RSX

MULTI-TASKER

November 1984 Issue

"The Following are trademarks of Digital Equipment Corporation:

DEC
DECnet
DECmate
DECsystem-10
DECSYSTEM-20
DECUS
DECwriter
DIBOL
Digital logo
EduSystem
IAS
MASSBUS
PDP
PDT
P/OS
Professional
Rainbow
RSTS
RSX
RT
UNIBUS
VAX
VMS
VT
Work Processor

UNIX is a trademark of Bell Laboratories.

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

RSX MULTI TASKER

Table of Contents

From the Editors

We are currently in the middle of a drought. We need contributions to make this Newsletter work. With the upcoming Symposium (which will have happened by the time this is read), we are hoping for a rush of material from the RSX community - from both users and Digital. Some ideas for contributions are: portions of trip reports which most of you will write, product demonstrations, an interesting idea raised during informal gatherings, etc...

If each member of the RSX Community contributed to the Multitasker once every ten years, we would have to multiply our page count many times. Please, don't be stingy. Share what you know with others.

Send submissions to:

Dominic DiNollo
Loral Electronic Systems
Engineering Computer Center
Ridge Hill
Yonkers, New York 10710

14 South 1 PPG Place
30 April 1984

Dear Multi-Tasker Editor:

This submission falls under the heading "programming hints and kinks": I have two indirect command file coding techniques that I'd like to pass on to those who haven't discovered them yet.

I. Parameterizing Indirect Command File Environmental Dependencies

I've found it helpful to "parameterize" all command file environmental dependencies near the head of my command file, letting string variables function like assembler-language equates. That way, I can get at the "magic constants" quickly, instead of rooting through command files when, say, trying to cannibalize a command file for a different application. Items that I parameterize include -

o ...AT. logical unit numbers
o file names
o uics
o device names
o ...MAC switches

II. Indirect Command File Switch Parameters

One method for adding user-accessible switch-processing capability to indirect command files is to treat the initial command file parameter as a string of one-character-long switch specifications if the parameter is preceded by a plus sign ("+"), adjusting subsequent parameter evaluation accordingly. One possible implementation of this paradigm:

```
        .SETS   PARM01 P1       ! initially, assume the switch
        .SETS   PARM02 P2       !   string is absent
        ..........
        .SETF   ASWIT           ! initially, assume that
        .SETF   BSWIT           !   no switches are active
        ..........

.; test for initial switch string; absent if leading char
.;  of the first parameter isn't a "+"

        .TEST   P1 "+"          ! test for switch indicator
        .IF <STRLEN> <> 1 .GOTO PRMDON
```

```
                .SETS   PARM01 P2       ! P1 is a switch string;
                .SETS   PARM02 P3       !  re-evaluate parameters

                .TEST   P1 "A"                  ! "A" switch selected?
                .IF <STRLEN> <> 0 .SETT ASWIT           ! yes; set flag
                ...........
        .PRMDON:
```

I try to provide the following switches, where appropriate, for my task-building command files -

A(uto) - reassemble all program objects automatically, then proceed directly to taskbuild (default: user queried before each re-assembly or re-compilation)

D(ebug) - include ODT in the task object

L(ist) - produce listings with assemblies/compilations

T(ask) - assemble no programs; proceed directly to taskbuild (supersedes A(uto) mode)

Respectfully,

Phillip Pfeiffer

systems analyst
PPG Industries
Glass Engineering

14 South 1 PPG Place
30 April 1984

Dear Multi-Tasker Editor:

The Fox 2/30 is a PDP-11/10-based process control minicomputer with limited program development capabilities, i.e.:

- single directory
- numbered files (file "names" typically range from 1 to 1024)
- single file backup via paper tape
- PAL-11R-like assembler
- older, character-oriented editor (early TECO?)
- an unsafe assembly language debugger (crashes possible)
- no command language processor

PPG has developed software for supporting Fox 2/30 development under RSX11M which it wishes to make available through the user-community on an unsupported basis:

bi-directional text transfer between RSX11M and the Fox 2/30

- assumes the full duplex driver and a Fox programmer terminal
- requires no file storage on the Fox 2/30
- includes assembler and FORTRAN language comment strippers

Fox 2/30 simulation environment running under RSX11M

- simulates the following Fox system calls -
  - assembler and FORTRAN file i/o
  - program request
  - register save/restore
  - time of day (partial)
- simulates the following Fox utilities -
  - the file creation utility
  - the Fox program scheduling utility

DEC-resident Fox 2/30 procedure-driving utilities and command files

- procedure-driving utilities (all command-line driven) -
  - halt currently running program
  - assemble Fox program
  - compile Fox FORTRAN program
  - catalogue Fox object module in library

    - create, delete, rename, truncate Fox files
    - give Fox files "permanent" status
    - repack the Fox drum
  - procedure-driving command files (all parameter-driven) -
    - build a Fox object library from RSX11M sources
    - link Fox object modules
    - load Fox linker output
  - "General purpose" Fox 2/30 subroutine library
    - sensible reworking of many Fox 2/30 system calls
    - enhanced Fox 2/30 file i/o, string formatting capability

Some of the software in this package may be useful for other RSX11M applications, i.e. -

- an underlying library of assembly language routines for -
  - file i/o
  - command line processing
  - error message handling
  - emit status directive processing
  - i/o via the full duplex terminal driver

- a channel test utility, which -
  - reads parameters from a command-line-specified input file -
    - a string
    - a tty number (full-duplex TT driver assumed)
    - a timeout parameter
    - an echoback (char) count
  - writes the string to the specified tty
  - captures echoback, subject to timeout/char count restraints
  - stores echoback in a command-line-specified output file
  - provides special syntax non-printing characters
  - comes in two versions, for seven- or eight-bit echoback
  - indirect command file provided for send/receive support

- the text transfer programs, which -
  - being well-partitioned, could be reworked for other systems

I would like to make this software available to other Fox 2/30 users, but there is no Fox 2/30 user's group. I do not want to distribute this software through the symposia tapes, for two reasons --

- most of the Fox-related work might not interest the RSX community
- my software cataloguing system conflicts with standard DECUS
  library conventions (see previous letter)

and therefore wish to offer the programs through the Multi-Tasker.

Respectfully,

Philip E Pfeiffer

Phil Pfeiffer
systems analyst
PPG Industries
Glass Engineering

Library Updates

new
11-SP-65

Symposium Tape From The RSX-11 SIG, Spring 1983, St. Louis

Version: Spring 1983

Author: Various

Submitted By: James K. Neeland, Hughes Research Labs, Malibu, CA

Operating System: IAS, RSX-11M, RSX-11M-PLUS

Source Language: FORTRAN IV, MACRO-11, TECO, DCL

Other Software Required: In most cases the software in this package is self-contained. Occasionally it references software on a prior RSX SIG tape.

This RSX SIG Tape contains approximately 60,000 blocks in two backup sets, of which the 2nd is a 22,000 block collection of printer pictures. The first backup set contains nearly 2000 files, including the following: new versions of RUNOFF, PORTACALC, multi-file virtual disk and memory disk drivers, CCL, SRC, and the TED editor. Also, there is: a device-independent graphics package with support for several terminals and plotters; a FORTRAN-callable FFT subroutine; a tutorial on RSX; a task image zapper; an appointment calendar scheduler; help files for DECUS utilities; Semaphores support for RSX11-M,M+; updated utilities to undelete files or reincarnate files with bad file headers. There are new versions of some video terminal games, and some video movies. There is a multi-column lister, a tutorial on EDT initializer files, a MACRO prefix file for 8080/8085 assemblies using the MACRO assembler, etc., etc.

No guarantees are made as to the completeness, usability, or quality of the programs on the tape. The material has not been checked or reviewed, and documentation may or may not be included.

Restrictions: Generally full sources are supplied. However, some programs may not have full sources.

Documentation and/or sources may or may not be included on the magnetic media.

Media (Service Charge Code): 2400' Magtape (PS)

Format: BRU V3.2

Keywords: Symposium Tape -
RSX-11, PORTACALC
Operating System Index:
RSX-11/IAS

August 20, 1984

revision
11-480

HEX: Hex File Editor

Version: August 1984

Author: Kevin Angley, Telex Computer Products, Raleigh, NC

Revising Author: Scott E. Smith, Telex Computer Products, Raleigh, NC

Operating System: RSX-11M V4.1, VAX/VMS V3.6 in compatibility mode.

Source Language: MACRO-11

The HEX utility is designed to manipulate ASCII hex formatted files as output by cross-assemblers and linkers for microprocessors (Z80, 8085, 68000, etc.). HEX supports all of the popular ASCII hex formats: INTEL, TEKHEX, Extended TEKHEX, Motorola, and MOSTEK. The principle components of the HEX utility is a 32K byte virtual memory area that allows the user to manipulate code in memory as if it were in the target machine. Note that the 32K byte virtual memory can be extended to cover the entire 2**32 address range via the OFFSET command. Hex will handle 16, 24, or 32-bit addresses. Operations that can be performed on a hex file in virtual memory:

COMPARE - compare specified addresses to another hex file.
COMPLEMENT - perform a Logical (1's) complement on a range.
COPY - copy from one area to another.
CRC - compute the standard cyclic redundancy check.
DISPLAY - display to the screen or to a listing file.
EDIT - examine and optionally replace values.
FILL - fill a range with a value.
FORMAT - establish the ASCII hex format.
IDENT - display the HEX program identification.
INIT - resets the utility to initial conditions.
MOVE - move byte or word values.
NEGATE - perform the arithmetic (2's) complement.
OFFSET - establish offset.
READ - read a hex file into virtual memory.
SEARCH - search the specified range for a byte or word value.
SUM - compute a 16-bit byte-wise summation.
TRANSFER - set the transfer address.
WRITE - write a range of virtual memory to a hex file.

Changes and Improvements: Many new commands, expanded functionality of old commands, supports many more hex formats.

Documentation on magnetic media.
Media (Service Charge Code): 600' Magtape (MA)
Format: BRU V4.1

Keywords: Microprocessor, File Management, RSX-11 - Utilities
Operating System Index: RSX-11/IAS

new
11-756

FORTRAN IV-PLUS and FORTRAN-77 On-Line Debugger

Version: June 1984

Revising Author: Gabor D. Miklos, Memorial Sloan-Kettering Cancer Center, New York, NY

Operating System: RSX-11M V3.2

Source Language: MACRO-11

Memory Required: 1000 KW

This program is a revision of the FODT On-Line Debugging Tool, originally written by David Beckwith (DECUS Program Number 11-270A). By using this modified dubugger, users can debug any application program written in FORTRAN IV-PLUS or FORTRAN-77 under RSX-11M, provided the task is NOT overlayed.

A detailed document can be found in the file 'FODT.DOC", which is on the media.

Note: Release notes are distributed with each medium.

Restrictions: At present program cannot debug overlayed tasks. It is difficult to reliably mark the beginning and especially the end of the executable code part of the loaded segment.

Documentation on magnetic media.

Media (Service Charge Code): DECtape (HA), Floppy Diskette (KA), 600' Magtape (MA)

Format: DOS-11

Keywords: Debugging
Operating System Index:
RSX-11/IAS

September 3, 1984

new
11-SP-67

Symposium Tape from the RSX SIG, Spring 1984, Cincinnati

Version: Spring 1984

Author: Various

Submitted By: Glenn C. Everhart, Ph.D.

Operating System: IAS, RSX-11D, RSX-11M, RSX-11M-PLUS, VAX/VMS, P/OS

Source Language: BASIC-11, VAX-11 BASIC, BASIC-PLUS, C, FORTRAN IV, FORTRAN IV-PLUS, FORTRAN 77, VAX-11 FORTRAN, MACRO-11, MACRO-32, PASCAL

This is the RSX Spring 1984 (Cincinnati) symposium tape. For the convenience of VMS users it is available in either BRU format or VMS Backup format at 1600 BPI. The DECUS part no. for the VMS tape is V-SP-27.

The tape contains approximately 60,000 blocks of code, including the following highlights:

Michael Reese BASIC, ATT, DEV, Force command line, DOS cross supports, virtual disks (memory and disk resident), spreadsheet, calendar, Runoff from Rice University, spelling checker, several communications utilities including updates and extensions of XMITR and DUPLEX, graphics, full Kermit distribution (as of 7/15/1984), BUG screen debugger for IAS, LBL Tools toys (LEX, YACC, LISP, RATFOR, AR, etc.), DECUS C for PRO350, phone list manager, mailing list manager, sorter, HEX file mgr, Task Image Zap, and numerous PRO350 loadable task images and runtimes including Swedish PASCAL, BASIC, DTC, AnalytiCalc (spreadsheet), DDT (sources for symbolic debugger), TECO, SRD, COPY, LISTRSX, and much more. Since many of these programs are as useful on VAX as on RSX, they are provided.

No guarantees are made as to the completeness, usability, or quality of the programs on the tape. The material has not been checked or reviewed and documentation may or may not be included.

Note: Most programs include complete sources; a few do not.

Complete sources not included. Documentation on magnetic media.

Media (Service Charge Code): 2400' Magtape (PS)

Format: BRU V3.2 (1600 bpi ONLY)

Keywords: Symposia Tapes - RSX-11
Operating System Index: RSX-11/IAS, VAX/VMS, P/OS

September 17, 1984

revision
11-5Ø7

SOS: A Program for Saving Deleted Files

Version: V2.1, July 1984

Author: James F. Carter, UCLA, Tokamak Fusion Laboratory,
Los Angeles, CA

Operating System: RSX-11M V4.Ø

Source Language: MACRO-11

Memory Required: 8KW

Other Software Required: F11 ACP Works only on FILES-11 Volumes.

When you accidentally delete a file on a FILES-11 volume, the data and file header are still intact, until re-used for another file. You can recover the data using SOS. It works much like PIP, copying the data to a new file on another unit. You can use wildcards in the input file specification, and you can specify a file owner other than yourself. If you omit the version you get the latest version of the file; version -1 gives you the earliest version. SOS can also read non-deleted files. SOS checks file protection on the input file and output directory.

Restrictions: The program is supposed to work on multi-header files, but this feature could not be tested.

Documentation on magnetic media.

Media (Service Charge Code): Write-Up (AA), Floppy Diskette (KA),
6ØØ' Magtape (MA)

Format: FILES-11

Keywords: File Management,
RSX-11 - Utilities
Operating System Index:
RSX-11/IAS

September 17, 1984

revision
11-719

PLANS: A Program to Plot Hardcopy on the Versatec Printer

Version: V2.0, June 1984

Author: Thomas E. Chenault, U.S. Government, White Sands Missle Range, NM

Operating System: RSX-11M V3.2 - V4.1

Source Language: FORTRAN 77

Memory Required: 32KB

Other Software Required: Versatec's Versaplot Library

Special Hardware Required: Versatec Printer-Plotter

The purpose of the program Plans is to produce a hardcopy on the Versatec Printer of plots, specifically, design drawings. (more detailed documentation is in PLANS.DOC)

PLANS utilizes user supplied data files, then converts the easily organized data files into complicated graphics output, using Versatec's Versaplot Graphics Library. Each data file plot is used as a tranparent overlay. The final plot consists of those plot overlays chosen by the user.

PLANS offers easily organized and coordinated user controls of:

1. MOVE(s) and DRAW(s) to all position(s).
2. Mixtures of multi-plotting(s), multi-rotation(s), multi-sizes with multi-line width(s) of
    1. 14 predefined symbols,
    2. 113 predefined character codes,
    3. 6 predefined fill patterns,
    4. All regular multi-sided figures,
    5. 871 user defined figures,
    6. 1000 labels.

Since dialog is enabled between the user and Versaplot, the scale of the whole drawing, and 27 other variables can be controlled.

Changes and Improvements: Correction to Translation and Rotation routine, change to fill routine, addition of ability to save processed files. Valid on 5 Operating Systems.

Restrictions: On a VAX, the installer would need to eliminate the overlays used in task building. Sources to Versatec's Versaplot Library are not included.

Associated Documentation: Versatec's Versaplot Operating Manual.

Complete sources not included. Documentation on magnetic media.

Media (Service Charge Code): Floppy Diskette (KA),
6ØØ' Magtape (MA)

Format: FILES-11

Keywords: Graphics, Plotting,
Conversion
Operating System Index:
RSX-11/IAS

September 17, 1984

revision
11-731

Kermit-11

Version: V2.20, August 1984

Author: Brian Nelson

Submitted By: Rebecca Dent, Change Software Inc., Toledo, OH

Operating System: RSTS/E V7.2 or later, RSX-11M V4.0 or later,
RSX-11M-PLUS V2.1, RT-11 V4.0 or later

Source Language: MACRO-11

Memory Required: 20KW

Kermit is a protocal originally developed at Columbia University which has been used to implement error free packet file transfer and communications between computer systems, both mainframe to mainframe and micro to mainframe.

This Kermit-11 was developed by the author for RSTS/E, RSX-11M-PLUS, RSX-11M and RT-11.

Kermit-11 will run on RSX-11M version 4.0 and RSTS/E version 7.2 as long as the task was built without using RMSRES. To be able to build Kermit on RSTS/E version 7.2 of RSX-11 version 4.0 you will have to get RMSLIB.OLB and MAC.TSK from RSX-11 V4.1 or RSTS/E V8.0. The need for version 2 of RMSLIB is due to the use of $SEARCH, $PARSE, $RENAME and $DELETE. The need for the newest MAC.TSK is due to the use of new directives such as .SAVE, .RESTORE and .INCLUDE /FILENAME/.

Note: For RSTS/E system users please note that you do not have to create RMS files as output. You can instead either type set record-format stream, modify the default in K11RMS, MAC or put the set command in one of the following files:

```
SY:KERMIT.INI
LB:[1,2]KERMIT.INI
SY:[1,2]KERMIT.INI
KERMIT:KERMIT.INI
```

Restrictions: Minimum System requirements to ASSEMBLE and LINK KERMIT:

RSTS/E V8.0 or later, with multiple private delimiters and RMS V2
RSX-11M V4.1 or later, with full duplex terminal driver and RMS V2
RSX-11M-PLUS V2.1 or later, with full duplex terminal driver and RMS V2. RT V5 will not run under RT-11 SJ. Needs FB or XM.

Minimum system requirements to RUN KERMIT:

RSTS/E V7.2 or later, with multiple private delimiters
RSX-11M V4.0 or later, with full duplex terminal driver
RSX-11M-PLUS V2.0 or later, with full duplex terminal driver
RT-11 V4.

Documentation on magnetic media.

Media (Service Charge Code): 600' Magtape (MA)

Format: DOS-11

Keywords: RSX-11 - Utilities, RSTS/E - Utilities, File Transfer, RT-11 - Utilities
Operating System Index: RSTS/E, RSX-11/IAS, RT-11

September 17, 1984

new
11-744

PARSE/RSX: A Flexible FORTRAN 77 Filespec Parser

Version: V3.0X, May 1984

Author: Ralston W. Barnard, Sandia National Laboratories,
Albuquerque, NM

Operating System: RSX-11M

Source Language: FORTRAN 77

Memory Required: 770 (decimal) Words

PARSE is a FORTRAN-77 subroutine which simplifies the specification of files used in a program, and reduces operator input. When used with a FORTRAN applications program, PARSE minimizes the amount of typing necessary to specify files by providing default values whenever they are not supplied at run-time. Defaults may be provided for devices, UIC's, and extensions. Version numbers are also accommodated. Once a "root" filespec has been created by PARSE, further filespecs can be created without any user action by means of successive calls to PARSE.

The code will currently support RSX UIC's or VMS directory manes of up to seven characters. It could be easily modified to handle VMS directory specifications of any length. Furthermore, the code could be extended to accept DECnet nodes or VAX device specifications.

The program TSTPAR demonstrates the use of PARSE, and also is a tutorial on the proper use of the subroutine. This version of PARSE has been written to take advantage of FORTRAN-77 language features and constructs.

Documentation on magnetic media.

Media (Service Charge Code): Floppy Diskette (KA),
600' Magtape (MA)

Format: FILES-11

Keywords: Tools - Application Development, RSX-11 - Utilities
Operating System Index:
RSX-11/IAS

Spetember 17, 1984

new
11-745

Spelling Checker with Dictionary Maintenance Utility

Version: V3, January 1984

Author: Alan Dunwell, University of Colorado, Boulder, CO

Submitted By: Judah Levine, University of Colorado, Boulder, CO

Operating System: RSX-11M V4.1

Source Language: FORTRAN 77, MACRO-11

Memory Required: 31KW

SP3 is a general purpose spelling checker program for text files. Words in the text file are compared to words in the resident Dictionary file. Failure to find a match in the Dictionary generates an operator prompt to verify the word in question. Correctly spelled words are saved and are merged into the Dictionary file when the entire list has been examined. Misspelled words are saved in file ERRLST. DAT. Abbreviations or other character strings may be skipped or inserted in the dictionary at the user's discretion.

ED3 is a Dictionary Maintenance program which allows direct access to the Dictionary. The operator is allowed the option to READ, ADD, or DELETE any word in the Dictionary. The programs are supplied with a dictionary containing about 4ØØØ words.

Documentation on magnetic media.

Media (Service Charge Code): Floppy Diskette (KB),
6ØØ' Magtape (MA)

Format: FILES-11

Keywords: Dictionary, Spell
Operating System Index:
RSX-11/IAS

September 17, 1984

new
11-747

Dictionary with Phonetics

Version: April 1984

Author: Seymour Shlien, Department of Communications, Ottawa,
Ontario, Canada

Submitted By: Eric Teutsch, POWERSOFT, Ottawa, Ontario, Canada

Operating System: RSX-11M V4.1B

Source Language: FORTRAN 77

Memory Required: 15712 Words

This tape contains a dictionary of over 7000 words with regular and phonetic spelling, as well as grammatical meaning (noun, adverb, etc.). In addition to the dictionary are a number of utilities that manipulate/use this dictionary: MODFIX to modify the dictionary interactively (contains code to listen to phonetics through the TeleSensory Systems PROSE 2000 speech synthesizer).

BLDFIX to build the dictionary.
BLDDAT to unbuild it (convert from direct-access to sequential file).
BIGLST to create a spoolable list file.
STRMAT to find words in the dictionary.

Note: The phonetic entries may be sent to a speech synthesizer by 'Speech-Plus, Inc.'. The interfacing to this uses Q10's.

Restrictions: Database contains singular nouns, not plural, present tense verbs, etc.

Documentation on magnetic media.

Media (Service Charge Code): Floppy Diskette (KA) Format: RT-11,
600' Magtape (MA) Format: DOS-11

Keywords: Speech, Dictionary
Operating System Index:
RSX-11/IAS

September 17, 1984

new
11-75Ø

TEM: A Terminal Emulator for RSX-11

Version: May 1984

Author: Thomas R. Wyant, III, DuPont de Nemours

Operating System: RSX-11M-PLUS V2.Ø or later

Source Language: MACRO-11

Memory Required: 13KW

Special Hardware Required: Dial-out Modem

TEM provides "dumb" terminal emulation over a full duplex TT: line. It allows the user to "become" a terminal on a remote system, and to do ASCII file transfer between systems. TEM has been used to communicate with TSX, VMS, and TOPS-2Ø systems, as well as non-DEC equipment. It requires no software on the remote system (and therefore has no error checking).

In addition to the basic functionality, TEM can automatically issue canned commands to smart modems at the beginning and end of a session. The user can also select from the following features:

1. Local Echo.
2. Automatic line feed on carriage return.
3. Translation of inbound control characters to ASCII abbreviations.
4. Passthru of control/s, control/q, control/o and conrol/x to the remote system.
5. User selectable attention and end-of-file characters.
6. Redefinition of any desired input character to any other.
7. Specifiable delay and prompt character for file transfer.

TEM requires at least RSX-11M-PLUS V2.Ø. RSX-11M V4.Ø with the full-duplex TT: driver, get/set multiple characteristics, and unsolicited input AST's should work, but has not been tested. TEM can be initiated from a terminal on the TT:. HT:, or VT:, driver, though there are restrictions on its use from a virtual terminal. It can communicate with any device on the TT:, HT:, or NL: driver.

Documentation on magnetic media.

Media (Service Charge Code): Floppy Diskette (KA) Format: RT-11,
6ØØ' Magtape (MA) Format: DOS-11

Keywords: Emulators, RSX-11 -
Utilities, Data Communications
Operating System Index:
RSX-11/IAS

September 17, 1984

More to Manipulating the FORTRAN environment

Pavel Muselik
Dopravoprojekt Brno
65830 Brno, Czechoslovakia

In The Multitasker from August 1979 there was a question in Q&A Session: Can one write a macro subroutine to return the number of arguments to the calling subroutine? The answer was: In general, no. In the November 1979 issue was printed a contribution of Bill Newcum, who wrote such subroutine. Other solution of the problem was in The Multitasker from December 1983. Both the subroutines were written only for FORTRAN IV PLUS.

Our program packages are writen primarily in FORTRAN. To determine the number and status of arguments passed to a subroutine we use a subroutine ARGMNT. The versions of the ARGMNT for FORTRAN IV and for FORTRAN IV PLUS (FORTRAN 77) follow. Due to the subroutine ARGMNT our packages of subroutines are relatively small, flexible and easy to use.

FORTRAN IV PLUS (FORTRAN 77) version:

```
        .TITLE ARGMNT
;Subroutine ARGMNT returns number and feature of arguments in call
;of a subroutine SUBR. ARGMNT should be called immediately after
;entering subroutine SUBR.
;
;Author: Pavel Muselik
;
;Call   CALL ARGMNT(NA)           returns the number of actual args
;       CALL ARGMNT(NA,IA)        returns NA and adrs of actual args
;       CALL ARGMNT(NA,IA,N)      returns NA and adrs of all possible
;                                 actual args (even if missing at the
;                                 end of the list)
;
;       NA      number of args in the call of the subroutine SUBR
;       IA      integer array for addresses of the args of the SUBR
;               IA should have at least NA elements (ARGMNT(NA,IA))
;               or N elements (ARGMNT(NA,IA,N)).
;       N       Number of formal arguments of the SUBR (maximum
;               possible number of actual arguments).
;Addresses:     IA(I).EQ.-1  Ith arg is missing (CALL SUBR(A,,,D) )
;               IA(I).NE.-1  Ith arg is used    (CALL SUBR(A,B,C,D) )
;
;Tested on FORTRAN IV PLUS compiler F4P V02-51
;Tested on FORTRAN 77 compiler      F77 V4.0-1
        .IDENT /JUL83/
        .PSECT  $CODE1
ARGMNT::MOV     2(SP),R1        ;save pointer on list of SUBR args
```

```
        CMPB    @R5,#3          ;has ARGMNT three args?
        BNE     C               ;NE = no fill array IA
        MOV     @6(R5),R2       ;dimension N of array IA
        MOV     4(R5),R3        ;address of array IA
D:      MOV     #177777,(R3)+   ;fill array by "not used"
        DEC     R2              ;next element
        BNE     D               ;last? NE = no
C:      MOV     (R1)+,R2        ;# args of SUBR
        MOV     R2,@2(R5)       ;  into NA
        BEQ     E               ;any args? EQ = no
        CMPB    @R5,#1          ;find only NA?
        BEQ     E               ;EQ = yes, nothing else
        MOV     4(R5),R3        ;find adrs of args. Adr of array IA
L:      MOV     (R1)+,(R3)+     ;adr of arg into IA
        DEC     R2              ;next argument
        BNE     L               ;last? NE = no
E:      RTS     PC              ;yes, finished
        .END
```

FORTRAN IV version:

```
        .TITLE ARGMNT
;Subroutine ARGMNT returns number and feature of arguments in call
;of a subroutine SUBR. ARGMNT should be called immediately after
;entering subroutine SUBR.
;
;Author: Pavel Muselik
;
;Call    CALL ARGMNT(NA)            returns the number of actual args
;        CALL ARGMNT(NA,IA)         returns NA and adrs of actual args
;        CALL ARGMNT(NA,IA,N)       returns NA and adrs of all possible
;                                   actual args (even if missing at the
;                                   end of the list)
;
;        NA      number of args in the call of the subroutine SUBR
;        IA      integer array for addresses of the args of the SUBR
;                IA should have at least NA elements (ARGMNT(NA,IA))
;                or N elements (ARGMNT(NA,IA,N)).
;        N       Number of formal args of the SUBR (max possible
;                number of actual arguments).
;Addresses:      IA(I).EQ.-1  Ith arg is missing (CALL SUBR(A,,,D) )
;                IA(I).NE.-1  Ith arg is used   (CALL SUBR(A,B,C,D) )
;
;Tested on FOTRAN IV compiler FOR V02.2-1
        .IDENT /JUL83/
        .PSECT  $CODE
ARGMNT::MOV     SP,R4           ;Save SP
        CMPB    @R5,#3          ;Has ARGMNT three args?
        BNE     2$              ;NE=less than three
        TST     (R4)+           ;three. Consider third arg on stack
        MOV     @6(R5),R2       ;Dimension N of IA array
        MOV     4(R5),R3        ;address of IA array
```

```
1$:     MOV     #177777,(R3)+   ;IA(I)="not used"
        DEC     R2              ;next element of IA
        BNE     1$              ;last? NE=no
2$:     CMPB    @R5,#1          ;has ARGMNT only one arg?
        BEQ     5$              ;EQ=yes, only NA
        TST     (R4)+           ;at least 2. Consider 2nd on stack
5$:     MOV     32(R4),R1       ;# args of SUBR
        MOV     R1,@2(R5)       ;into NA
        BEQ     E               ;no args? EQ=no, no addresses
        CMPB    @R5,#1          ;NA only?
        BEQ     E               ;EQ=yes, nothing else
        MOV     4(R5),R3        ;address of array IA
10$:    MOV     34(R4),(R3)+    ;address of SUBR arg into IA(I)
        TST     (R4)+           ;pointer on next arg
        DEC     R1              ;decrement counter of args
        BNE     10$             ;last address moved? NE=no
E:      RTS     PC              ;yes, last. Finished
        .END
```

```
C Test program for ARGMNT
        CALL SUBR(P1,P2,,,P5,P6)
        END
        SUBROUTINE SUBR(P1,P2,P3,P4,P5,P6,P7,P8,P9,P0)
        DIMENSION IA(10)          !max 10 addresses
        CALL ARGMNT(NA)
        WRITE(5,10) NA
        CALL ARGMNT(NA,IA)
        WRITE(5,10) NA,IA
        CALL ARGMNT(NA,IA,10)     !max 10 actual args
        WRITE(5,10) NA,IA
 10     FORMAT(11I6)
        END
```

Output from F77 program:

```
     6
     6  1214  1218    -1    -1  1222  1226     0     0     0     0
     6  1214  1218    -1    -1  1222  1226    -1    -1    -1    -1
```

Output from FOR program

```
     6
     6  7336  7340    -1    -1  7344  7348     0     0     0     0
     6  7336  7340    -1    -1  7344  7348    -1    -1    -1    -1
```

Sometimes it is possible to use standard RSX system subroutine - GETADR. But GETADR cannot determine the number of actual arguments in call and addresses of any argument beyond the list (after the last actual argument) are not defined (at least in F77 program).

```
C Test program for GETADR      .
        CALL SUBR(P1,P2,,,P5,P6)
        END
        SUBROUTINE SUBR(P1,P2,P3,P4,P5,P6,P7,P8,P9,P0)
        DIMENSION IA(10)        !max 10 addresses
        CALL GETADR(IA,P1,P2,P3,P4,P5,P6,P7,P8,P9,P0)
        WRITE(1,10) IA
10      FORMAT(10I6)
        END
```

Output from F77 program:

```
  1216  1220     0     0  1224  1228  2466  2566     0     0
```

Output from FOR program:

```
  7190  7194     0     0  7198  7202     0     0     0     0
```

DECUS

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

MOVING OR REPLACING A DELEGATE?

Please notify us immediately to 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: ____________
Phone No.: ____________

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

[redacted]