

; LOW PART, GOES ON RL DISK

        .=1002                  ; WHERE ON DISK TO PATCH

KICK2:  MOV     #DL$CSR,R0      ; POINT TO CSR

1$:     TST     @R0             ; END-OF-TRACK ERROR ?
        BPL     1$              ; NOT YET

        MOV     #13,@#DL$DA     ; ASK FOR DEVICE RESET
        MOV     #6,@R0

2$:     TSTB    @R0             ; DONE ?
        BPL     2$

        CLR     PC              ; OFF TO REAL BOOT

FUDGE:  BR      KICK2           ; OVERLAYS BR .

; HIGH PART, GOES IN DIODE ROM
;       (NOTE, BM792-YB USERS CHANGE LOCATION 173156 TO 5007 [CLEAR PC]

        .=ROM

START:  MOV     #14,@#DL$CSR    ; ASK FOR READ

        MOV     #777,@#FUDGE    ; SETUP BR .
        MOV     #FUDGE,PC       ; AND GO THERE
```

CENTRE NATIONAL
DE LA RECHERCHE SCIENTIFIQUE

LABORATOIRE DE PHYSIOLOGIE
COMPAREE DES REGULATIONS
23, RUE DU LOESS
STRASBOURG-CRONENBOURG
TEL. (88) ~~XX.XX.XX~~ 29.90.33

B. P. 20 CR
67037 STRASBOURG - CEDEX

FRANCE

STRASBOURG, LE 18 juin 1979

Dear Mr. Demers,

I think that the readers of Mini-tasker would find an interest in this publication to help them in the redaction of articles.

D. GUINIER

$$
$ DECUSCOPE : DIGITAL EQUIPMENT COMPUTER USERS SOCIETY$
$$

APPLICATION NOTE : RT 11 (FORTRAN IV SOURCES.)
(AVAILABLE FROM AUTHORS).

****

TEXTE : A FORTRAN IV overlay program to automatically realize printed manuscripts with table of contents and indexes from any ASCII media.

****

BY D.GUINIER AND R.KIRSCH
LABORATOIRE DE PHYSIOLOGIE COMPAREE DES REGULATIONS
GROUPE DE LABORATOIRES DU CNRS DE STASBOURG-CRONENBOURG
23 RUE DU LOESS
B.P.20 CR
67037 STRASBOURG FRANCE

1. Introduction :

To facilitate the edition of printed matter such as books, manuscripts, thesis, courses, manuals like 'directions for use', etc., it is convenient to have an efficient and practical and easy to use program to realize the following operations :

-the formatting depending on the reproduction conditions : number of lines per page, margination, page numbering and shifting to obtain capital or small letters.

-the construction of the table of contents and the subject and author indexing.

2. Realization :

The input ASCII information is obtain from perforated cards or any file given by EDIT or equivalent, for exemple. All the operations are performed on three lines :

-the input line : BUFE.
-the residual line : RESIDU
-the current line : COUR

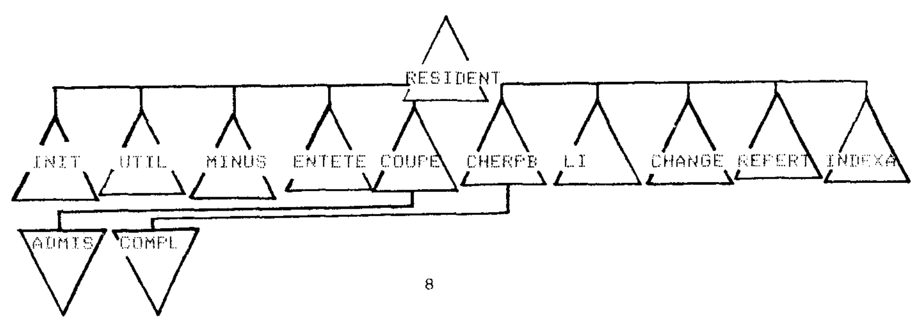
COUR is the result of the different operations on the successive BUFE encountered and RESIDU actualized. COUR is printed on the desired media (File to be retreated with EDIT or line printer directly) when it has satisfied all the required conditions.

2.1. Program organisation :

The program TEXTE is an overlay of 12 modules and a resident part written in FORTRAN IV langage which facilitate the transport of it on any computer with few modifications. It required a maximum length of about 1400 words.

- 1.) INIT : Initialize the input parameters.
- 2.) UTIL : Give the number of useful characters.
- 3.) MINUS : Treat the characters as capital or small letters.
- 4.) ENTETE : Perform the output of the page numbering with heading.
- 5.) COUPE : Perform the adjustment of RESIDU and COUR, the cutting must agree with NADMIS.
- 6.) ADMIS : Determine NADMIS, the number of admissible of BUFE or RESIDU in COUR.
- 7.) CHERPB : Determine the position of the ponctuation signs and spaces.
- 8.) COMPL : Add random spaces to the positions determined by CHERPB to perform an even right margin if desired.
- 9.) LI : Ajust the effects of <FF>,<TAB>,<CTRL/B>,' ' (spaces) encountered at the begining of an input line BUFE.
-10.) CHANGE : Execute the change of the output formatting directly indicated in the input file, if the first character read in BUFE is <CTRL/G>;the program returns to the initial formatting after a new encounter with <CTRL/G>.
-11.) REPERT : Construct the table of contents if the first character of BUFE is <TAB> and the second any number.
-12.) INDEXA : Construct the subject and author indexes.

2.2. Organisation diagram in tree :



3.1. Input parameters :

```
MAX.NUMBER OF CHARACTERS IN A LINE : 68
MAX.NUMBER OF LINES IN A PAGE      : 30
NO.OF THE FIRST PRINTED PAGE       : 1
NBR.OF INTERSPACES (0,1,ETC.)      : 0
NBR.OF HEADING LINES               : 5
DO YOU WANT AN EVEN RIGHT MARGIN ? : Y
LEFT MARGIN : NBR.OF CHARACTERS    : 8
INPUT ON                           : PUB. DAT
TABLE OF CONTENTS OUTPUT           : FTN1.DAT
SUBJECT INDEX OUTPUT               : FTN2.DAT
AUTHOR INDEX OUTPUT                : FTN3.DAT
OUTPUT ON                          : LP:
```

3.2. Conventions :

[...] : capital letters (prohibided if small letters in the input file.

\...\ : word to include in the subject index.

{...{ : word to include in the author index.

<FF> : automatic return to the next page.

<TAB> : automatic return to the next line.

' ' : shift left hand margin to the left, when encountered spaces at the begining of the read line BUFE.

'',^^, etc. : backspace to print an accent when encountered two successive accents if the line printer allows.

3.3. Output listings :

TABLE OF CONTENTS

The present publication represent this listings.

3.4. Input listings :

... 1. [I]NTRODUCTION :

[T]O FACILITATE THE EDITION OF PRINTED MATTER SUCH AS BOOKS, MANUSCRIPTS,THESIS,COURSES,MANUALS LIKE 'DIRECTIONS FOR USE',ETC., IT IS CONVENIENT TO HAVE AN EFFICIENT AND PRACTICAL AND EASY TO USE PROGRAM TO REALIZE THE FOLLOWING OPERATIONS :

-THE FORMATTING DEPENDING ON THE REPRODUCTION CONDITIONS : NUMBER OF LINES PER PAGE,MARGINATION,PAGE NUMBERING AND SHIFTING TO OBTAIN CAPITAL OR SMALL LETTERS.

-THE CONSTRUCTION OF THE TABLE OF CONTENTS AND THE SUBJECT AND AUTHOR INDEXING.

2. [R]EALIZATION :

[T]HE INPUT [ASCII] INFORMATION IS OBTAIN FROM PERFORATED CARDS OR ANY FILE GIVEN BY [EDIT] OR EQUIVALENT,FOR EXEMPLE.[A]LL THE OPERATIONS ARE PERFORMED ON THREE LINES :

-THE INPUT LINE : [BUFE].

-THE RESIDUAL LINE : [RESIDU]

-THE CURRENT LINE : [COUR]

[COUR] IS THE RESULT OF THE DIFFERENT OPERATIONS ON THE SUCCESSIVE [BUFE] ENCOUNTERED AND [RESIDU] ACTUALIZED. [COUR] IS PRINTED ON THE DESIRED MEDIA ([F]ILE TO BE RETREATED WITH [EDIT] OR LINE PRINTER DIRECTLY) WHEN IT HAS SATISFIED ALL THE REQUIRED CONDITIONS.

2.1. [P]ROGRAM ORGANISATION : ...

DEC INPUT

ANNOUNCING FMS-11 FORMS MANAGEMENT SYSTEM

Digital announces the FMS-11 Forms Management System, a comprehensive set of software tools for developing form applications for Digital's new VT100/PDT terminal family.

HIGHLIGHTS

- Type your form directly on the screen - no layout charts or forms languages.
- Write form application programs in your choice of MACRO-11, FORTRAN IV or BASIC-11.
- Develop form applications under the RT-11 operating system and run them under RT-11 or its run-time subsets, RT^2 and RT^2/PDT.
- Execute the application program at each VT100 terminal independently of the programs running at the other terminals.

. Utilize VT100 features such as four video modes, scrolling, and 132 character screen.

. Enter and edit source files with the new Keypad Editor, which gives you full-screen context display and function-key ease of use.

. Modify your forms without recompiling your application.

. Simplify program maintenance and increase program flexibility by associating constant data with the form, not with the program.

PRODUCT DESCRIPTION

FMS-11 is a package of programs and subroutines that allow easy development of form applications using the VT100 terminal. A FMS-11 form application consists of user-written application programs, the ARTS and Form Driver components of FMS-11 and the user's forms in a media-resident or memory-resident form library. The remaining components of FMS-11 - FRED, KED, and FRMUTL - are tools used in developing the application. More detailed descriptions of the individual FMS-11 components follow.

FRED - The Interactive Form Editor allows the user to create forms by merely typing them on the screen. The form layout process uses powerful editing facilities invoked via the VT100 function keypad (e.g., cut and paste, delete/undelete, and repeat). Another stage of the form editing process assigns individual field attributes:

- Field Name
- Protected
- Alpha/numeric/signed numeric/mixed
- Required/must fill
- Auto tab
- Clear character/zero-fill
- Default value
- Right/left justify/fixed decimal
- Video attributes (bold, reverse, blink, underline)
- Scrolled
- Indexed
- Display only/no echo/entry by supervisor only
- Entry by supervisor only
- "HELP" text

Constant data related to the application, such as file names, names of related forms, range check parameters, or information to control the logical flow of the application may be stored along with the form. This feature, called "Named Data", allows the programmer to write highly generalized programs with parameters stored in the form. These parameters can then be edited with FRED (as can any other part of the form) without requiring a recompile or relink of the application program.

FDV -
The Interactive Form Driver is a set of subroutines called from the application program to do the following:

- Display memory-resident forms linked with the application.
- Extract form descriptions from libraries and display them.
- Provide operators with cursor control operations.
- Accept/display data fields.
- Check data validity according to field attributes.
- Respond to operator requests for HELP.

The Form Driver relieves the application programmer of most of the burden of programming the operator interface. Screen manipulation and control is performed by such calls as "GET FIELD", "PUT ALL FIELDS", and "OUTPUT A SCROLLED LINE". Built with operating efficiency in mind, the Form Driver consists of less than 8K bytes of reentrant code which can be shared by all terminals on the system.

ARTS - The Application Run-Time Supervisor is a multi-tasking submonitor that runs in the background portion of the RT-11 or RT^2 monitor. Application programs to run under ARTS may be written without regard to the number of terminals on the system; ARTS makes certain that all I/O and other requests to the monitor are associated with the current task. In other words, ARTS does for programs written in FORTRAN and MACRO what MU BASIC did for programs written in BASIC-11. ARTS options include the following:

- System tasks not attached to a terminal.
- Messages between tasks (on the same system).
- Dynamic systems where terminals can change programs independently.
- Static systems where each terminal can run only the program loaded for it initially.
- Hardware configuration specified at ARTS SYSGEN time or when ARTS starts up.

- o Reentrant resident user subroutine libraries linked with ARTS and sharable among all tasks for greater memory efficiency.

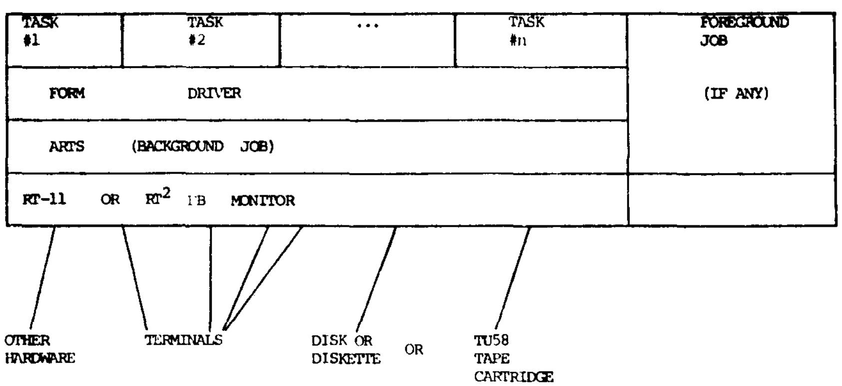
Because of its elementary demand scheduler and lack of memory mapping, ARTS is not a substitute for the RSX-11M or RSTS/E multi-user operating systems. However, it is perfectly suited for small, interactive multiterminal application systems.

FRMUTL - The Form Utility is a system program that allows you to:

- o Create versions of form descriptions that are suitable for hard-copy listings.
- o List the names of forms in a form library.
- o Produce object modules of forms descriptions, which can then be linked with the application program to produce memory-resident forms.

Also packaged with the FMS-11 software is the video Keypad Editor, KED. KED simplifies the preparation of program source files or any ASCII text file. The screen forms an 80 or 132-column window into the file which can be moved forward or backward at will. You can invoke most of KED's functions by one or two keystrokes on the VT100 function keypad. These functions include character, word, line, "section", and "page" manipulation, string search and replace, and cut and paste operations. KED is consistent with the proposed corporate standard for editors, and it may well be incorporated into the next versions of the principal operating system.

HOW A MULTI TERMINAL FMS-11 APPLICATION RUNS



1) The RT-11 monitor schedules and performs I/O and other requests for service from the foreground and background jobs.

2) ARTS accepts requests from the user tasks and passes them on to the monitor, keeping track of which task asked for what and passing the results back upon completion.

3) The Form Driver accepts field and form-level terminal I/O requests from the application program and performs these by a combination of its own processing and terminal I/O calls to ARTS. Application program requests not involving forms are made directly to ARTS.

WHERE SHOULD I USE FMS-11?

You should think of FMS-11 the same way you think of a language compiler or an editor - as a general-purpose tool for developing applications. Conceptually, FMS does for the front (operator) end of an application what a data base management system does for the back (data) end.

FMS-11 is unique among existing Digital forms products for several reasons. First, it is not inherently tied to any one language (like DEC form) operating system (like ATL), although in its initial version it is supported only under RT-11. Second, forms are entered directly on the screen instead of via forms languages, which often lack the versatility of the standard trilogy of MACRO, BASIC and FORTRAN.

The principal application area for FMS-11 is in traditional intelligent terminal applications such as source data entry and inquiry/response. Such applications are frequently ideal candidates for the PDT-11/130 or 151, and FMS-11 has been designed with the PDT in mind. However, many applications will need greater processing power, higher throughout or more disk storage than the PDT provides. In these instances, you can run FMS-11 on a PDP-11 system with hard disk and RT-11.

In fact, FMS-11 makes the operator-terminal interface of ANY program easier to design, implement and maintain. It could be used with a data acquisition application, for example, to prompt for start-up information, or to scroll through an array of processed or newly acquired data. Forms could also be used to format the periodic display of status from the monitoring of a real-time task. In all of these cases, FMS-11 makes it easier to design, implement, and maintain the terminal interface portion of the program.

Keep in mind that both the features and the documentation of the first release of FMS-11 are targeted at the reasonably sophisticated RT-11 programmer. This is especially true for ARTS, the multi-terminal Application Run-Time Supervisor. For this reasons, OEMs and large, sophisticated end users are prime markets for FMS-11.

MINIMUM HARDWARE REQUIREMENTS

A valid RT-11 or RT[2] or RT[2]/PDT system with VT100 terminals is required for application execution. ARTS without forms may be used with any supported RT-11 system terminal. Memory requirements for run time system components are 8K bytes for the Form Driver, and 2K to 12K bytes for ARTS, depending on the functionality included at ARTS SYSGEN time. (Note: This applies to Version 1 only; subsequent versions may increase in size to include more functionality.) The following chart summarizes memory requirements for FMS-11 application execution.

SYSTEM MEMORY REQUIREMENTS FOR FMS-11 FORMS APPLICATIONS

	MACRO-11	FORTRAN IV	BASIC-11
SINGLE TERMINAL	32K bytes	* 32K bytes	56K bytes
MULTI TERMINAL	56K bytes	* 56K bytes	---

*(ED. Somewhat limited because of FORTRAN OTS memory requirements.)

Form application development requires a valid RT-11 system with at least 56K bytes of memory and at least one VT100 terminal. The Digital-supplied installation procedures require that the development system have at least 2 RK05s or equivalent disk storage.

OPTIONAL HARDWARE

Additional VT100 terminals up to RT-11 maximum for each configuration.

PREREQUISITE SOFTWARE

For Application Execution

RT-11 Operating System, Version 3B or later, or

RT[2] or RT[2]/PDT, Version 3B or later.

For Application Development

RT-11 Operating System, Version 3B or later.

OPTIONAL SOFTWARE

BASIC-11/RT-11, VERSION 2

FORTRAN IV/RT-11, VERSION 2.1

XM INSTALLATIONS (MU BASIC-11/RT-11 V2)

(1) The XM version of MU BASIC does not allocate any user areas in the lower 28K words of memory. If MU BASIC is not to be run in conjunction with a foreground job, there is sufficient space to build a non-overlaid version, vastly improving performance by utilizing space that would otherwise be wasted. This can be easily done by following method (b) in the Small Buffer article* in Volume 481 (7 June 1979) entitled 'Overlay Structure Modification under XM'; see note (2) below.

(2) The formula for calculating the size of the System Data Area (SDA) given in the MU BASIC Release Notes does not work -- the value produced is too small; an SDA size of approximately 800 words appears to be necessary for an 8 user system with one system buffer.

(3) When specifying the number of words of extended memory in response to the MUCNFG program prompt, a default of zero will divide the available XM evenly per user; approximately 300. words of overhead is subtracted from each user area. (Note that the total number of words of extended memory available is total - 28K, not total - 32K.)

(4) The XM version of MU BASIC runs as a virtual job, prohibiting direct I/O page accesses by an assembly language routine linked with MU BASIC. Two techniques have been determined for overcoming this restriction; those interested may contact me directly.

* this can be made available through your local Digital office

PRINTER PORT ON PDT-11

The PDT serial printer port should be compatible with all standard line printer software, without the need for special software XON/XOFF support. The PDT printer port microcode transparently supports XON/XOFF by controlling the setting of the LPCSR done bit based on whether or not XON or XOFF characters have been received. (The microcode responds to addresses 177510 and 177512, making the port appear as if it had a DL(V)11 compatible receiver CSR and buffer, but these addresses are not otherwise functional.) In addition, the EIA data terminal ready signal is used to determine the setting of the LPCSR error bit, allowing software detection of power off, off line, or paper fault conditions.

When interfacing an LA120, LS120, or serial LA180, these terminals should be set for automatic XON/XOFF generation; this is a switch/jumper setting on the LS120 and serial LA180, and a SETUP option on the LA120.

For maximum throughput, it is recommended that the printer port/terminal be operated at 9600 baud for an LA120 or serial LA180 and 4800 baud for an LS120; the FMS-11 SPEED program (or an equivalent FORTRAN or MACRO program) can be used to change the default printer port baud rate of 1200 baud. (The RT2/PDT Installation Guide (AA-D980A-TC), part of the obsolete RT2/PDT kit, contains useful I/O programming information, as yet unpublished elsewhere.)

INTERFACING A TERMINAL AS LP: IN RT-11 V3B

The following procedure can be used in lieu of a sysgen to modify the RT-11 LP handler so that it can be used in conjunction with a serial ASCII terminal (such as the LA34 or LA36), interfaced through a DL(V)11 family interface. Before beginning the procedure, the vector and status register (csr) addresses of the DL(V)11 corresponding to the terminal must be determined. These addresses correspond to the receiver portion of the interface; they are adjusted in the patch below since the LP handler communicates only with the transmitter portion.

```
.SET LP CR<ret>

.SET LP LC<ret>

.R PATCH<ret>

FILE NAME--
*LP.SYS<ret>
*1000/  200       vector+4<ret>
*1054/  177514   csr+4<ret>
*1204/  177516   csr+6<ret>
*E
```

For example, the following patch modifies the LP handler for use in conjunction with the MINC printer port:

```
.R PATCH<ret>

FILE NAME--
*LP.SYS<ret>
*1000/  200       324<ret>
*1054/  177514   176524<ret>
*1204/  177516   176526<ret>
*E
```

This procedure need only be performed once; it permanently modifies the on-disk copy of the LP handler. The modified handler can be copied to other system disks with the command COPY/PRE/SYS LP.SYS dev:.

NOTES

(1) If the DL(V)11 has been installed at the 'standard' line printer addresses, a patch is still necessary for the reason given above. The value '204' would be entered to replace <vector+4>; the next two lines in the patch would be superfluous and can be eliminated. (If a sysgen is to be performed including line printer support in this case, you must specify a non-standard vector address of 204 during the sysgen dialogue; if a terminal is specified as a line printer during sysgen, it cannot also be specified as one of the terminals accessed via multiple terminal support.)

(2) When interfaced in this manner, the LP handler cannot detect a terminal power down, off line, or paper out condition; output sent to the printer under these conditions will be lost without warning.

(3) The standard LP handler does not provide XON/XOFF handshaking or form feed emulation. The former must be used in conjunction with an LA120, LS120 or serial LA180 if it is to be run at its maximum speed (>=2400 baud; 9600 baud recommended for LA120 and LA180, 4800 baud for LS120). The latter may be used with an LA34, LA35, or LA36 if the LP handler is to emulate the form feed function not present on these terminals; this is necessary in order to get properly paginated FORTRAN or MACRO program listings. The Small Buffer article by Fred Zayas in Volume 436 (20 July 1978) entitled 'LA180 and LA35 Support' contains source code modifications to the LP handler for these purposes that can also be applied to the uncommented source distributed in the V3B binary kit.; this can be made available, perhaps at a slight fee, from your local Digital office.

SYSGEN PROCEDURE OMISSION (RT-11 V3, V3B)

There is a serious omission in the RT-11 System Generation Manual regarding the steps to be followed at the end of a sysgen procedure to make the newly created monitor be the currently active one. Immediately after renaming the .SYG files to .SYS files (page 3-29), if the newly created monitor is to be made the one booted at volume boot time, the bootstrap blocks should be recopied via the COPY/BOOT command. In addition, if the newly created monitor has replaced the previously active monitor of the same name, the system should be immediately rebooted.

Failure to do either of these steps can result in strange crashes, if the newly created monitor has replaced a previously active monitor with the same name.

Please relate this information to any customers who may be attempting a system generation.

LINE FREQUENCY CLOCK ON PDT

All of the PDT-11/150s I have encountered in the field have had their line frequency clocks disabled. One result of this is that the FMS-11 SIS UETP program hangs immediately after being run, and I suspect that MU BASIC-11/RT-11 V2 would behave the same way. The state of the clock can be easily determined by issuing the RT-11 TIME command; if the time prints as all zeros, the clock is disabled.

To enable the clock, remove the PDT top cover by removing the two screws at the upper left and upper right of the back of the PDT cabinet. A small rectangular pack of switches will be found on the top of one of the PDT modules. To enable the line frequency clock, switch number 2 should be moved from the ON to the OFF position.

To avoid problems with any system or user software which may require the clock being enabled, it is recommended that customers be made aware of this condition and its diagnosis and cure.

COPYING DISKS USING DUP (RT-11 V3B)

Copying disks using DUP has several advantages over using PIP:

(1) the target disk need not be initialized

(2) the bootstrap (if any) and volid/owner information are copied along with the files

(3) it is faster than PIP if a large number of files are being copied (multiple directory rewrites are avoided)

To use DUP in this manner, run DUP explicitly; when the asterisk appears, type in the command line, as below. When 'continue?' appears, dismount the system disk, mount the new disk, then type a Y followed by a carriage return. When the operation is completed, DUP will prompt with 'insert system disk,Are you ready?'; at this point, dismount the new disk, remount the system disk, and then type Y followed by a carriage return.

If the USR has been set NOSWAP before running DUP, multiple disks may be copied without having to remount the system disk between each copy operation (this may not be operable in 8K words). The /W option must be used to copy the first disk; when the 'insert system disk...' message appears, respond Y even though the system disk has not been remounted, for example:


```
.SET USR NOSWAP

.R DUP
*DY:A=DY1:/I/W
continue?Y
Insert system disk,Are you ready? Y
*DY:A=DY1:/I/Y
*DY:A=DY1:/I/Y
*^C    (remount system disk before ^C)
```

NOTE: When copying floppy disks, care must be taken to ensure that the disks are formatted in the same density; if not, a invalid transfer will take place, but no warning message will be issued.

USING EXTERNAL FIELD SEPARATORS (FORTRAN IV/RT-11 V1C,V2,V2.1)

The operation of FORTRAN formatted I/O when using external field separators (see page 6-17 of the PDP-11 FORTRAN Language Reference Manual) can give unexpected results when the FORMAT contains X or T field descriptors. As an example, consider the following two FORTRAN statements:

```
      ACCEPT 100,I,J
  100 FORMAT(I4,2X,I4)
```

If '1,123' is entered, I will be assigned the value 1, and J the value 3! Usage of the T field descriptor gives equally unexpected results.

It is recommended that X and T field descriptors be avoided in FORMATs that will be used with external field separators.

VIRTUAL ARRAYS (FORTRAN IV/RT-11 V2, V2.1)

When installing FORTRAN on a 32K word PDP-11 with memory management hardware (e.g., an 11/34 or 11/60), it should be noted that there are 4K words of extended memory available that can be used for array storage via the FORTRAN VIRTUAL statement, with very little time or space overhead; for such systems, build the FORTRAN library with the VIRNP module.

MultiCept Corporation
201 West Pine Street Rome, New York 13440 (315) 337-1000

June 19, 1979

FM-11 FILE MANAGEMENT EXECUTIVE

Product Description

FM-11 is a software system designed for use with Digital Equipment Corporation's RT-11 operating system to manage the storage and retrieval of data residing on random access devices. It can be called from application programs written in FORTRAN or MACRO-11 and provides the control of input and output of data to and from the data base as well as the maintenance of relationships between data files and the records within these files. FM-11 was designed to meet the needs of RT-11 users for a small, efficient, yet comprehensive file manager which allows the programmer to use sophisticated data structures and access techniques with a minimum of programming effort.

FM-11 is a complete data management system that performs all file related functions. It can be called to create files, delete files, rename files, expand or contract the size of files, and provide access to data records by a variety of methods. Requests to FM-11 are easily incorporated into the application program by means of simple function or subroutine calls.

If anyone is interested in the above product, please contact:

Robert A. McKie
MultiCept Corp.
201 West Pine St.
Rome, New York
13440
315 337-1000

OPERATING SYSTEM	VERSION	SYSTEM PROGRAM OR DOCUMENT TITLE	VERSION OR DOCUMENT PART NO.	DATE
RT-11	V3B	BASIC-11	V2	7/11

(SEE EXAMPLE IN INSTRUCTIONS)

NAME: Ron Trellue - 1523
FIRM: Sandia Laboratories
P. O. Box 5800
ADDRESS: Albuquerque, NM 87185
ZIP:

SUBMITTED BY: Nick Bourgeois PHONE: (505) 264-8088

DEC OFFICE:
DO YOU HAVE SOURCES?

REPORT TYPE: ☒ SOFTWARE ERROR ☐ DOCUMENTATION ERROR ☐ INQUIRY ☐ FOR YOUR INFORMATION/SUGGESTION

PRIORITY: ☐ LOW ☒ STANDARD ☐ HIGH

CAN THE PROBLEM BE REPRODUCED AT WILL? YES ☒ NO ☐

ATTACHMENTS: MAG TAPE ☐ FLOPPY DISKS ☐ DECTAPE ☐ LISTING ☒ OTHER

COULD THIS SPR HAVE BEEN PREVENTED BY BETTER OR MORE DOCUMENTATION? PLEASE EXPLAIN IN PROVIDED SPACE BELOW. YES ☐

CPU TYPE	SERIAL NO.	MEMORY SIZE	DISTRIBUTION MEDIUM	SYSTEM DEVICE	DO NOT PUBLISH
11/34A	807	64KW	RK05	RK05	

A LINPUT # statement such as Line 220 in the attached listing will not accept a maximum length string (255 characters)

```
RT-11

.BASIC
BASIC-11/RT-11 V02-03N
OPTIONAL FUNCTIONS (ALL, NONE, OR INDIVIDUAL)?

READY
OLD TEST

READY
LIST

TEST      11-JUL-79  12:45:26

100 REM TEST.BAS        NAB     11-JUL-79/
110 REM TEST SEQUENTIAL FILE I/O (STRINGS)
120 I%=TTYSET(255%,255%)
130 FOR I%=1% TO 255%
140 OPEN 'TEST' FOR OUTPUT AS FILE #1
150 J$=''
160 FOR J%=1% TO I%
170 J$=J$&'+'
180 NEXT J%
190 PRINT #1,J$
200 CLOSE #1
210 OPEN 'TEST' FOR INPUT AS FILE #2
220 LINPUT #2,K$
230 PRINT I%
240 IF J$=K$ THEN 260
250 PRINT LEN(J$),LEN(K$)
```

```
260 CLOSE #2
270 NEXT I%
280 KILL 'TEST'
290 END

READY
RUN

TEST        11-JUL-79   12:45:46

 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 :
 :
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
?LINE TOO LONG AT LINE 220
 134
 134          0
?LINE TOO LONG AT LINE 220
 135
 135          1
?LINE TOO LONG AT LINE 220
 136
 136          2
?LINE TOO LONG AT LINE 220
 137
 137          3
?LINE TOO LONG AT LINE 220
 138
 138          4
?LINE TOO LONG AT LINE 220
 139
 139          5
?LINE TOO LONG AT LINE 220
 140
 140          6
?LINE TOO LONG AT LINE 220
 141
```


NAME Geoffrey R Grinton	PHONE 03 429 1511	DATE 2/7/79
COMPANY NAME & ADDRESS State Electricity Commission of Victoria, Howard Street, Richmond, 3121, Victoria, Australia		

COMPUTER	MEMORY	MASS STORAGE	OTHER OPTIONS	SOFTWARE SPECIALIST
11/34	96 KW	RK05		

SYSTEM PROGRAM & VERSION	MONITOR & VERSION	DOCUMENT	CODE	PAGE
FORTRAN V02.1-5	RT-11FB(S) V03B-00I		DEC–	

ATTACHMENTS: [X] TELETYPE PRINTOUT [ ] OBJECT TAPE [ ] SOURCE TAPE [X] LISTING [X] EXAMPLE

DETAILS OF QUESTION, SUGGESTION, PROBLEM, OR CORRECTION – PLEASE PROVIDE A COMPLETE DESCRIPTION AND ATTACH PERTINENT SUPPORTING MATERIAL

Compiler generates incorrect code:

The FORTRAN compiler generates incorrect code when floating an integer in a statement which refers to an array element.

The simplest case which shows the error is -

```
      REAL B(1)
      A=I-B(J)
      END
```

The attached terminal output and listing demonstrate the problem.

```
FORTRAN IV      Generated Code for Program Unit .MAIN.

                                        ; Statement #0002
000006     MOV    #2,@$AOTS                                      R-012345
000014     MOV    J,-(SP)                                        R-012345
000020     ASL    @SP                                            R-012345
000022     ASL    @SP                                            R-012345
000024     MOV    I,-(SP)                                        R-012345
000030     JSR    PC,$CVTIF                                      R-012345
000034     MOV    (SP)+,R0                                       R- 12345
000036     MOV    B-2(R0),-(SP)                                  R- 12345
000042     MOV    B-4(R0),-(SP)                                  R- 12345
000046     JSR    PC,$SUBF                                       R- 12345
000052     MOV    (SP)+,A                                        R- 12345
000056     MOV    (SP)+,A+2                                      R- 12345
                                          Statement #0003
000062     INC    @$AOTS                                         R-012345
000066     RTS    PC                                             R-012345
```

Top of stack now has float of I, not 4*J as this assumes.

OPERATING SYSTEM	VERSION	SYSTEM PROGRAM OR DOCUMENT TITLE	VERSION OR DOCUMENT PART NO.	DATE
RT-11	V3B	FORTRAN	2.1	6-28-7

(SEE EXAMPLE IN INSTRUCTIONS)

NAME: RON TRELLUE / 1523
FIRM: SANDIA LABORATORIES
ADDRESS: P O BOX 5800
ALBUQUERQUE NM
ZIP: 87185

DEC OFFICE: ALBQ
DO YOU HAVE SOURCES? YES ☐ NO ☐

REPORT TYPE: ☒ SOFTWARE ERROR ☐ DOCUMENTATION ERROR ☐ INQUIRY ☐ FOR YOUR INFORMATION/SUGGESTION
PRIORITY: ☐ LOW ☐ STANDARD ☒ HIGH

CAN THE PROBLEM BE REPRODUCED AT WILL? YES ☒ NO ☐

SUBMITTED BY: NICK BOURGEOIS
PHONE: 264-8088

ATTACHMENTS: MAG TAPE ☐ FLOPPY DISKS ☒ DECTAPE ☐ LISTING ☒ OTHER

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

CPU TYPE	SERIAL NO.	MEMORY SIZE	DISTRIBUTION MEDIUM	SYSTEM DEVICE	DO NOT PUBLISH
807	11/34A	64 KW	RK05	RK05	☐

CARRIAGE CONTROL WORKS WITH FORTRAN 2.0 AND NOT WITH FORTRAN 2.1 WHEN USING THE CALL ASSIGN SUBROUTINE.

```
TEST USING FORTRAN 2.0

START OF TEST
SPACE:  ADVANCE ONE LINE
SPACE:  ADVANCE ONE LINE
SPACE:  ADVANCE ONE LINE

ZERO:  ADVANCE TWO LINES

ZERO:  ADVANCE TWO LINES

[illegible]NEXT PAGE
DOLLAR SIGN:  INHIBIT CARRIAGE RETURN
                                 DOLLAR SIGN:  INHIBIT CARRIAGE RETURNPLUS AND DOLLAR SIGN:PLUS AND DOLLAR SIGN:PLUS AND DOLLAR S
IGN:
    END OF TEST

TEST USING FORTRAN 2.1

0START OF TEST
 SPACE:  ADVANCE ONE LINE
 SPACE:  ADVANCE ONE LINE
 SPACE:  ADVANCE ONE LINE
0ZERO:  ADVANCE TWO LINES
0ZERO:  ADVANCE TWO LINES
0ZERO:  ADVANCE TWO LINES
1ONE:  ADVANCE TO TOP OF NEXT PAGE
1ONE:  ADVANCE TO TOP OF NEXT PAGE
+PLUS:  INHIBIT ADVANCE
+PLUS:  INHIBIT ADVANCE
+PLUS:  INHIBIT ADVANCE
+PLUS:  INHIBIT ADVANCE
+PLUS:  INHIBIT ADVANCE
$DOLLAR SIGN:  INHIBIT CARRIAGE RETURN
$DOLLAR SIGN:  INHIBIT CARRIAGE RETURN
+PLUS AND DOLLAR SIGN:
                      +PLUS AND DOLLAR SIGN:
                                            +PLUS AND DOLLAR SIGN:
                                                                  END OF TEST
```


OPERATING SYSTEM	VERSION	SYSTEM PROGRAM OR DOCUMENT TITLE	VERSION OR DOCUMENT PART NO.	DATE
RT-11	V3B			5-31-7

(SEE EXAMPLE IN INSTRUCTIONS)

NAME: RON TRELLUE / 1523
FIRM: SANDIA LABORATORIES
ADDRESS: P O BOX 5800
ALBUQUERQUE, NM 87185
ZIP:

DEC OFFICE: ALBQ
DO YOU HAVE SOURCES? YES NO

REPORT TYPE: ☐ SOFTWARE ERROR ☐ DOCUMENTATION ERROR ☐ INQUIRY ☐ FOR YOUR INFORMATION/SUGGESTION
PRIORITY: ☐ LOW ☐ STANDARD ☒ HIGH

CAN THE PROBLEM BE REPRODUCED AT WILL? YES ☒ NO ☐

SUBMITTED BY: NICK BOURGEOIS
PHONE: 264-8088

ATTACHMENTS: MAG TAPE ☐ FLOPPY DISKS ☒ DECTAPE ☐ LISTING ☒ OTHER PHOTO

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

CPU TYPE	SERIAL NO.	MEMORY SIZE	DISTRIBUTION MEDIUM	SYSTEM DEVICE	DO NOT PUBLISH
11/34A	807	64 KW	RK05	RK05	

AN ATTEMPT TO USE IRCVDW IN A FOREGROUND JOB CAUSES THE SYSTEM TO HALT. NOTE IN THE ENCLOSED PHOTOGRAPH OF THE CONSOLE SCREEN THAT THE PC SHOWS 034072 WHEN THE PROGRAM WAS LOADED AT 115554.

OPERATING SYSTEM	VERSION	SYSTEM PROGRAM OR DOCUMENT TITLE	VERSION OR DOCUMENT PART NO.	DATE
RT-11	3B-00	MACRO	03.02B	14-Ju

NAME: Daniel Smith
FIRM: Eye Research Institute
ADDRESS: 20 Staniford Street
Boston, Mass. 02114
CUST. NO.: 000184

DEC OFFICE: Waltham (S. Simpson)
DO YOU HAVE SOURCES? YES ☐ N

REPORT TYPE/PRIORITY: ☒ PROBLEM/ERROR ☐ SUGGESTED ENHANCEMENT ☐ OTHER
1. ☐ HEAVY SYSTEM IMPA
2. ☐ MODERATE SYSTEM I
3. ☒ MINOR SYSTEM IMPAC
4. ☐ NO SIGNIFICANT IMP.
5. ☐ DOCUMENTATION/SUG

SUBMITTED BY: Daniel Smith
PHONE: 617 742-3140 x 260

CAN THE PROBLEM BE REPRODUCED AT WILL? YES ☒ N

ATTACHMENTS: MAG TAPE ☐ FLOPPY DISKS ☐ LISTING ☒ DECTAPE ☐ OTHER:

COULD THIS SPR HAVE BEEN PREVENTED BY BETTER OR MORE DOCUMENTATION? PLEASE EXPLAIN IN PROVIDED SPACE BELOW. YES ☐ N

CPU TYPE	SERIAL NO.	MEMORY SIZE	DISTRIBUTION MEDIUM	SYSTEM DEVICE	DO NOT PUBLISH
11/40			RK05	RK05	

Problem A (relatively serious)

MACRO has a bug concerning the use of automatically-generated local symbols in conjunction with arguments called by keyword.

The effect of the bug is that the ?-argument gets set to BLANK instead of to ##$.

The combinations eliciting the bug are hard to characterize but are completely reproducible.

One example is
```
        .MACRO TEST ONE,TWO,THREE,FOUR=4,FIVE=5,SIX=6,?SEVEN
```
called by
```
        TEST 1,2,3,FOUR=4          elicits the bug
                                   (SEVEN expands to null)
```

while

```
        TEST 1,2,3,FIVE=5          does not
                                     (SEVEN expands to 64$, etc.)
```

Other examples are shown in the attachment.

Because the problem is dependent on the keywords used (among other things), this problem has caused some serious difficulties in that locally-used system macros which had appeared to be tested and working fully would fail mysteriously when certain combinations of keyword arguments were invoked. Although each specific combination gives fully reproducible results, the lack of any obvious rule for predicting which combinations cause trouble makes it very difficult to bypass the problem except by avoiding <u>all</u> use of keyword arguments in combination with generated symbols.

Problem B (much less serious but annoying): Even if .ENABL LC is invoked, it is impossible to use lower-case material within a macro definition; it gets converted to upper case. It is very disappointing to encounter problems like this in a product as mature as MACRO.

This section documents miscellaneous facts that are otherwise obscure or undocumented.

AR11 Maintenance Mode

To simulate a tick of the 1MHz. clock, it is necessary to turn bit 11 on, <u>then</u> <u>off!</u> (e.g. BIS #4000,CLK.C;BIC #4000,CLK.C). The counter increment occurs when the bit is cleared.

MACRO

1) If .MACRO FOO I,J,K,?X is invoked by the call FOO A, there are still four arguments as far as .NARG is concerned.

2) If .ENABL LC, lower-case arguments can be passed to macros; however, lower-case strings within the body of a macro definition will still be converted to upper case.

PDP-11

Double-precision notes

The sequences ADD #-1,R0 and SUB #1,R0 are not identical in effect. The effect on R0 is the same, but the first instruction will set the "carry" bit and the second will not. (This is in accord with the interpretation of C as a "borrow" after a SUB, the use of SBC, etc. but is not obvious because it does imply a difference in the condition-setting logic for the two instructions.)

60 Hz. interrupt

Under the FB monitor, with no special scheudling or timer support explicitly enabled, each interrupt from the 60 Hz. clock takes about 0.4 milliseconds (!) to process.

This was determined by running a simple FORTRAN program (TRY60H.FOR) which includes the loop

```
      DATA IPANEL/"116176/
200   CALL IPOKE(IPANEL,"177776)
      CALL IPOKE(IPANEL,"16)
      GO TO 200
```

which flips the panel "sync out" bits as fast as the loop runs (about 64 microseconds, incidentally), and observing the result on a scope. 0.4 millisecond gaps, synchronized with the line, are plain. (Note, however, that they occur at a low priority, i.e. 6).

.INTEN register preservation

It appears to be an undocumented-in-version-3 fact that after an .INTEN call, R4 and R5 are available for use (i.e. .INTEN saves them and the RTS PC from .INTEN restores them).

This conclusion is based on the following observations:

1) The .INTEN call expands into a JSR R5, . . . thus saving R5

2) The first instruction in the INTEN routine is a MOV R4,-(PC), saving R4;

3) .SYNCH expands into a sequence beginning with MOV address,%4. Also, the text notes that a successful SYNCH request alters R4 and R5. Therefore, if .SYNCH is to appear in an interrupt routine, R4 and R5 must be saved and restored. If this is not done automatically, it is not clear how it is to be done— especially since the SYNCH description warns that nothing may be pushed onto the stack between INTEN and SYNCH. Notice that the example shows no user code for preservation of R4 and R5.

4) In the version 2C manual, the INTEn and SYNCH descriptions refer to an appendix H, which does not appear in the version 3 manual. H.1.2 states that .INTEN preserves R4 and R5.

Tape Compatibility, V02-01 and V03B-00

Page 1-31 of the V3B Advanced Programmers Guide states:

> The file-structure handler searches through file names . . . The algorithm used is compatible with the DIGITAL standard. It allows tapes written under RT-11 V02C and earlier versions to be read by V03 and later versions and matched (these tapes don't have a dot to separated the file name from the file type).

The statement in the V3 manual (p. 1-26) is more explicit:

> . . . it allows tapes written under RT-11 V02, V02B and V02C to be read and matched . . .

What this appears to mean in practice is that, running V03B with a V02-01 tape mounted on (say) MT1:, commands such as

```
.COPY MT1:QXZ.FOR RK0:
~~.COPY MT1:*.* RK1:~~ → Bombs!
.TYPE MT1:NOTES.DOC
```

will work.

The reason for this note is that

```
.DIR MT1:
```

will not work: it rewinds the tape, reads forward slightly, and causes a processor halt.

One way to find out what it on a V02 tape while running V03B is

```
.COPY MT1:*.* NL:
```

and noting the "Files copied:" message. This makes only one pass over the tape and is, in fact, faster than using V02 and doing

```
.R PIP
*MT1:*.*/L
```

which appears to make two.

OPERATING SYSTEM	VERSION	SYSTEM PROGRAM OR DOCUMENT TITLE	VERSION OR DOCUMENT PART NO.	DATE
RT-11	V3B	APL-11	V1.00	1-AUG-79

(SEE EXAMPLE IN INSTRUCTIONS)

NAME: N Bevan
FIRM: National Physical Laboratory
ADDRESS: DNACS
Teddington, Middx, TW11 0LW
England

DEC OFFICE: UK

DO YOU HAVE SOURCES? YES ☐ NO ☒

REPORT TYPE/PRIORITY: ☐ PROBLEM ERROR ☒ SUGGESTED ENHANCEMENT ☐ OTHER — ☐ 5. ☐ 4. ☒ 3. ☐ 2. ☐ 1.

SUBMITTED BY: N Bevan PHONE: 01-977 3222

CAN THE PROBLEM BE REPRODUCED AT WILL? YES ☐ NO ☐

ATTACHMENTS: MAG TAPE ☐ FLOPPY DISKS ☐ LISTING ☒ DECTAPE ☐ OTHER

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

CPU TYPE	SERIAL NO.	MEMORY SIZE	DISTRIBUTION MEDIUM	SYSTEM DEVICE	DO NOT PUBLISH
11/10	0404101	30KW	floppy	RK	☐

APL-11 is generally an impressive system, but has one or two shortcomings which limit its usefulness:

1. The symbol table is ridiculously small. 800 bytes leaves room for 65 symbols with an average length of 4 characters. This sounds adequate until seen in terms of ten functions, when it means an average of 5.5 symbols each for variables and labels! Given the modular use of functions in APL this is a serious restriction. As it is difficult to decide what the optimum value would be, what is required is a command to set the symbol table size in a clear workspace, of failing that a patch or LINK option to increase the size.

2. APL-11 is a faithful implementation of APL\360, but leaves something to be desired in comparison with current-day systems. One or two minor additions would make it far easier to implement functions developed elsewhere:
   a) Monadic .FM (format), similar to monadic quote but producing an array rather than a vector.
   b) .XQ as an alternative to .EP.
   c) Diamond (or overstruck <>) for proper multiple-statement lines.
   d) Quad variables as alternatives to I-beams.
   e) Dyadic format, more difficult to implement but extremely useful. The conventional alternative of an FMT function is currently rather thirsty on the symbol table.
   f) Comments at the end of an executable line are documented as not being allowed, but generally seem to work.

3. Now that RT-11 V3B is standard, could we have a version of APL which traps ^C please.

4. Your convention of using a character error to delete a line can be VERY frustrating after typing in a complex overstruck line. It would be far more useful if the line was regenerated up to the position of the error so that the remainder could be retyped.

5. It is difficult to make efficient use of APL without a good library of functions. Is there any chance of aquiring one for APL-11? Perhaps there is a DEC 10 library which could be adapted and distributed via DECUS?

6. A better method is required for building an APL library. The only way to do this at present is to have a large number of files, each with one or 2 functions. What is required is:
)READ filename objectname(s)
so that the functions required can be retrieved from a library file.

7. Why not ")WRITE file" analagous to ")READ file" which would:
)CREATE 1 file,)WRITE 1,)CLOSE 1.

8. A facility which would be particularly valuable in APL-11 (which probably has many inexperienced users) is the)OBSERVE command to show the intermediate values of traced statements.

9. Transfer of functions from other systems is complicated by the lack of any support for reading APL character-set function definitions from file. The ASCII representation of the APL character set is portable, whereas the mnemonics vary from system to system.

10. It is difficult to check the ASCII value of data, eg to convert lower case to upper case, or to find a <CR> character. An "atomic vector" of characters in ASCII sequence would solve this problem.

11. Could "VAR .QQ 'Prompt'" be used to allow input on the same line?

Language errors

1. It is not possible to rotate a single element: **A1**
    1 .RV 1

2. It is not possible to rotate a vector by a 1-element vector: **A2**
    (1 .RO 1) .RV 1 2 3

File I/O problems

1. With an RK system:)ASSIGN 1 DX1:TEST.DAT
   sometimes gives: DEVICE NOT FOUND
   unless DX is loaded.

2. Documentation of .PT is confusing: 1 .PT 5 **B2**
   sets the pointer to 5, but returns value 4!

3.)ASSIGN does not close a channel which is already open. **B3**

4.)READ does not replace existing functions in the workspace. **B4**

5. 1 .IQ 1 reads 1 line rather than 1 character as documented.

6. An underscored letter written to file as an ASCII character **B6**
   is stored in APL internal format when in LA36 mode.

7. RT-11 allows filenames starting with a digit. Although not accessable from APL, what is the effect of:
)ASSIGN 1 DK:12.DAT ???

Line editing errors

1. When editing function header the dummy line number: [0] **C1**
   is omitted, thus misaligning the editing pointer.

2. If a blank is inserted near the end of the line, after pressing <CR> the carriage returns to the end of the line instead of the first inserted blank.

3. <LF> does not echo <CR>, thus moving the line to the right.

APL System errors

1. Comment lines occasionally have <CTRL D> appended to them **D1**
   when written to file. This has to be edited out before they can be read again.

2. Certain error messages reset the upper case bit temporarily. **D2**

3. Some error conditions result in labels becoming global **D3**
   variables.

4. Some monadic quote errors return to the monitor, eg: **D4**
    .EN 'A B'

5. Various SYSTEM ERRORS and M-Traps attached. **D5**

RT-11 errors

1. Backspace echoes ^H. This makes APL practically unusable in LA36 mode! I have my own temporary RT-11 patch, but how about an official one?

2. APL patch 7, System error on parameter return, does not **E2**
   work!! Patches to APL00 attached.

ABSTRACT

A mechanism is described whereby two jobs running under the RT-11 FB monitor may share a common set of reentrant subroutines, thus reducing the amount of physical memory needed.

INTRODUCTION

Although the sixteen-bit virtual address space of the PDP-11 is often a troublesome constraint for users of all PDP-11 operating systems, its impact is felt most strongly in an unmapped system such as RT-11. Whereas user tasks in a system which supports memory management may be a full 32K words long, the RT-11 FB monitor requires that both the foreground job and the background job reside simultaneously in a region approximately 24K words in size (26K on some 11/03 systems). This can be a serious problem, particularly if both jobs are written in Fortran.

It's frequently the case that the two jobs use a common set of subroutines which are reentrant, and which could therefore be shared if a mechanism for doing so were provided. Other operating systems (RSX-11M, for example) provide a construct known as a "resident library", which is simply a region of memory containing reentrant code which can be shared by two or more tasks. Such systems provide a method for creating resident libraries, and for making use of information from an associated symbol table when tasks which will use the resident library are built.

RT-11 does not support resident libraries. It is possible, however, to implement something which is functionally equivalent. This is accomplished by putting reentrant subroutines into what the system thinks is a device handler (henceforth referred to as a "resident library handleroid"), and providing a mechanism for telling the linker the addresses at which these routines can be found when this pseudo-handler has been loaded into memory.

CREATING A RESIDENT LIBRARY HANDLEROID, LB.SYS

To make an RT-11 resident library, it's necessary to do the following:

(1) Establish the set of routines which will constitute the resident library. Create a file LB.MAC which references these routines.

(2) Determine the address at which the shared code will live.

(3) Use the linker to create a file LB.SAV, which contains the resident library code beginning at the predetermined address. At the same time, create a symbol table, LB.STB, which will contain information reflecting the addresses at which the subroutines in the resident library can be found when the handleroid is in memory.

(4) Use the information in LB.SAV to create a file LB.SYS, which contains the same code, but beginning at block 1 (virtual address 1000_8).

(5) Whenever a program which will use the resident library is built, specify LB.STB as an input file at link time.

(6) When the resident library is required, install and load the LB handleroid, ensuring that the code is indeed resident at the addresses reflected in the symbol table file.

It's possible to simplify things somewhat, and also help ensure system integrity, if a few conventions are observed:

--- Only one resident library is used at any given time. Normally, there would not be more than one resident library in any case.

--- The resident library is always loaded immediately below RMON. "INS LB" and "LOA LB" are the first two commands in STARTF.COM.

--- To protect against inadvertent use of LB.SYS (as a device handler, rather than as a resident library), the handleroid will contain null handler code copied from NL.SYS. The last word of the LB handleroid (except for the last two words, which are used for the system's INTEN and FORK pointers) will contain the size (in bytes) of this null handler prologue.

--- LB.MAC, in addition to referencing all global routines which will make up the resident library, also defines two global symbols: LB$VER, a version number which is always updated whenever a change is made to LB.MAC, and LB$ADR, a word in LB.MAC (which will therefore immediately follow the null handler code prologue in LB.SYS) which contains the value of LB$VER.

--- Any program which uses the resident library must, before calling any resident library routines, check that:

(1) The resident library handleroid is loaded
(2) The version number of the loaded handleroid matches the version number defined in LB.STB at the time the job was linked.
(3) The handleroid is loaded at the address at which the user program expects it to be, based on information in LB.STB at the time the job was linked.

The program which creates LB.SYS from LB.SAV must therefore put the following information into LB.SYS, beginning at block 1:

--- Null handler code prologue
--- All code from LB.SAV
--- A word which contains the size of the null code prologue
--- Two words of zero (for the system's INTEN and FORK pointers)

It must also put information into block 0 to reflect the size of the LB handleroid, as well as whatever other information is required to be present in the first block of a device handler. An example of such a program, MAKELB, is given in figure 2.

To create a resident library handleroid, then, one would proceed as follows (assuming that STARTF.COM contains the commands to install and load LB.SYS, and that LB.MAC has been set up appropriately):

```
.MAC LB
ERRORS DETECTED:0

.E 54
140002

.R LINK
*LB,LB,LB=LB,MYLIB,FORLIB/H:137774
*^C

.R MAKELB
STOP -- LB.SYS handleroid created successfully

.BOO SY:

RT-11FB  V03B-00C

.INS LB

.LOA LB
```

The value used with the /H switch is obtained by subtracting 6 from RMON's base address, to take into account the space which will be required following the resident library code for the word which holds the size of the null code prologue and for the two words needed for the INTEN and FORK pointers.

To build a job which uses the resident library, include LB.STB as an input file at link time:

```
.LINK/EXE:PROG LBMAIN,PROG,SUBS,LB.STB,FORLIB
```

Macro-11 programs should begin by checking that the LB handleroid is loaded, and that its version number and starting address are what they're expected to be. A Fortran main program should be converted to a subroutine which is called by a Macro-11 main pro-

```
        .TITLE  LB -- RT-11 Resident Library Definition Module


        .GLOBL  LB$ADR,LB$VER

LB$VER  =       ^1A     ; Current handleroid version number

LB$ADR: .WORD   LB$VER


        .GLOBL  $AOTS

$AOTS   =       270     ; Fortran impure area pointer


; Routines from FORLIB

        .GLOBL  ABS     ; ABS
        .GLOBL  ADF$IM  ; ADDM
        .GLOBL  ADF$IP  ; ADDP
        .GLOBL  AIF$    ; AIF
        .GLOBL  ALOG    ; ALOG
        .GLOBL  BTK$F   ; ABFRET
        .GLOBL  ASSIGN  ; ASSIGN
        .GLOBL  AND$    ; BITDID
        .GLOBL  BEG$    ; BRAS
        .GLOBL  CAI$    ; CALL
        .GLOBL  $CLOSE  ; CLOSE
        .GLOBL  CLOSE   ; CLS
        .GLOBL  CMF$II  ; CMPF
        .GLOBL  CIC$    ; CONV2
        .GLOBL  CCI$    ; CONV3
        .GLOBL  DCO$    ; CONVF
        .GLOBL  ICI$    ; CONVI
        .GLOBL  LCI$    ; CONVL
        .GLOBL  CPD$SM  ; COPY
        .GLOBL  DATE    ; DATE
        .GLOBL  MOD$MS  ; DMOV5
        .GLOBL  $DUMPL  ; DUMPLA
        .GLOBL  SAD$IM  ; DVEC
        .GLOBL  DEC$    ; ENCODE
        .GLOBL  END$    ; ENDERR
        .GLOBL  EOL$    ; EOL
        .GLOBL  EXIT    ; EXIT
        .GLOBL  EXP     ; EXP
        .GLOBL  ADD$    ; FADD
        .GLOBL  $FCALL  ; FCALL
        .GLOBL  $FCHNL  ; FCHNL
        .GLOBL  DIF$IS  ; FDIV
        .GLOBL  $FIO    ; FIO
        .GLOBL  MOF$SS  ; FMOV1
        .GLOBL  MOF$IS  ; FMOV2
        .GLOBL  MOF$MS  ; FMOV3
        .GLOBL  MOF$SA  ; FMOV4
        .GLOBL  MOF$SM  ; FMOV6
        .GLOBL  MOF$IM  ; FMOV7
        .GLOBL  MOF$OA  ; FMOV8
        .GLOBL  MOF$MA  ; FMOV9
        .GLOBL  MOF$RA  ; FMOVR
        .GLOBL  MUF$IS  ; FMUL
        .GLOBL  NGD$A   ; FNEG
        .GLOBL  SAF$PM  ; FPVEC
        .GLOBL  SAF$IM  ; FVEC
        .GLOBL  SAF$IP  ; FVECP
        .GLOBL  $GETFI  ; GETFIL
        .GLOBL  $GETRE  ; GETREC
        .GLOBL  JMC$    ; GOTO
        .GLOBL  IABS    ; IABS
        .GLOBL  ADI$IA  ; IADDS
        .GLOBL  CMI$II  ; ICMPS
        .GLOBL  IDATE   ; IDATE
        .GLOBL  DII$IS  ; IDIV
        .GLOBL  IFR$    ; IFR
        .GLOBL  IFW$    ; IFW
        .GLOBL  MOI$RA  ; IMOVR
        .GLOBL  MOI$IA  ; IMOVS
        .GLOBL  MUI$IS  ; IMUL
        .GLOBL  DCI$A   ; INCR
        .GLOBL  $INITI  ; INITIO
        .GLOBL  ADI$IP  ; IPADDS
        .GLOBL  CMI$IP  ; IPCMPS
        .GLOBL  MOI$IP  ; IPMOVS
        .GLOBL  SUI$IP  ; IPSUBS
        .GLOBL  SAI$PM  ; IPVEC
        .GLOBL  SUI$IA  ; ISUBS
        .GLOBL  SAI$IM  ; IVEC
        .GLOBL  SAI$IP  ; IVECP
        .GLOBL  CML$MI  ; LCMPI
        .GLOBL  CML$IM  ; LCMPS
        .GLOBL  MOL$IA  ; LMOVS
        .GLOBL  COI$A   ; LNOTS
        .GLOBL  LEG$    ; LOADS
        .GLOBL  SAL$PM  ; LPVEC
        .GLOBL  TSL$I   ; LTEST
        .GLOBL  SAL$IM  ; LVEC
        .GLOBL  SAL$IP  ; LVECP
        .GLOBL  MAX0    ; MAX0
        .GLOBL  MIN0    ; MIN0
        .GLOBL  $VRINT  ; NOVIR
        .GLOBL  NMI$II  ; NXT1
        .GLOBL  NMI$IM  ; NXT2
        .GLOBL  NMI$IP  ; NXT4
        .GLOBL  DEO$    ; OBJENC
        .GLOBL  $OBJFM  ; OBJFMT
        .GLOBL  $PUTRE  ; PUTREC
        .GLOBL  RET$    ; RETS
        .GLOBL  RWD$    ; REWIND
        .GLOBL  DEF$    ; RIO
        .GLOBL  $EOFIL  ; RWBLK
        .GLOBL  SAVR4$  ; SAV4RG
        .GLOBL  SAVRG$  ; SAVREG
        .GLOBL  SIGN    ; SIGN
        .GLOBL  SQRT    ; SQRT
        .GLOBL  FOO$    ; STOP
        .GLOBL  $OTIS   ; SUBR
        .GLOBL  TSD$I   ; TESTS
        .GLOBL  TAD$    ; TRARY
        .GLOBL  TVD$    ; VTRAN
        .GLOBL  $WAIT   ; WAIT
        .GLOBL  XFF$    ; XFF
        .GLOBL  XFI$    ; XFI


; Other reentrant routines

        .GLOBL  EQUSTR
        .GLOBL  KHAR
        .GLOBL  LAND
        .GLOBL  LOR
        .GLOBL  MOVSTR
        .GLOBL  MVWRDS
        .GLOBL  TYPE


        .END
```

Figure 1: Example of a typical LB.MAC

```
      PROGRAM MAKELB
C
C--->  This program creates an RT-11 "handleroid" which can be used
C--->  in the same manner as an RSX-11 resident library containing
C--->  reentrant subroutines which can be shared by different jobs.
C--->  The "handleroid" (LB.SYS) is produced from the null device
C--->  handler (NL.SYS) and file LB.SAV which, in turn, is built
C--->  with the linker's /H:xxxxxx switch, where the value of xxxxxx
C--->  is equal to <RMON's base address>-6.
C
C--->  In order to provide some protection against inadvertent use
C--->  of the handleroid (as a handler, rather than as a resident
C--->  library), the handleroid contains a prologue consisting of
C--->  code copied from the null handler.  In LB.SYS, the word
C--->  immediately preceding the INTEN and FORK pointers contains
C--->  the size of this prologue (in bytes), so that a program can
C--->  determine at run time the address of the resident library
C--->  section of the handleroid, in order to confirm that it is
C--->  indeed loaded at the correct address.
C
C--->  Since this program uses SYSLIB routines LOOKUP and IENTER,
C--->  it's advisable that the USR not swap over it
C
C
      IMPLICIT INTEGER (A-Z)
C
      COMMON  BLKSZW,BLKSZB,NCHN,LCHN,HCHN,NBLK,LBLK,HBLK,
     *  NBUFF(256),LBUFF(256),HBUFF(256)
C
      REAL*8  NFILE,LFILE,HFILE
C
      DATA  BLKSZW,BLKSZB/256,512/, NBLK,LBLK,HBLK/3*-1/
C
      DATA  NFILE/12RSY NL    SYS/, LFILE/12RDK LB    SAV/,
     *  HFILE/12RSY LB    SYS/
C
C
C--->  Open input files
C
      NCHN = IGETC ()
      LCHN = IGETC ()
      HCHN = IGETC ()
C
      IF ( LOOKUP ( NCHN, NFILE ) .LT. 0 ) STOP 'NL.SYS lookup error'
      IF ( LOOKUP ( LCHN, LFILE ) .LT. 0 ) STOP 'LB.SAV lookup error'
C
C
C--->  Get file parameters
C
      NLSIZB = GETNL ( "52 )
      PRLSZB = NLSIZB - 4
      PRLSZW = PRLSZB / 2
C
      LBSTRT = GETLB ( "42 )
      LBSIZB = GETLB ( "50 ) - LBSTRT
      LBSIZW = LBSIZB / 2
C
C
C--->  Open output file
C
      LSTADR = BLKSZB + NLSIZB + LBSIZB + 2
      CALL CONVRT ( LSTADR, LSTBLK, DUMMY )
      IF ( IENTER ( HCHN, HFILE, LSTBLK+1 ) .LT. 0 )
     *  STOP 'NL.SYS enter error'
C
C
C--->  Copy handler code from null handler to LB handleroid
C
      NADDR = "1000
      HADDR = "1000
C
      DO 100 W = 1,PRLSZW
100   CALL PUT ( HADDR, GETNL ( NADDR ) )
C
C
C--->  Copy reentrant code from LB.SAV to LB.SYS
C
      LADDR = LBSTRT
C
      DO 200 W = 1,LBSIZW
200   CALL PUT ( HADDR, GETLB ( LADDR ) )
C
C
C--->  Write out prologue size, and fill out with zeroes (at least
C--->  two, for the INTEN and FORK pointers)
C
      CALL PUT ( HADDR, PRLSZB )
      CALL PUT ( HADDR, 0 )
C
300   CALL PUT ( HADDR, 0 )
      IF ( MOD ( HADDR, BLKSZB ) .NE. 0 ) GO TO 300
C
C
C--->  Update information in block 0 of LB.SYS, and close the file
C
      NADDR = 0
      HADDR = 0
C
      DO 400 W  = 1, BLKSZW
      VAL = GETNL ( NADDR )
      IF ( HADDR.EQ."50 .OR. HADDR.EQ."52 ) VAL = VAL + LBSIZB + 2
400   CALL PUT ( HADDR, VAL )
C
      CALL WRTBLK
C
      CALL CLOSEC ( NCHN )
      CALL CLOSEC ( LCHN )
      CALL CLOSEC ( HCHN )
C
      STOP 'LB.SYS handleroid created successfully'
C
C
      END

      FUNCTION GETNL ( ADDR )
C
C--->  Returns contents of specified word from NL.SYS; on
C--->  exit, ADDR is bumped to point to next word
C
      IMPLICIT INTEGER (A-Z)
      COMMON  BLKSZW,BLKSZB,NCHN,LCHN,HCHN,NBLK,LBLK,HBLK,
     *  NBUFF(256),LBUFF(256),HBUFF(256)
C
      CALL CONVRT ( ADDR, BLK, WORD )
      IF ( BLK .EQ. NBLK ) GO TO 100
      IF ( IREADW ( BLKSZW, NBUFF, BLK, NCHN ) .LT. 0 )
     *  STOP 'NL.SYS read error'
      NBLK = BLK
C
100   GETNL = NBUFF ( WORD )
      ADDR = ADDR + 2
      RETURN
C
      END
---------------------------------------------------------------------
      FUNCTION GETLB ( ADDR )
C
C--->  Returns contents of specified word from LB.SAV; on
C--->  exit, ADDR is bumped to point to next word
C
      IMPLICIT INTEGER (A-Z)
      COMMON  BLKSZW,BLKSZB,NCHN,LCHN,HCHN,NBLK,LBLK,HBLK,
     *  NBUFF(256),LBUFF(256),HBUFF(256)
C
      CALL CONVRT ( ADDR, BLK, WORD )
      IF ( BLK .EQ. LBLK ) GO TO 100
      IF ( IREADW ( BLKSZW, LBUFF, BLK, LCHN ) .LT. 0 )
     *  STOP 'LB.SAV read error'
      LBLK = BLK
C
100   GETLB = LBUFF ( WORD )
      ADDR = ADDR + 2
      RETURN
C
C
---------------------------------------------------------------------
      SUBROUTINE PUT ( ADDR, VALUE )
C
C--->  Writes word to output file LB.SYS (if block change
C--->  is required, current block is written to disk and
C--->  the appropriate block is read in; otherwise, the
C--->  value is simply put into the output buffer); on
C--->  exit, ADDR is bumped to point to the next word
C
      IMPLICIT INTEGER (A-Z)
      COMMON  BLKSZW,BLKSZB,NCHN,LCHN,HCHN,NBLK,LBLK,HBLK,
     *  NBUFF(256),LBUFF(256),HBUFF(256)
C
      CALL CONVRT ( ADDR, BLK, WORD )
      IF ( BLK .EQ. HBLK ) GO TO 100
C
      IF ( HBLK .GE. 0 ) CALL WRTBLK
      IF ( IREADW ( BLKSZW, HBUFF, BLK, HCHN ) .LT. 0 )
     *  STOP 'LB.SYS read error'
      HBLK = BLK
C
100   HBUFF ( WORD ) = VALUE
      ADDR = ADDR + 2
      RETURN
C
      END
---------------------------------------------------------------------
      SUBROUTINE WRTBLK
C
C--->  Writes current output buffer to LB.SYS
C
      IMPLICIT INTEGER (A-Z)
      COMMON  BLKSZW,BLKSZB,NCHN,LCHN,HCHN,NBLK,LBLK,HBLK,
     *  NBUFF(256),LBUFF(256),HBUFF(256)
C
      IF ( IWRITW ( BLKSZW, HBUFF, HBLK, HCHN ) .LT. 0 )
     *  STOP 'LB.SYS write error'
      RETURN
C
      END
---------------------------------------------------------------------
      SUBROUTINE CONVRT ( ADDR, BLK, WORD )
C
C--->  Routine to convert an address to a <block,word> pair (note
C--->  that WORD ranges from 1 to 256, not from 0 to 255); requires
C--->  a function LSHIFT which performs a logical shift  argument 1
C--->  is value to be shifted, argument 2 is shift count (positive
C--->  for left shift, negative for right shift)
C
      IMPLICIT INTEGER (A-Z)
C
      BLK = LSHIFT ( ADDR, -9 ) .AND. "177
      WORD = ( LSHIFT ( ADDR, -1 ) .AND. "377 ) + 1
      RETURN
C
      END
```

Figure 2: Program to create LB.SYS from LB.SAV

gram which first (1) performs the appropriate checks and (2) invokes the Fortran OTS initialization code. This Macro-11 main program must also define three global symbols which would normally be defined by the Fortran main program to reflect the options specified by the /SWAP (or /NOSWAP), /UNITS, and /RECORD switches, and a global symbol $$OTSI (in PSECT OTS$I) which is used at initialization time to set the default USR swap address. [Note: A Fortran main program also defines global symbols $$OTSC, $$OTSO, and $RF2A1; it appears that none of these are used by the Fortran OTS.] In addition, the Fortran OTS impure area pointer may need to be context-switched (see below). For an example of a typical Macro-11 main program, see figure 3.

INCLUDING THE FORTRAN OTS IN A RESIDENT LIBRARY

Nearly all the routines which make up the Fortran object-time system are reentrant, and are therefore candidates for inclusion in a resident library. There are, however, a few difficulties which must first be overcome.

It's clearly necessary that a resident library contain only pure code and pure data. The only impure parts of the Fortran OTS (other than the OTS work area, channel table, and so forth, all of which have space allocated from the OTS dynamic area at initialization time) are $AOTS (which contains the address of the OTS work area) and $ERRTB (a table used by the OTS error handler). One must ensure, therefore, that the modules which define $AOTS and $ERRTB are kept out of the resident library.

The Fortran OTS routines can be divided into three distinct categories:

(1) Those routines which, if included in a resident library, would cause the module which defines $ERRTB to be included
(2) Those which would force $AOTS to be included, but not $ERRTB
(3) All other routines

Everything in group 3 can be included in a resident library using the techniques outlined above.

Routines in group 2 would normally be precluded from being part of a resident library. However, if one can arrange to have $AOTS context-switched, routines from category 2 can be included. Since an address within the resident library itself would not be eligible for context-switching, it's necessary to reserve a location in low memory (an unused interrupt vector, for example) for $AOTS, include a definition for $AOTS in LB.MAC, and perform a .CNTXSW before the OTS initialization routine is invoked. [Note: It may also be necessary to protect this word by patch-

```
        .TITLE  .MAIN.

; Archetypal main program for resident library usage

; Define "FORTRN" for Fortran jobs
; Define "FAOTS" if $AOTS context-switching is required

FORTRN=0
FAOTS=0

.IIF DF FAOTS   FORTRN=0

        .MCALL  .DSTAT,.CNTXSW,.PRINT,.EXIT

        .GLOBL  LB$ADR  ; Expected address of loaded LB handleroid
        .GLOBL  LB$VER  ; Expected LB handleroid version number

.IF DF FORTRN
        .GLOBL  $$OTI   ; Fortran OTS initialization routine
        .GLOBL  $EXIT   ; Fortran exit routine
.ENDC

.IF DF FAOTS
        .GLOBL  $AOTS   ; Address of Fortran OTS impure area pointer
.ENDC

        .GLOBL  MAIN    ; Subroutine which was formerly main program

.IF DF FORTRN
.IIF NDF USRSW  USRSW=0     ; Default is /SWAP
.IIF NDF NLCHN  NLCHN=6     ; Default is /UNITS:6
.IIF NDF LRECL  LRECL=132.  ; Default is /RECORD:132.

$USRSW  ==      USRSW   ; USR swap flag ("/SWAP" --> 0, "/NOSWAP" --> 1)
$NLCHN  ==      NLCHN   ; Number of I/O channels needed ("/UNITS")
$LRECL  ==      LRECL   ; Maximum record size ("/RECORD")
.ENDC

START:  .DSTAT  #DSTBLK,#LBDEV  ; Has the LB handleroid
        MOV     H.ADDR,R1       ;  been loaded?
        BEQ     NOLB            ; If not, go complain
        SUB     #6,R1           ; Point to start of handleroid
        MOV     R1,R2           ; Point to end of handleroid (to get
        ADD     H.SIZE,R2       ;  word containing prologue size)
        ADD     -6(R2),R1       ; Bump past null handler code
        CMP     (R1),#LB$VER    ; Is it the correct version?
        BNE     BADVER          ; If not, go complain
        CMP     R1,#LB$ADR      ; Is handleroid at correct address?
        BNE     BADADR          ; If not, go complain

.IF DF FAOTS
        MOV     #CNTXSW,R0      ; Context-switch the
        .CNTXSW                 ;  impure area pointer
.ENDC

.IF DF FORTRN
        JSR     R4,$$OTI        ; Initialize the
        .WORD   PRONAM          ;  Fortran OTS
        MOV     #ARGBLK,R5      ; Set up Fortran argument pointer
.ENDC

        JSR     PC,MAIN         ; Call the (formerly) main program

; (We should never get here, but just in case .....)
.IF DF FORTRN
        JMP     $EXIT           ; Go exit
.IFF
        .EXIT                   ; Goodbye .....
.ENDC

NOLB:   .PRINT  #NLBMSG         ; Print error message
        .EXIT                   ;  and exit

BADVER: .PRINT  #VERMSG         ; Print error message
        .EXIT                   ;  and exit

BADADR: .PRINT  #ADRMSG         ; Print error message
        .EXIT                   ;  and exit

LBDEV:  .RAD50  /LB /           ; Handleroid device name (Radix-50)

DSTBLK: .BLKW   4               ; Block to receive .DSTAT information
H.SIZE  =       DSTBLK+2        ; .DSTAT returns handler size here,
H.ADDR  =       DSTBLK+4        ;   and <handler address> + 6 here
.IF DF FAOTS
CNTXSW: .BYTE   0,33            ; EMT argument block for
        .WORD   .+2,$AOTS,0     ;  context-switching $AOTS
.ENDC

.IF DF FORTRN
PRONAM: .RAD50  '.MAIN.'        ; Name of new (this) main program

ARGBLK: .WORD   0       ; Arg block for calling old main program

        .PSECT  OTS$I,RW,I,LCL,REL,CON
$$OTSI::

        .PSECT
.ENDC

BEL     =       007     ; ASCII "Bell" character

NLBMSG: .ASCIZ  <BEL>"LB not loaded"<BEL>
VERMSG: .ASCIZ  <BEL>"LB version incorrect"<BEL>
ADRMSG: .ASCIZ  <BEL>"LB address wrong"<BEL>

        .END    START
```

Figure 3: Main program used with resident libraries

ing the monitor to set the appropriate bit in RMON's low memory bitmap. Conflicting advice has been received as to whether this actually needs to be done. In any case, it can't hurt.]

Routines in category 1 may not be included in a resident library under any circumstances. Unfortunately, nearly all of the Fortran I/O routines are in this group. This need not be the case, however. It's a result of some non-modularity in the way that the Fortran OTS is put together: Most Fortran I/O routines require a module named EOL; this module calls a routine named $$SET which lives in the OTS initialization module, which, in turn, references $ERRTB. Inclusion of a Fortran I/O routine in a resident library would therefore force $ERRTB to be included.

$$SET was apparently included in OTI since this module contains the Fortran error handler, which also uses $$SET. There's no particular reason, though, that $$SET need reside there. If $$SET is moved off to a module of its own, the Fortran I/O routines become part of category 2, and may then be included in a resident library, provided that $AOTS is context-switched. [Note: Unfortunately, this requires that one have source code for the Fortran OTS. The programs MAKELB and LBMAIN will be submitted to the DECUS library and/or will be included on the Fall 1979 RT-11 SIG tape. It's hoped that an arrangement can be made with Digital whereby replacement object modules for OTI and $$SET could be included without violating any software licensing agreements.]

There are two other caveats regarding the use of the Fortran OTS in a resident library: First, the fact that there will be very few OTS routines in the user job area could cause problems with USR swapping in

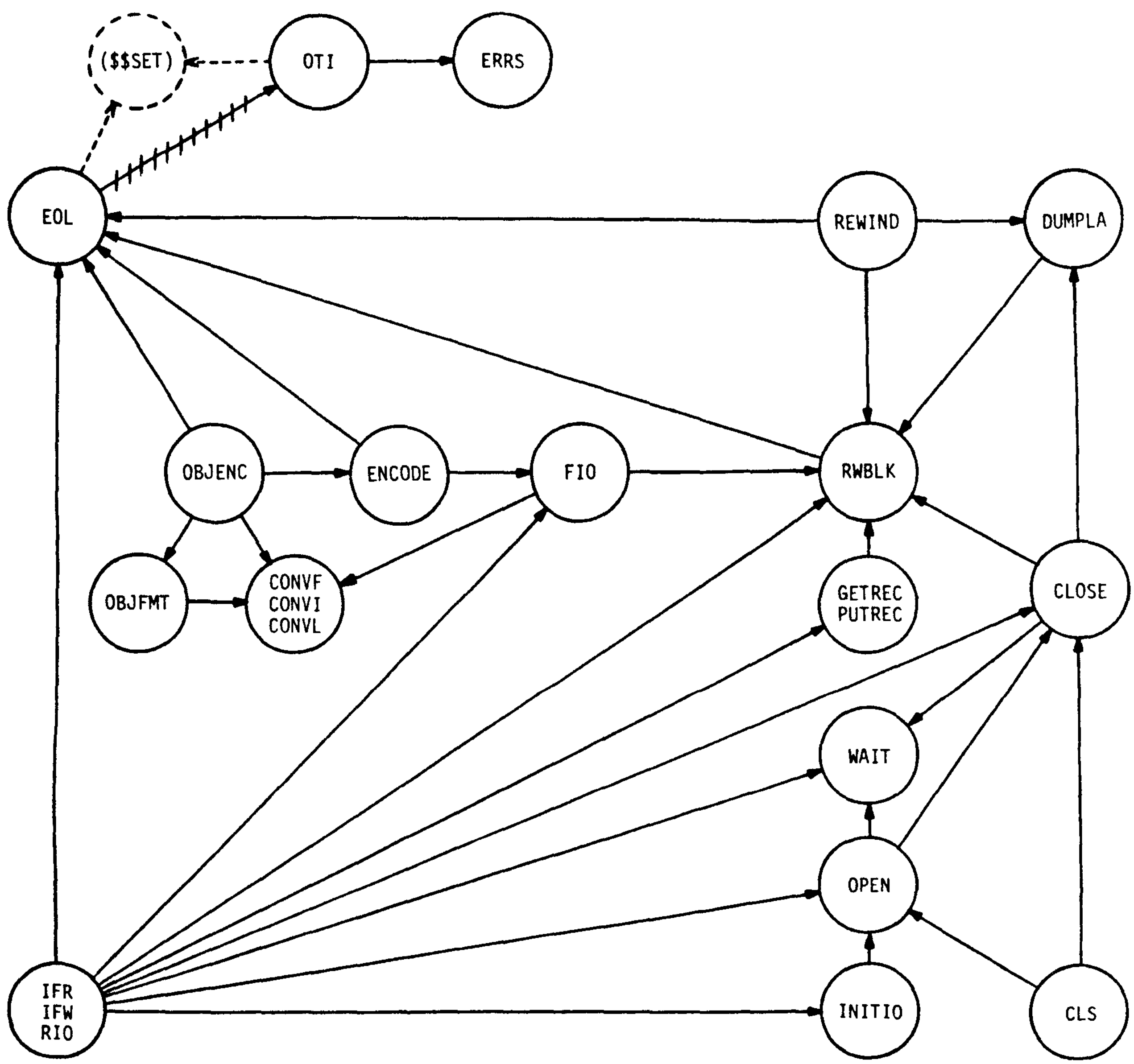


Figure 4: OTS change necessary to permit Fortran I/O routines to be included in a resident library

applications which make use of completion routines and/or certain SYSLIB subroutines. The default USR swap address is defined by global symbol $$OTSI. If $$OTSI is at or near the start of the user program, the result may be that the USR will swap over an argument passed to a SYSLIB routine such as LOOKUP or IENTER. The USR will also probably swap over the OTS initialization module; since this module contains the Fortran error handler, a system crash is likely if an error occurs in a completion routine while the USR is swapped over the user program. Some care should be taken, therefore, to either make the USR non-swapping, or to provide it with a safe area to swap over.

Secondly, there is a PSECT named OTS$P, with attributes RW,D,GBL,REL,OVR. This PSECT contains a table which holds such things as addresses of format conversion routines. Various OTS modules define different entries in this table. If a job which uses a resident library requires a module which (1) references OTS$P and (2) is not contained in the resident library, the user job will have its own copy of this PSECT. Although there are some situations in which this will not cause problems, it's advisable that one avoid letting it happen, unless one has sufficient familiarity with the internals of the Fortran OTS. OTS$P will always appear in the user job's map. If its length is nonzero, however, the resident library should be rebuilt to include the module(s) which reference OTS$P.

Space limitations prevent inclusion of a complete module-by-module list of all Fortran OTS routines with information as to which of the above-mentioned three categories each module belongs to. Such a list will be furnished on request.

[Note: All the applications for which Fortran OTS modules have been included in a resident library have used the version of FORLIB without virtual array support. It is not known which, if any, of the OTS virtual array modules may be included.]

SOME RESULTS

The application for which this approach to implementing a resident library was developed consisted of a 16K foreground job and several background jobs, most of which were approximately 10K words in size. All jobs are written primarily in Fortran, and are heavily overlaid to the degree that additional overlaying would not significantly reduce program size. It was not possible, therefore, to run a background job without removing the foreground job.

To remedy the problem, all reentrant code used by the foreground job was moved into a resident library. Its size is approximately 5K words, of which a typical background job uses about 3K. Since the combined size of the present foreground job and the resident library is only minimally larger than the size of the original foreground job, the reduction in the background jobs' size was sufficient to permit them to be run with the foreground job present:

Without Resident Library			With Resident Library		
RMON	--	4K	RMON	--	4K
			Res. Lib.	--	5K
Foreground	--	16K	Foreground	--	11K
Background	--	10K	Background	--	7K
Total		30K	Total		27K

OTHER APPLICATIONS

A side effect of the use of a resident library is a decrease in the amount of space a program occupies on disk. A 5K word resident library results in a .SAV file size reduction of as much as 20 disk blocks (the amount depends on how much of the resident library code the program actually uses). Because of this, even SJ users (particularly those with RX01-based systems) might find it worthwhile to implement a resident library, simply for the increase in file storage that can be realized.

ACKNOWLEDGMENTS

This method of implementing resident libraries is based on a suggestion from Carl Ralston of Digital.

Work supported in part by NASA Grant NGL-05-002-188, and by the U.S. Department of Energy under Contract EY76-C-03-0068.

Funding Questionnaire

Would you be willing to pay a yearly subscription fee for the newsletter? (a minimum of 6 newsletters per year)

yes____ no____

Comments:

What yearly rate would you be willing to pay?

$6____ $8____ $10____ $12____

Comments:

Will you continue to subscribe to the newsletter if it is no longer free?

yes____ no_____

Comments:

Would you buy back issues if they were made available?

Yes____ no____

Comments:

Please return this questionnaire to: Ken Demers
MS-48
United Technologies Research Center
Silver Lane
East Hartford, CT 06108

DECUS

DIGITAL EQUIPMENT COMPUTER USERS SOCIETY
ONE IRON WAY, MR2-3/E55
MARLBORO, MASSACHUSETTS 01752

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

MOVING OR REPLACING A DELEGATE?

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

( ) Change of Address
( ) Delegate Replacement

DECUS Membership No.: ______________

Name: ______________

Company: ______________

Address: ______________

____

State/Country: ______________

Zip/Postal Code: ______________

Mail to: DECUS - ATT: Membership
One Iron Way, MR2-3
Marlboro, Massachusetts 01752 USA

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