

Pascal News

(FORMERLY PASCAL NEWSLETTER)

NUMBER 11

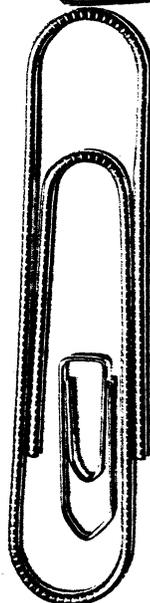
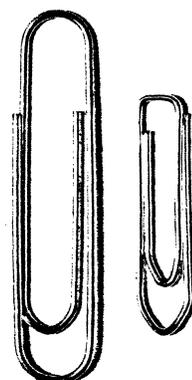
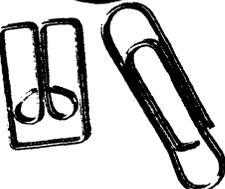
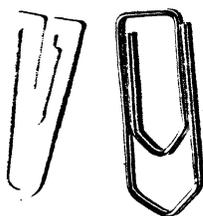
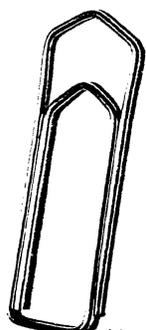
COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS

FEBRUARY, 1978

TABLE OF CONTENTS

COVER: Paper Fasteners from the mail of international Pascalers

0	POLICY: <u>Pascal News</u>
1	ALL PURPOSE COUPON
3	EDITOR'S CONTRIBUTION
4	HERE AND THERE WITH PASCAL
4	News (Jobs, Help Wanted!, Tidbits from Pascalers)
8	Pascal in the News
10	Conferences
10	Books and Articles
13	Errata to <u>Pascal User Manual and Report</u> , Second Edition
16	Review of <u>Pascal Newsletters 5 - 8</u>
19	Roster Increment
33	ARTICLES
33	"Type Compatibility Checking in Pascal Compilers" Pierre Desjardins
34	"A Novel Approach to Compiler Design" James Q. Arnold
36	"Status of UCSD Pascal Project" Kenneth L. Bowles
40	"Suggestions for Pascal Implementations" Willett Kempton
41	"Suggested Extensions to Pascal" Robert A. Fraley
48	"What to do After a While" David W. Barron and Judy M. Mullins
51	"Adapting Pascal for the PDP 11/45" David D. Miller
54	"Pascal: Standards and Extensions" Chris Bishop
57	OPEN FORUM FOR MEMBERS
64	Special Topic: Pascal Standards
70	IMPLEMENTATION NOTES
70	General Information
70	Applications
70	Portable Pascals
72	Pascal Variants
75	Feature Implementation Notes
80	Machine Dependent Implementations
104	Index to Implementation Notes
105	POLICY: Pascal User's Group



POLICY: PASCAL NEWS (77/12/30)

- * Pascal News is the official but informal publication of the User's Group.

Pascal News contains all we (the editors) know about Pascal; we use it as the vehicle to answer all inquiries because our physical energy and resources for answering individual requests are finite. As PUG grows, we unfortunately succumb to the reality of (1) having to insist that people who need to know "about Pascal" join PUG and read Pascal News - that is why we spend time to produce it! and (2) refusing to return phone calls or answer letters full of questions - we will pass the questions on to the readership of Pascal News. Please understand what the collective effect of individual inquiries has at the "concentrators" (our phones and mailboxes). We are trying honestly to say: "we cannot promise more than we can do."

- * An attempt is made to produce Pascal News 4 times during an academic year from July 1 to June 30; usually September, November, February, and May.
- * ALL THE NEWS THAT FITS, WE PRINT. Please send written material for Pascal News single spaced and in camera-ready form. Use lines 18.5 cm wide!
- * Remember: ALL LETTERS TO US WILL BE PRINTED UNLESS THEY CONTAIN A REQUEST TO THE CONTRARY.
- * Pascal News is divided into flexible sections:

POLICY - tries to explain the way we do things (ALL PURPOSE COUPON, etc.).

EDITOR'S CONTRIBUTION - passes along the opinion and point of view of the editor together with changes in the mechanics of PUG operation, etc.

HERE AND THERE WITH PASCAL - presents news from people, conference announcements and reports, new books and articles (including reviews), notices of Pascal applications, history, membership rosters, etc.

ARTICLES - contains formal, submitted contributions (such as Pascal philosophy, use of Pascal as a teaching tool, use of Pascal at different computer installations, how to promote Pascal, etc.)

OPEN FORUM FOR MEMBERS - contains short, informal correspondence among members which is of interest to the readership of Pascal News.

IMPLEMENTATION NOTES - reports news of Pascal implementations: contacts for maintainers, implementors, distributors, and documentors of various implementations as well as where to send bug reports. Qualitative and quantitative descriptions and comparisons of various implementations are publicized. Sections contain information about Software Writing Tools for a Pascal environment, Portable Pascals, Pascal Variants, Feature Implementation Notes, Machine Dependent Implementations, etc.

- * Volunteer editors are:

Andy Mickel - editor
Tim Bonham and Jim Miner - Implementation Notes editors
Sara Graffunder - Here and There editor
John Strait and John Easton - Tasks editors
Rich Stevens - Books and Articles editor
Rich Cichelli - Software Tools and Applications editor
George Richmond - past editor (issues 1 through 4)

PASCAL USER'S GROUP

USER'S

GROUP

ALL PURPOSE COUPON

(77/12/30)

Pascal User's Group, c/o Andy Mickel
University Computer Center: 227 EX
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455 USA

+ Clip, photocopy, or
+
+ reproduce, etc. and
+
+ mail to this address.

// Please enter me as a new member of the PASCAL USER'S GROUP for ___ Academic year(s) ending June 30 _____. I shall receive all 4 issues of Pascal News for each year. Enclosed please find _____ (\$4.00 for each year). (* When joining from overseas, check the Pascal News POLICY section on the reverse side for a PUG "regional representative." *)

// Please renew my membership in PASCAL USER'S GROUP for ___ Academic year(s) ending June 30 _____. Enclosed please find _____ (\$4.00 for each year).

// Please send a copy of Pascal News Number(s) _____. (* See the Pascal News POLICY section on the reverse side for prices and issues available. *)

// My new ^{address} _{phone} is printed below. Please use it from now on. I'll enclose an old mailing label if I can find one.

// You messed up my ^{address} _{phone}. See below.

// Enclosed please find a contribution (such as what we are doing with Pascal at our computer installation), idea, article, or opinion which I wish to submit for publication in the next issue of Pascal News. (* Please send bug reports to the maintainer of the appropriate implementation listed in the Pascal News IMPLEMENTATION NOTES section. *)

// None of the above. _____

Other comments: From: name _____
mailing address _____

phone _____
computer system(s) _____
date _____

(* Your phone number aids communication with other PUG members. *)

JOINING PASCAL USER'S GROUP?

- membership is open to anyone: particularly the Pascal user, teacher, maintainer, implementor, distributor, or just plain fan. Memberships from libraries are also encouraged.
- please enclose the proper prepayment - we will not bill you.
- please do not send us purchase orders - we cannot endure the paper work! (if you are trying to get your organization to pay for your membership, think of the cost of paperwork involved for such a small sum as a PUG membership).
- when you join PUG anytime within an academic year: July 1 to June 30, you will receive all issues of Pascal News for that year unless you request otherwise. You will receive a membership receipt.
- please remember that PUG is run by volunteers who don't consider themselves in the "publishing business." We consider production of Pascal News as simply a means toward the end of promoting Pascal and communicating news of events surrounding Pascal to persons interested in Pascal. We are simply interested in the news ourselves and prefer to share it through Pascal News (rather than having to answer individually every letter and phone call). We desire to keep paperwork to a minimum because we have other work to do.

JOINING THROUGH "REGIONAL REPRESENTATIVES" ?

- To join through PUG(USA), see address on reverse side. International telephone: 1-612-376-7290. PUG(USA) produces Pascal News and keeps all mailing addresses on a common list. Regional representatives collect memberships from their regions as a service and reprint and distribute Pascal News using mailing labels sent from PUG(USA). Persons in the Australasian Region must join through PUG(AUS).

European Region (Europe, North Africa, Middle and Near East):
 send £2.50 to: Pascal Users' Group (UK)
 c/o Computer Studies Group
 Mathematics Department
 The University
 Southampton SO9 5NH
 United Kingdom
 telephone: 44-703-559122 x700

Australasian Region (Australia, New Zealand, Indonesia, Malaysia):
 send \$A10 to: Pascal Users Group (AUS)
 c/o Arthur Sale
 Dept. of Information Sci.
 University of Tasmania
 GPO Box 252C
 Hobart, Tasmania 7001
 Australia
 telephone: (002) 23 0561

RENEWING?

- please renew early (before August) and please write us a line or two to tell us what you are doing with Pascal, and tell us what you think of PUG and Pascal News to help keep us honest. To save PUG postage, we do not send receipts when you renew.

ORDERING BACKISSUES OR EXTRA ISSUES?

Our unusual policy of automatically sending all issues of Pascal News to anyone who joins within an academic year (July 1 to June 30) means that we eliminate many requests for backissues ahead of time, and we don't have to reprint important information in every issue - especially about Pascal implementations!

- Issues 1, 2, 3, and 4 (January, 1974 - August, 1976) are out of print.
- Issues 5, 6, 7, and 8 (September, 1976 - May, 1977) are out of print.
 (A few copies of issue 8 remain at PUG(UK) available for £1 each.)
- Extra single copies of new issues (current academic year) are:
 \$2 each - PUG(USA); £1 each - PUG(UK); and \$A3 each - PUG(AUS).

SENDING MATERIAL FOR PUBLICATION?

(such as ideas, queries, articles, letters, opinions, notices, news, implementation information, conference announcements and reports, etc.) "ALL THE NEWS THAT FITS, WE PRINT." Please send written material for Pascal News single spaced and in camera-ready form. Use lines 18.5 cm wide! Remember: ALL LETTERS TO US WILL BE PRINTED UNLESS THEY CONTAIN A REQUEST TO THE CONTRARY.

MISCELLANEOUS INQUIRIES? Please remember we will use Pascal News as the vehicle to answer all inquiries and regret to be unable to answer individual requests.



UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455
(612) 376-7290

The DEADLINE for written contributions to Pascal News #12 is March 20. Please send DARK copy! New companies committed to Pascal (add to the list in PUGN#9): Ericsson Telephone and ICL in Europe, Interdata and Tektronix in the US. TI continues to be very mysterious about their heavy use of Pascal - they haven't told us a word in a year now! DEC may be finally waking up because of DOD-1 (see Here and There). Thanks to everyone who sent material for this issue. We sent renewal notices to 315 holdouts in November. We may have to stop sending receipts for membership - it is getting too time consuming. We will probably have to combine issues 13&14 next autumn.

Judy sent the letter below to "The Editor, Pascal News":

- Andy



UNIVERSITY OF SOUTHAMPTON

Faculty of Mathematical Studies

Southampton, SO9 5NH. Telex 47661. Tel 0703 559122 Ext 2387

5th December, 1977.

Dear Andy,

PUG (UK) PRINTING and POSTAGE

Now that PUGN 9/10 is out of the way, I thought I would share some statistics on printing costs.

PUGN		USA		UK	
number	pages	per copy	per page	per copy	per page
5	64	\$0.70	1.09c	?	?
6	96	\$1.18	1.23c	\$0.50	.52c
7	48	\$0.69	1.44c	\$0.14	.29c
8	64	\$1.07	1.67c	\$0.40	.62c
9/10	112	?	?	\$0.56	.51c

As you can see, we have managed to keep our costs at well below half yours. After No. 8, we outgrew the Departmental printing service and took 9/10 to the University Printing Unit. They were able to do the job at the same price - in fact slightly cheaper because I did much of the collating myself to hurry it along. However, indications are that there could be a steep rise in costs in the New Year. It may be possible to avoid it, and in order to do so we need to know, accurately,

No. of pages in No. 11
Date of arrival of Masters.

However, should it not be possible to get preferential rates, we shall have to face a cost of about 90c for a 64 page issue (compared to \$1.07 in the U.S.A. and 40c previously). Are you still relying on our ultra cheap rates or can PUG afford to pay the going rate?

Editor's Contribution

Postage for 9/10 was

Country	group	Unit Cost	Previous group	Unit Cost
U.K.	350g	45c	300g	39c
Europe	500g	48c	250g	26c

and came to a total of \$115 (approx.). The mailing included close on 50 renewals received after you ran the labels off. As you can see, it is unlikely that any future issue will hit the lower group for European postage, so that we might have to face 90c + 48c = \$1.38 for getting a copy to a European member. However, with luck we can still do it at under a dollar.

That seems to be all. I have today handed over the files in good order to David who will handle everything after my departure. Following his article in Computing we have had an influx of queries, especially from Industry. Pascal lives!

Thank you very much for all the numerous snippets of information over the past year, and for the most recent ones on South Africa. The personal touch is sincerely appreciated. You will be glad to know that Wits are seriously contemplating switching from Fortran to Pascal for first years (including engineers) in January. It all depends on whether a decent 370 compiler is ready.

I'll keep in touch, of course, and won't forget to send a photo of the wedding.

All the best,

Judy Mullins
Judy Mullins

Professors: H.B. Griffiths, S.A. Robertson (Pure Mathematics); P.T. Landsberg (Applied Mathematics); J.W. Craggs (Engineering Mathematics); D.W. Barron (Computer Studies); T.M.F. Smith (Statistics).

(* It might help explain to new PUG members a few related facts. Judy Mullins last year (76-77) proposed and implemented a reprinting and distribution service of PUGN for PUG members in Europe. Not only was delivery speeded, but also the rates were kept low. Last March, the University of Southampton Computer Studies Group headed by Prof. David Barron held their third annual computing symposium on "Pascal - the Language and Implementation." Both Judy and David have done PUG many great services. Judy graduated this month and is going home to South Africa where she will marry her fiance. David (who by the way edits the ever-popular journal: SOFTWARE - Practice and Experience) recently wrote an outstanding article for the 77/10/24 issue of Computing Europe (a kind of Computerworld for Europe); please see Here and There.

David continues to man the PUG European region and as a result had to quit as a Books and Articles editor for PUGN.

Regarding the question marks under "USA" for PUGN 9/10 in Judy's letter, the costs were \$1.10 and \$0.96. Ken Robinson (see Open Forum) asks for a public explanation of the high (\$A10) cost of PUGN for Australasian members for the new distribution service provided this year by Arthur Sale. Arthur on 77/09/07 sent this information about his estimated costs: \$2.80 for printing per issue (based on the size of #8); postage within Australia = \$0.70. Arthur says that he thinks the cost is "dubiously low" and that \$10 might leave his operation "out of pocket, and to understand the costing, you have to realize that Australia has a high postal charge, and I also am taking on New Zealand." I think it is unfortunate that Arthur's costs are so high, because it is not in the cheap spirit of PUG. Until Ken wrote I didn't know the \$A10 price was relatively "high". Remember though that last year we had severe distribution problems to Australia. I'm grateful to Arthur for volunteering to do the work, and I'm sure he's watching costs. - Andy *)

Here and There With Pascal

NEWS

PASCAL JOBS...

People keep calling us at PUG central asking for people to employ who know Pascal. (*If that isn't evidence of Pascal's viability, I don't know what is!*) With the interest of Pascalers in mind we list here as a service contacts who desire people with compiler experience and knowledge of Pascal:

David Shaw, Structured Systems Corp., Suite 605, 2600 El Camino Real, Palo Alto, CA 94306. (415) 321-8111

Charles Moore, ADP Network Services, 175 Jackson Plaza, Ann Arbor, MI 48106. (313) 769-6800 (also Neil Barta, same address and phone)

Gregory Hopwood, Sperry Univac Mini Computer Division (formerly Varian), 2722 Michelson Drive, Irvine, CA 92713. (714) 833-2400

HELP WANTED!

If Pascal is to make any inroads into serious scientific computing (currently the almost exclusive preserve of FORTRAN) it must have a decent library of scientific subroutines - which means, as far as the U.K. is concerned, that there must be a Pascal version of the NAG (Numerical Algorithms Group) library. (*...and as far as the U.S. is concerned, that there must be a Pascal version of the IMSL (International Mathematics and Statistics Library) library...*)

It should be possible to make a Pascal NAG library largely machine-independent, with all machine-dependent features being collected into the "X" routines. Probably the easiest method of production of the library would be straight transcription of the existing ALGOL-60 versions, together with the writing of the set of "X" routines for each different range of machines.

Please send your views on this matter, and offers of help, to:

Professor D. W. Barron,
Computer Studies Group,
Department of Mathematics,
The University,
Southampton, Hants, SO9 5NH (United Kingdom)

who is coordinating this project and negotiating with NAG.

TIDBITS

D. B. Anderson, 280 Bella Vista Drive, Hillsborough, CA 94010: "I am particularly interested in implementations usable on my company's Interdata 7-32." (* 77/12/12 *)

David B. Anderson, Dept. of Math., Lehigh University, Bethlehem, PA 18015: "By the way, the section in the newsletter called 'Here and There with Pascal' has been very helpful in stimulating the interest of non-believers." (* 77/12/17 *)

Peter A. Armstrong, Digital Data Systems, 1113 Dexter Ave. N., Seattle, WA 98109: "We are immediately interested in information on PASCAL compilers for PDP-11 processors running DEC's RSTS/E monitor. However, we are also interested in any mini- and or

microcomputer PASCAL capabilities." (* 77/12/13 *)

Paul Barr, Raytheon Co., Equipment Div., Boston Post Road, Wayland, MA 01778: "Am attempting to use PASCAL for a Signal Processing application (FFT). Am designing hardware to fit the compiler." (* 77/11/21 *)

Michael Behar, 428 Windy Hill Rd., Orange, CT 06477: "Do you know if there is a version of PASCAL for a MICRO-MIND-II computer (manufactured by ECD of Cambridge, MA)?" (* 77/9/22 *)

Roy E. Bollinger, Dept. 1965, BLD 529, Lockheed, P. O. Box 504, Sunnyvale, CA 94088: "Are there any plans to have any Pascal seminars?" (* 77/11/8 *)

Steven L. Brecher, 5221 Marina Pacifica Dr., N. Key 19, Long Beach, CA 90803: "More generally, I am interested in information on any implementation which can be run on/adapted to a Digital Equipment LSI-11 based system." (* 77/12/19 *)

A. Charles Buckley, Data/Information Systems, Urban Studies Center, Gardencourt/Alta Vista Road, Louisville, KY 40205: "We are currently interested in any work being done to implement Hansen's CONCURRENT PASCAL on a DEC-10 and/or an IBM 370." (* 77/11/28 *)

David Burnett-Hall, Univ. of York, Heslington, York, YO1 5DD, England: "In the discussions on whether array parameters could be dynamic in size, there have been some suggestions that only numerical analysts handling matrices need these facilities. A much more important use, to my mind, is to be able to pass strings of varying lengths. E. g., the DEC-10 compiler has to have 9 almost identical error routines, to handle errors of lengths 15, 20, 25, . . . 55 characters: Stupid." (* 77/8/10 *)

Joe Celko, Box # 11023, Atlanta, GA 30310: "Is there a Nova Pascal sitting around?" (* 77/12/6 *)

Grant M. Colvin, Management Shares, 2121 W. Airport Frwy., Suite 660, Irving, TX 75062: "Do you know of PASCAL implementations for the Hewlett-Packard 3000 series?" (* 77/12/5 *)

C. R. Corner, 514 S. 9th St., Moorhead, MN 56560: "I have a PDP 11/05 and am interested in Pascal activity on the 11 and on micro-based systems." (* 77/09/29 *)

Lawrence S. Cram, 64 Bowen Street, Newton, MA 02159: "Although I am not now a user of PASCAL, I certainly would like to be, and I would like to be on your mailing list. I program commercial applications on a DECsystem-10 in COBOL and am fed up with the limitations inherent in COBOL. I was introduced to PASCAL in Wirth's book, Data Structures + Algorithms = Programs and have followed up with Hansen's Principles of Operating Systems and the PASCAL Users' Guide. I currently have a second-hand bootleg PASCAL compiler and I dabble with it occasionally." (* 77/11/8 *)

Pierre Desjardins, Departement d'informatique, Universite de Montreal, Immeuble principal V-240, Montreal 101 Quebec H3C 3J7, Canada: "My implementation of the Concurrent Pascal machine on Sigma 6 is presently being used to implement (using Concurrent Pascal, of course) the "line access protocol" necessary for communicating with a packet switching service of Bell Canada called DATAPAC. "I am currently involved in the organisation and realization of a primitive distributed microprocessor system. System programs will be written in CP and executed by a CP machine contained in every MP." (* 77/10/13 *)

Robert I. Demrow, 11 Linda Rd., Andover, MA 01810: "I am interested in finding a copy of Pascal that will run on my 8080 computer--presently have 32K and am planning an expansion." (* 77/11/26 *)

John DeRosa, The Boston Systems Office, 400-1 Totten Pond Road, Waltham, MA 02154: "We're presently beginning development of PASCAL systems for micro's as well as resident compilers with the intent of creating a PASCAL well-suited for writing system software. Any news from people working in this direction would be appreciated." (* 77/9/21 *)

George B. Diamond, Diamond Aerosol Corp., RD # 1, Glen Gardner, NJ 08826: "Any other information on PASCAL would be appreciated especially compilers or assemblers for the Z80 CPU." (* 77/12/14 *)

Roberto Dias, 134 Colin Ave., Toronto, Ont M5P 2C3, Canada: "I am a humble owner of a digital group Z-80 minicomputer and as such very interested in learning new languages. I understand that you are publishing four times a year a paper on Pascal. I would very much like to subscribe to it but instead of sending US\$4 right now, I would like to know if the price would be the same for a subscriber in Brazil, as I will be moving to that country in February 78, with my computer." (* 77/11/30 *)

Richard Dievendorff, Dept. 84F, IBM, 620 North Brand Blvd., Glendale, CA 91203: "Although I am having this sent to my business address, this is a personal, hobby venture." (* 77/12/3 *)

Felix F. Dreher, Computer Science, Pittsburg State Univ., Pittsburg, KS 66762: "I am interested in obtaining information about the possible implementation of PASCAL on a small IBM 370/125 machine. Do you have any data suggesting that this has been done? Is there a bootstrap interpreter/compiler available that might be modified for this system? If so, from whom can it be obtained?" (* 77/10/13 *)

William E. Drobish, Silicon Systems, 16692 Hale Ave., Irvine, CA 92714: "Additionally, I would appreciate any information on PASCAL compilers and the availability of one for the Interdata 7/32." (* 77/11/14 *)

C. E. Duncan, 865 Thornwood Dr., Palo Alto, CA 94303: "I am not at present a user, but would like to be one. We have available a number of computing systems, and I would be particularly interested to obtain a running system for IBM 360/370, Univac 1110, Data General NOVA and Intel 8080A. Perhaps not altogether at once; these systems happen to be conveniently available." (* 77/10/24 *)

Randall B. Enger, 28 Briar Patch Lane, Sudbury, MA 01776: "I'm planning to try an implementation on a small machine, mostly because I'm tired of assembly language, but also because I've been away from programming languages stuff for too long. "I like to believe I'm relatively free from 'N.I.H.' disease--'not invented here,' and consequently will eagerly build upon the work of others (borrow from, steal from. . .). Whatever will help me get an implementation going--listings/source on tape/whatever - I'd be willing to spend a few \$ happily (especially if it were to cover copying charges. . .)" (* 77/11/10 *)

Robert B. Finch, 910 N. Lk. Samish Dr. # 30, Bellingham, WA 98225, "My interest is personal/hobbyist computing, and I am currently in the process of implementing Per Brinch Hansen's Sequential Pascal on an Alpha-Microsystems AM-100."

Read T. Fleming, Program in Computer Science, Box F, Brown Univ., Providence, RI 02912: "Brown has an IBM 360/67 running CP-CMS for interactive work, an IBM 370/138 for batch, running VS 1. We have on order a Pascal compiler from the Australian Atomic Energy Commission. When it arrives, we hope to put it up on the batch machine immediately, and at some later date add it to the interactive (CP/CMS) system. "Everybody here is looking forward to Pascal; we hope to use it in a course on compile design next semester, and we're anxious to see how it works in an instructional environment." (* 77/12/18 *)

Jim Fontana, 3519 W. Warner Ave., Santa Ana, CA 92704: "The implementor/maintainer of the 2550/Cyber 18 Pascal compiler is Gordon Wood at CDC LaJolla operation. The compiler is distributed from PSD in Sunnyvale." (* 77/11/2 *)

Ed F. Gehring, Dept. of Computer Science, Math Sciences Bldg., Purdue Univ., West Lafayette, IN 47907: "You guys sure are lax about sending out renewal notices. Most periodicals bombard you with notices for months before your subscription expires. With PUG, you don't get a single notification until 5 months after your subscription expires." (* 77/12/16 *)

Thomas Giventer, 1250 Post Road, Scarsdale, NY 10583, "I am currently working on a PASCAL compiler for the TMS 9900 and would like to find out what other work is being done in this area."

Steven B. Hall and Arthur Dartt, 1599 Orchard Grove, Lakewood, OH 44107 (* address for Hall *): "The installation with which we are professionally affiliated and are students (Cleveland State University) currently is running VS1 on a 370/158. . . . Any help you can give us in differentiating between the various PASCALS will be appreciated (as we do not wish to waste valuable man-hours at vain attempts). We will look forward to hearing from you, as several people are anxious to implement and use PASCAL." (* 77/12/6 *)

Michael E. Harris, 309 W. Edwards # 4, Springfield, IL 62704: "Does anyone have a "full" PASCAL that will work with minor modification on an HP3000 or on an IBM 370/MVS system? Micros? Any computer graphics activity in PASCAL?" (* 77/10/26 *)

Charles Hedrick, Computer Science Dept., Rutgers Univ., Hill Center, New Brunswick, NJ 08923: "Implementors should give some thought to implementing machine-independent representations of data so that data is transportable as well as programs. This involves the generating of files which are not textfiles. What may be the only way out is to use ASCII text representations (using blanks as separators where appropriate)." (* 77/09/20 *)

H. F. Hession, Adv. Record Systems Eng., Gov't. Systems Div., Western Union, 7916 Westpark Drive, McLean, VA 22101: "We are programming Zilog Z-80 microprocessors in assembly language for communications controller applications, and noted an article in the December 1977 issue of BYTE magazine referring to your user group on PASCAL. "None of us has had training in PASCAL, but given the proper documentation, we are confident we can master it." (* 77/12/5 *)

Charles Hethcoat, 2416 Yorktown # 371, Houston, TX 77056: "I obtained a copy of the Pascal PCODE assembler-interpreter with a view to study how it works. I suggest that a worthwhile project would be collect copies of this program as written for a variety of machines and languages, or to write them up for those machines not having a version yet. (This would be especially appealing as a way to implement Pascal on the 8 bit micros). A project like this would go a long way toward assuring that a common language is widely distributed, and a the same time it would simplify life for those wishing to try out language extensions. Also, the PCODE language can be extended to include interruption handling, queues and other real-time techniques for operating system development, as Brinch-Hansen has done with Concurrent Pascal." (* 77/10/10 *)

Robert B. (Buzz) Hill, Eyedentify, Inc., P.O. Box 2006, Longview, WA 98632: "We are a new company in the business of developing and manufacturing custom dedicated microprocessor devices. Our main product, the Eyedentifier, is a microprocessor based image recognition system that utilizes the retinal image as a means of identification (as opposed to a finger print). "Although the Eyedentifier is a simple machine whose software was written in Motorola 6800 assembly language, we anticipate the support products we intend to build for it will require development with a high level structured language. "Recently, a group from my company attended a talk at Lewis and Clark college near here, by Kenneth L. Bowles, UCSD on the PASCAL language. As a result, we are very interested in implementing it on a 6800."

Philip T. Hodge, Habco, P.O. Box 305, Schererville, IN 46375: "As a Z-80 based microprocessor user anxiously awaiting the UCSD version of Pascal, it is heartening to find others who share my opinion of both Basic and Pascal." (* 77/12/6 *)

Ross F. Householder, 1725 Brooks Drive, Arlington, TX 76012: "I am a Pascal user working at Texas Instruments and would like to see what is going on with Pascal in other parts of the country." (* 77/09/11 *)

R. Warren Johnson, Dept. of Math. and Comp. Sci., St. Cloud State U., St. Cloud, MN 56301: "I am seeing more and more hobbyists in beginning courses who need some convincing that PASCAL is real." (* 77/09/15 *)

Here and There With Pascal

Ernest W. Jones, 59 Billou St., San Rafael, CA 94901: "I am interested in certain languages for use on 16-bit micro's and have investigated the MUMPS language. Pascal would no doubt provide more suitable capabilities, but may not be suitable for so small a machine. I would like to learn more about it nevertheless." (* 77/12/81 *)

Mark Jungworth, 13318 Newland St., Garden Grove, CA 92644: "We have recently implemented Pascal on our CDC 7000 machines at McDonnell Douglas in Huntington Beach. I am a complete novice at Pascal usage, but can't wait to BEGIN." (* 77/09/12 *)

Milan Karspeck, 1149 North Michigan, Pasadena, CA 91104: "I am dying to get my hands on a PDP-11 PASCAL implementation. According to the December editorial in Byte Magazine, you ran a list of PASCAL implementations in your issue # 8. I would appreciate it if you could begin my subscription with that issue." (* 77/12/11 *)

Neil T. Keane, Stansaab Elektronik AB, System Development, Veddestavagen 13, Jaarfaalla, Sweden S-175 62: "As a manufacturer of Turn Key Computer Systems we are currently engaged in the assessment of a suitable high level programming language to which we can standardize our in-house programming. Since we are mainly engaged in real time applications we are particularly interested in Concurrent Pascal. To this end, we would like to know the extent to which it has been implemented in the US (apart from the Solo System), and the status of such implementations." (* 77/11/17 *)

Paul Kelly, Educational Data Systems, 1682 Langley Ave., Irvine, CA 92714: "If you are aware of any FORTRAN compilers written in PASCAL which are available at a reasonable cost, I would be quite interested to hear about it." (* 77/10/26 *)

Thomas J. Kelly, Jr., 58-B Meadowlake Drive, Downingtown, PA 19335: "Here at Burroughs I have been using the UCSD implementation of PASCAL for the B6700. It is fairly reliable, although a number of problems have been noted. I have been sending bug reports (most with fixes) directly to UCSD. If anyone is interested in the bugs and/or the fixes, drop me a line. I'll be glad to send listings (most fixes are less than 1 page). "You may also be interested in the fact that a colleague and I have brought up the CDC 6000 compiler at Burroughs (although the code generation has been disabled). This is to allow us to run checks on what constructs were implemented there. We think we have found a bug in it. If so, we'll pass along a bug report." (* 77/11/29 *)

William Kempton, Language Behavior Research Lab, 2220 Piedmont Ave., University of California, Berkeley, Berkeley, CA 94720: "As a linguistic anthropologist, I find the reasons for using Pascal versus other languages fascinating. You find a lot of the same factors operating that operate in any other multilingual speech community. Clearly the merit of the language and the utility of the compiler are only two of many factors affecting language choice, they may not be the most important for most users. The best strategy for a long large change is to have computation center staff at least very familiar with Pascal, and to have it taught in the introductory course in computer science." (* 77/09/22 *)

John Kenyon, Technical Staff, International Computing, 4330 East-West Highway, Bethesda, MD 20014: "We are currently involved in the planning and concept development for the USAF Foreign Technology Division data processing for FY78-82. One of the primary subjects of our study will be the use of standard higher-order systems programming languages within the Department of Defense. I understand that Pascal has been chosen as the candidate language for all DOD and I would appreciate any information you could send me on this subject."

Stephen Klein, 188 Judy Farm Rd., Carlisle, MA 01741: "Computer programming has been a hobby of mine for a few years, mostly in FORTRAN and BASIC, but now I'm sure there are better languages around so I'm also looking into APL and LISP (* besides Pascal *) to get an idea what type of work each language is best suited for." (* 77/12/27 *)

John C. Knight, MS 125A, NASA Langley Research Center, Hampton, VA 23665: "Any interest withing PUG in actively pursuing a PASCAL standard with the National Bureau of Standards?" (* 77/10/20 *)

Henry Ledgard, Comp. and Info. Sci., U. of Mass., Amherst, MA 01002: "We've been inundated with over 50 requests for our prettyprinter and losing money distributing it in the process, too." (* 77/9/15 *)

K. P. Lee, Dept. of Computer Science, 102 Nicholson, Louisiana State Univ., Baton Rouge, LA 70803: "You may be interested to know that we are in the process of getting the Australian compiler. We will be happy to share with PUG any experience we may have with it." (* 77/10/6 *)

Maria Lindsay, Microcomputer Library ! Resource Center, 5150 Anton Dr., Room 212, Madison, WI 53719: "Thank you so much for sending us your brochures and issues. Your newsletter is very impressive. I, for one, now view Pascal as a favorable language. Hopefully it will be available for microcomputers through the manufacturer soon. You can be sure that when we are asked about computer languages, Pascal is mentioned in a very favorable light." (* 77/9/21 *)

Peter Linhardt, 1890 Arch St. Berkeley, CA 94709: "I'm interested in PASCAL for use on a personal system. I understand there is a system that will run on my machine (* TDL Z80 system *)." (* 77/12/9 *)

Ron Mahon, Video Link, 201 N. Main St., P. O. Box 688, Masontown, PA 15461: "Be very interested in any compilers for direct use on a micro, preferably Motorola 6800." (* 77/12/06 *)

John P. McGinitie, P.O. Box 655, Berkeley, CA 94701: "During my education at U.C. Berkeley, I had the honor of learning Pascal as well as Basic, Fortran, Lisp, C, Snobol, . . . Having experienced many languages Pascal has impressed me the most." (* 77/12/17 *)

Michael McKenna, Time Share Corp., Box 683, Hanover, NH 03755: "We are currently using ESI/OMSI Pascal for the PDP 11. We are planning a distributed network using LSI 11's in stand alone mode and with RT 11; the host computer is an 11/60 under RSTS/E - all will be programmed in Pascal." (* 77/12/27 *)

James S. Miller, Intermetrics Inc., 701 Concord Ave., Cambridge, MA 02138: "Today my interest is in finding a solid Pascal compiler for Data General equipment, Novas and/or Eclipses." (* 77/11/10 *)

Roderick Montgomery, Statistical Associate, Health Products Research, 3520 U.S. Route 22, Somerville, NJ 08876: "I would also appreciate receiving information on the availability of back issues for the Newsletter and on PASCAL implementations that produce object code for the Intel 8080 microprocessor. (Either "resident" or "cross" compilers would be acceptable implementations for my purposes, although a "resident" implementation would be preferable.)" (* 77/12/4 *)

Herbert E. Morrison, 1257 2nd St., Manhattan Beach, CA : "I am interested in implementing PASCAL on my Poly 88 (8080) computer. Is there someone you know of who has done this in the Los Angeles area?" (* 77/12/7 *)

G. o'Schenectady, 144 Lancaster St., Albany, NY 12210: "Most pleased that the most rational language ive seen has found 1087 adherants. Read of you-all in Microcomputer SCCS Interface, I id like to be one of you. "My own activity is presently restricted to hardware selection, and i doubt my 8K S-100 system-to-be will support too much of Pascal without additions, but even so it will be good to be in touch with whats happening." (* 77/9/17 *)

David Peercy, BDM Corp., 2600 Yale Blvd. S.E., Albuquerque, MN 87106: "I was previously with Texas Instruments, where Pascal is beginning to flourish." (* 77/12/15 *)

Darrell Preble, Computer Center, Georgia State Univ., University Plaza, Atlanta, GA 30303: "Georgia State University would like to implement Pascal on our Univac 70/7 or barring that, our Interdata 8/32. If you have a working version of Pascal on either of these machines please contact me at the above address. We would like to obtain a working source copy of Pascal for either of these machines." (* 77/11/28 *)

Jerry Pournelle, 12051 Laurel Terrace, Studio City, CA 91604: "A consulting engineering firm is at the moment putting together my Cromemco Z-80, with which I hope to put together some word-processing and small-business bookkeeping--as well as play about. I can see some limits to BASIC, and from years ago when I had my only previous experience with computers I know there are limits to FORTRAN. . . ." (* 77/12/3 *)

Edward K. Ream, 508 Farley Avenue, Apt. 5, Madison, WI 53705: "I am particularly interested in implementations for the 8080 or Z80." (* 77/11/30 *)

Peter Richetta, Computer Science, Slippery Rock State College, Slippery Rock, PA 16057: "I have been trying to get Brinch Hansen's Concurrent PASCAL compiler. After distributing hundreds of systems he stopped distribution. Dr. Hartmann, who wrote much of the compiler, gave me a list of sources to try. Can you help? Any suggestions? "Our computers are NOVA 3 (soft discs) and 370/135 using DOS/VS. Educational use is all we want." (* 77/10/26 *)

Mark Riordan, User Services, Computer Laboratory, Michigan State University, East Lansing, MI 48824: "Here at MSU we are developing (in PASCAL, of course) a word processing system we call Redact. Our CDC 6500 is becoming badly overloaded, so we are considering moving Redact to an Ontel Op-1 intelligent terminal, based partly on the availability of a PASCAL compiler or cross-compiler for an 8080 chip. (We have even considered modifying the venerable CDC 6000 PASCAL compiler to do the trick.) Any input from other microprocessor PASCALers would be appreciated." (* 77/10/24 *)

T. P. Roberts, Kern Instruments, 111 Bowman Ave., Port Chester, NY 10573: "We will soon be accepting delivery of a Nova 3/12 computer with floppy disk and 32K words of memory. I wonder if your PASCAL Users Group has a compiler/interpreter suitable for this machine? An interpreter alone is not of interest to me, but a compiler would be of interest. "If you have such programs, please inform me of the price, core required, and rough comparison of compile times with Data General Fortran." (* 77/09/08 *)

Robert Rogers, 18625 Azalea Drive, Derwood, MD 20855: ". . . being in Minneapolis, do you know of any implementations of PASCAL for a Control Data Corp. 3500? I am aware of the CYBER implementation, but I have a CDC 3500 available for my use." (* 77/12/01 *)

Herb Rubenstein, 1036 6th St., Golden, CO 80401: "I would like any information on Varian V75 Pascal--even a Pascal to Fortran preprocessor (translator)." (* 77/12/19 *)

Janne Sahady, Systems Programmer, LAMBDA, Div. of Biol. and Med., Brown Univ., Providence, RI 02912: "We have recently implemented a Pascal compiler on a V77-600 Univac minicomputer (formerly Varian Data Machines). This compiler is the sequential version of P. Brinch Hansen's Concurrent Pascal compiler. Our current emphasis is on upgrading the I/O interface and we hope to be writing major system utilities (a mag tape utility to start with) in Pascal in the near future. "Herb Rubenstein, currently working at Autotrol, has referred us to your newsletter and mentioned that you are maintaining a Pascal software library. . . we would definitely be interested in contributing to it as we develop useful routines--most likely in the areas of graphics, signal processing and I/O utilities." (* 77/09/16 *)

Stephen C. Schwarm, duPont Co., 101 Beech St., Wilmington, DE 19898: "Should have Sweden PDP-11 compiler self-compiling soon." (* 77/12/6 *)

Ted Shapin, 5110 E. Elsinore Ave., Orange, CA 92669: "I have access to an IBM 370 and Stanford's version." (* 77/12/15 *)

Thomas E. Shields, Software Resources, 2715 Bissonnet, Suite 212, Houston, TX 77005: "We have UCSD Pascal compiler for B6700 - currently a 1 x 1 (1 cpu, 1 I/O processor); soon to become a 2 x 2)." (* 77/11/04 *)

John Sigle, Computing and Information Sciences, Trinity Univ., 715 Stadium Drive, San Antonio, TX 78284: "We have two Digital Group systems, a Motorola 6800 and a Z-80, and I am interested in developing PASCAL compilers and/or interpreters for them." (* 77/12/7 *)

Jon Singer, 1540 W. Rosemont CE, Chicago, IL 60660: "Do you know of anyone around here who has a micro running PASCAL? I would like to see such a system." (* 77/12/14 *)

Dave Skinner, Communication Mfg. Co., 3300 E. Spring St., P. O. Box 2708, Long Beach, CA 90801: "I just finished reading the article 'Is PASCAL the next BASIC?' in the December issue of BYTE magazine, where they mention the Pascal User's Group. As a former PASCAL user (on the Univ. of Colorado CDC 6400's), I am interested in following the

developments of the language as well as perhaps finding a compiler for one of our machines (NOVA, PDP-11, or any microprocessor)." (* 77/12/06 *)

Eric Small, 680 Beach St., San Francisco, CA 94109: "Am using ESI Pascal in process control type application in broadcasting." (* 77/11/30 *)

Jon A. Solworth, 7 W. 14th St., Apt. 15A, New York, NY 10011: "Please send info on implementations on any minicomputer and addresses if possible (especially Interdata)." (* 77/09/08 *)

Turney C. Steward, 201 Drake St., San Francisco, CA 94112: "I am at present using a Pascal compiler running on a CDC 6600 at Berkeley, Cal., but would be most appreciative to obtain info on versions for microcomputers, especially 8080 or Z80 systems, either resident or cross-compilers." (* 77/12/14 *)

Jim Stewart, 194B Pleasantview Rd., Piscataway, NJ 08854: "I am interested in the implementation of a subset of PASCAL on a Z-80 based micro-computer system." (* 77/11/20 *)

Jyrki Tuomi and Matti Karinen, Room 2113, Computing Center, Tampere University of Technology, Box 527, 3310 Tampere 10, Finland: "When you wrote to us with info about PUG, you said that there are 4 members in Finland already. "Well, now we are doubling that, and more. The coupons are enclosed and here's the money, too. "We have a PDP 11/70 at our disposal and have sent for a couple Pascal implementations. What comes out of this, we shall see. . . ." (* 77/10/7 *)

Steven Vere, Asst. Prof., Dept. of Information Eng., Univ. of Illinois at Chicago Circle, Box 4348, Chicago, IL 60680: "In the December 1977 issue of Byte Magazine Carl Helmers mentioned that a PASCAL compiler exists for the Z80 microprocessor. Do you have any direct information on this compiler, or know where information can be obtained? I would like to know

1. the core requirements
2. cost of obtaining the compiler
3. if it runs on the Z-80 or is a cross-compiler
4. where and how it can be obtained." (* 77/12/12 *)

Wayne Vyrostek, Tektronix, Inc., MS 74-329, P.O. Box 500, Beaverton, OR 97077: "I am a Technical Instructor on Microprocessor Development aids for Tektronix, Inc. I would like to enroll our training department in the Pascal users group and receive back issues that are available. I would also appreciate information you have about training programmers in the use of PASCAL; particularly for Microprocessor software development." (* 77/09/06 *)

Donald Warren, 130 W. 81st St., Apt. 7, New York, NY 10024: "I heard about the group in Creative Computing. I've been programming in Pascal for the past four years, first at the State Univ. of N.Y. at Buffalo, and currently at N.Y. University, and I'm pleased to see its use has spread enough to merit this organization." (* 77/09/12 *)

Hellmut Weber, Leibniz-Rechenzentrum, Der Bayerischen Akademie der Wissenschaften, Barer Strasse 21, D-8000 Munchen 2, West Germany: "I am collecting from colleagues some more user-oriented points of view. (I feel that simple users who want to write production programs haven't found enough attention in the PASCAL community)." (* 77/09/12 *)

Terry Weymouth, 4702 Beau Bien Lane East, Lisle, IL 60532: "I'm interested in any news on micros with PASCAL (or should that be PASCAL with micros?)" (* 77/12/7 *)

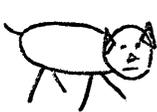
Fulton Wright, Jr., Yavapai College, 1100 East Sheldon Street, Prescott, Arizona 86301: "I'm the educational coordinator for Computer Services at Yavapai College. I've just read an editorial in BYTE magazine about PASCAL. I know almost nothing about it, but the editorial makes it sound like the language of my dreams. The editorial suggested you as a source of further information. What should I and my DEC 10 do next?" (* 77/12/7 *)

Mark Zimmer, #10 2750 Dwight Ave., Berkeley, CA 94704: "PASCAL is supported in U.C. Berkeley (in which I am a student) for the use of teaching data-structures (and now) compilers courses. I learned it in our style course - CS 40. I am interested in implementing PASCAL on the DG ECLIPSE machine. The specification is done (PASCAL has been modified slightly so that ALGOL code from DG can be a subset) and the code is pouring forth for the compiler. Since PASCAL is a "one-pass language," some readers may be interested in my three-pass approach with special emphasis on the reversability of the parse-tree into "source" form." (* 77/09/23 *)

* * *

Dear Editor,

Just so this newsletter isn't quite so serious, can I draw your attention to the evolution of the pug dog that is the heraldic emblem of the Pascal Users Group? The York Herald of Arms, in visiting Tasmania recently, was at pains to emphasize that artists were free to re-interpret heraldic emblems and that this was a medieval norm. He said it was regrettable that in this machine age a sad uniformity had crept in. As you can see from the samples reproduced here, Pascallers are free from this mechanistic taint, and the guardian of rational programming has changed significantly over the months, even at one stage being rather pig-like (possibly an unintended pun).



1976, U.S.A.



1977, Europe



1977, Australia

Arthur

PASCAL IN THE NEWS

(* This new section will list articles which take note of Pascal, sometimes just in passing. Most of the entries here don't really belong in a bibliographical section like "Books and Articles," but they do give some indication of the currency of Pascal. The references have now become so frequent that they merit being set off in a separate section of "News." Several PUG members have mentioned that "Here and There" is visible proof of interest in Pascal. We hope that this section is useful in the same way. The three most important articles listed here are Carl Helmers' editorial in *Byte* and David Barron's article in *Computing Europe*, and the press release (with editorial comment by Andy) from the US Dept. of Defense. PUG member David Mundie is doing heroic work, writing letters to the editor in praise (and defense) of Pascal. Three of his letters are listed here, as are one each by PUG members George Cohn and Stephen Alpert. More PUG members should do the same. In addition, we'd appreciate copies of references made to Pascal in publications you read so that we can make this list more complete. *)

BYTE, 77/09, p. 174, two letters to the editor, one from PUG member George Cohn. Both suggest Pascal as a high-level language for micro-processors.

BYTE, 77/10, "C: A Language for Microprocessors?" J. Gregory Madden. Mentions Pascal as "a reasonable candidate" for a high level, machine-independent language for microprocessors, but goes on to tout Bell Labs' language C as the candidate of choice.

BYTE, 77/11, "Language Development. A Proposal," Glen A. Taylor. Mentions Pascal as a "good structured programming language," but rejects any "large" language "as the best choice for a standard home computing language."

BYTE, 77/11, Two letters to the editor suggesting that Pascal be considered as a standard language for programs and as an excellent high-level language for microcomputers. The writers are PUG members Stephen Alpert and David Mundie.

BYTE, 77/12, "Is Pascal the next B ASIC?" editorial by Carl Helmers. "We at *BYTE* are interested in giving Pascal a boost," best sums up the author's attitude. The editorial demolishes, point-by-point, several arguments frequently made in favor of BASIC, and argues the superiority of Pascal in several areas. Well worth reading, if you are interested in personal computers.

Computer Weekly, 77/10/20, "ICL Pascal users expect boost." Report of a Pascal users' group for exchange of software among ICL users: organizer, David Joslin, Univ. of Sussex, Brighton, England. Pascal users in Britain are dickering with the Numerical Algorithms Group, which produces scientific routines in other languages, to get a scientific library translated into Pascal. See David Barron's Help Wanted ad in another part of "Here and There" for details.

Computer Weekly, 77/10/20, "More support for Pascal." Reports that "the Swedish defence procurement agency, EMV, has specified a Pascal-based language as its standard for real time software development."

Computing, 77/10/20. "Two pleas to Pascal users." Similar to the first article from *Computer Weekly*.

Computerworld, 77/11/14, "'Sounds of Computing' a Record for History," Miles Benson. A tongue-in-cheek description of a new record entitled "Sounds of Computing," the second movement of which is called "Pascal Time-Sharing Terminal."

Computing Europe, 77/10/24, pp. 18-19, "Letting the dinosaur know that it's dead," David Barron. An argument for burying FORTRAN, however great its effect on computing might once have been. Barron argues the need for using compact, conceptually clear languages that make writing correct programs and specifying data structures easy. Pascal is the language of choice.

Creative Computing, Sept/Oct 1977, p. 11. "A Plug for Pascal" letter to the editor from PUG member David Mundie. He counters an argument for creating a structured COBOL/BASIC by showing how clearly a main program can be in Pascal. His example program (5 lines long) is a Pascal version of the one used in the article he criticizes.

Dataline, 77/10/31, "Blazing the trail for Pascal." More reports from the UK about the ICL user's group, the Swedish defense language contract, and David Barron's commission to write a Pascal compiler for the ICL 2900 series.

First Computer Faire Proceedings, pp. 245-247, "Computer Languages: the Key to Processor Power," by Tom Pittman. Discusses the virtues of various high level languages for personal computers. Mentions Sequential Pascal, and says that Pascal is in many ways better than FORTRAN or BASIC.

Kilobaud, October 1977, p. 11. Another letter to the editor by PUG member David Mundie. He says that Pascal is a better language than a structured BASIC discussed in an earlier article, and gives a sample main program in Pascal which duplicates the program suggested by the author of the earlier article.

MACC Computing News (University of Wisconsin Academic Computing Center), 77/11/28. Mentions plans to distribute a Pascal compiler under a license agreement. Reference comes in an article about proprietary software.

SCCS Microcomputer Interface, Aug. 1977, p. 52. An announcement about the existence of PUG and Pascal News.

Stanford Campus Computing Bulletin, Nov. 1977, p. 24. A user wrote in to ask that Stanford acquire a good Pascal compiler. The editor's response was that Stanford is looking into what compiler to acquire.

Twin Cities Technical Hobbyist, (77:9), pp. 01111-10000, "Pascal in Micros," Geoff Wattles. An article describing Pascal, with a discussion of the syntax of the language and Pascal's usefulness for hobbyists.



The U. S. Department of Defense High Order Language Effort (or "IRONMAN" or "DOD-1")

(* In PUGN8, May, 1977, on page 3, we passed along the summary of a press release by the U. S. Department of Defense which was distributed by the British Computer Society on March 3, 1977.

William A. Whitaker, Lt. Col. USAF, Defense Advanced Research Projects Agency (DARPA) described in the full version of that press release a three year effort by the U.S. Defense Department (DoD) to develop specifications for a real-time language called DOD1--a single common military computer programming language for "embedded systems," computer systems on board tanks and ships, on rifles, etc.).

The language would replace FORTRAN, COBOL, JOVIAL, and all the others. A working group at DARPA was formed in January, 1975. A rigorous language definition was sought, but as it turned out, 4 different stages of development/evolution have transpired: first STRAWMAN, a set of relatively complete, although tentative requirements. WOODENMAN and TINMAN followed. With each step, the proposals were widely distributed for comment. Existing languages were evaluated, and, as we reported in PUGN8, only Pascal, Algol-68, and PL/1 survived.

At the IRONMAN (4th step), specifications gave way to a language definition. In July, 1977, IRONMAN was released to vendors for competitive bidding. The report below, sent to us by William Whitaker, tells the results. The four successful contractors will be narrowed to two in February, 1978. What is at stake is \$3 billion spent on defense-related software per year.

It is truly amazing how a giant operation such as the HOLWG (Higher Order Language Working Group) came up with 95% Pascal almost independently in three years working with committees.

There has been sporadic news coverage of these events in the computer trade journals. One, in October 1977 Datamation, reported on how a French software organization's bid got lost in the mail in the original competition. In the December, 1977, SIGPLAN Notices, it is amusing to see the confusion resulting from their just having learned about IRONMAN and still not realizing that it's going to be based on Pascal.

- Andy Mickel *)

The Defense Supply Service Washington has announced the award of four contracts to produce competitive prototypes of a common high order computer programming language for Department of Defense embedded computer systems. These awards came as a result of a request for proposal and offers received from fourteen firms, both U.S. and foreign. The successful contractors were Honeywell (CII-Honeywell Bull), Intermetrics, Softech, and SRI-International.

While different approaches were offered, all four winning contractors proposed to start from the computer language PASCAL as a base. They will provide modifications to construct a resulting language to satisfy military needs as expressed in the "DoD Requirements for High Order Computer Programming Languages (Revised IRONMAN, July 1977)".

The contracts provide for three phases at the discretion of the government. The first phase is to be six-months and will produce a preliminary language design. At the end of the first phase, an evaluation of the products will result in some of the contractors being continued through full formal design, rigorous definition, and prototype implementation. The one contractor whose language is selected by the government will be continued for refinement and initial maintenance. The language will be ready for initial use in 1979.

This language design is the next step in a Department of Defense effort to reduce software costs of embedded computer systems. Earlier actions included issuing DoD Directive 5000.29, "Management of Computer Resources in Major Defense Systems," which, as one of several management actions, required the uses of approved high order languages in future defense systems software. DoD Instruction 5000.31, "Interim List of DoD Approved High Order Programming Languages," stopped proliferation by approving only seven existing languages.

The technical effort in high order languages has, over the last three years, brought increasingly refined sets of requirements, produced an evaluation of existing languages, and has established the technical feasibility of a single language for these applications. The successful design of such a language will be followed by testing and evaluation, compiler and tool generation, and the necessary long-term language control. This program is presently being directed by the DoD High Order Language Working Group, chaired by Lt Col William A. Whitaker, Defense Advanced Research Projects Agency, 1400 Wilson Blvd., Arlington, Va., 22209.

CONFERENCES

German ACM meeting on Pascal, held 77/10/14-15 in Kaiserslautern.

(* We received a postcard from Hans Wipperman, Albrecht Beidl, Manfred Sommer, Helmut Schauer, Lutz Christoph, and Thomas Wagner, all of whom attended the conference. We haven't as yet received a report on the proceedings; therefore we are printing a list of the papers presented so that you can write for more information if you like. The address for inquiries is Hans-Wilm Wipperman, Informatik, F13, Univ. of Kaiserslautern, Pfaffenbergstr. 95, Kaiserslautern D-6750, Germany. The German titles are from the program; the English ones PUGN's attempt at translation. *)

- H. Burkhardt (ETH Zuerich), "Ein interaktives System zur Programmierung in PASCAL." (* "An interactive System for Programming in PASCAL." *)
- H. Balzert (Univ. Kaiserslautern), "PASCAL aus didaktisch-methodischer Sicht." (* "Pascal from the point of view of teaching methods." *)
- Brunnstein (Univ. Hamburg): "Erste Erfahrungen mit dem Einsatz von PASCAL-E im Schulversuch." (* "First experiences on the introduction of PASCAL-E into a school project." *)
- H. J. Hoffman (TH Darmstadt): "Überlegungen zu PASCAL und zur PASCAL-Implementierung." (* "Overview of Pascal and Pascal implementations." *)
- M. Sommer (Siemens Muenchen), "Das PASCAL-BS 2000 Programmiersystem." (* "The Pascal-BS 2000 Programming System." *)
- R. T. Kolsch (Univ. Kiel), "Laufzeitbeschleunigung durch den Einsatz von Mikroprogrammen." (* "Run-time system through a microprogram." *)
- H. D. Petersen (Univ. Stuttgart), "Über die Implementierung von PASCAL auf der TR 440." (* "Implementing Pascal on the TR 440." *)
- H. Grauer (Kernforschungszentrum Karlsruhe), "Hilfsmittel zur Analyse des dynamischen Verhaltens von PASCAL-Programmen." (* Aids to analysis of the dynamic constructs of PASCAL programs. *)
- G. Peresch and G. Winterstein, "Testdatengenerierung fuer PASCAL-Programme." (* "Test data generation for Pascal programs." *)
- Th. Weller (Philips, Eiserfeld): "Concurrent PASCAL als Entwurfs- und Implementierungssprache fuer ein kommerzielles Betriebssystem." (* Concurrent Pascal project and implementation language for a commercial operating system. *)
- Spieß (Univ. Braunschweig): "Die Implementierung von PASCAL fuer die PRIME 300." (* "Implementing Pascal on the PRIME 300." *)
- W. Metzger (Univ. Karlsruhe), "Entwurf eines dialogorientierten Programmiersystems fuer Kleinrechner auf der Basis der Programmiersprache PASCAL." (* A dialogue-oriented programming system for a mini-computer based on the programming language PASCAL. *)
- W. Remmele (Siemens Muenchen), "Ein portables PASCAL-System fuer Mikro-rechner (PPS)." (* "A portable PASCAL system for micro-computers (PP's)." *)

ACM '77, Seattle, held 77/10/27.

- (* report from Richard J. Cichelli, loosely transcribed from a phone conversation: *)
- "Basically, ACM '77 screwed up the schedule so that the computer chess tournament and SIGFISH conflicted with the PUG meeting and confused everyone. Fifteen people came to the Pascal gathering.
- "Some people quoted a "Pascal's I/O is no good" rumor. The confusion was in thinking that textfiles were Pascal's only form of I/O.
- "I emphasized that there should be no attempt to add things to Pascal to make things compatible with COBOL/FORTRAN; Pascal's I/O is fine: textfiles are for people, not for programs."

(* Ken Bowles is planning a summer workshop about Pascal. See the letters section for his letter with details. *)

APPLICATIONS

S. Matwin, M. Missala, "A Simple, Machine Independent Tool for Obtaining Rough Measures of Pascal Programs," SIGPLAN Notices (11:8), August, 1976, pp. 42-45.

"Description of a Pascal program to augment Pascal programs with code to gather execution time information by procedure entry and exit." (* A listing of the programs will appear as a software tool in PUGN 12. *)

James L. Peterson, "On the Formatting of Pascal Programs," SIGPLAN Notices (12:12), December, 1977, pp. 83-86.

"One aspect of programming style which affects the usefulness of programs is their readability. A program is readable if a programmer can pick up the program and read and understand it. Many aspects of style affect readability, including variable names, commenting, modularity, and formatting. It is this last aspect of readability that we discuss here." (* from the abstract *)

P. Roy, "Linear Flowchart Generator for a Structured Language," SIGPLAN Notices (11:11), November, 1976, pp. 58-64.

"This article refers to a paper by Nassi and Shneiderman published in this review. They introduced a type of flowchart specially designed for structured programming. We have defined a similar flowchart language for the Pascal programming language and designed a program which, given a program written in Pascal, generates the corresponding flowchart. The article presents a description of the output produced by this flowchart generator." (* from the abstract *)

Joachim W. Schmidt, "Some High Level Language Constructs for Data of Type Relation," ACM Transactions on Database Systems (2:3), September, 1977, pp. 247-261.

"For the extension of high level languages by data types of mode relation, three language constructs are proposed and discussed: a repetition statement controlled by relations, predicates as a generalization of Boolean expressions, and a constructor for relations using predicates. The language constructs are developed step by step starting with a set of elementary operations on relations. They are designed to fit into Pascal without introducing too many additional concepts." (* from the abstract *)

D. A. Thomas, B. Phaguvek, R. J. Buhr, "Validation Algorithms for Pointer Values in DBTG (Data Base Task Group) Data Bases," ACM Transactions on Database Systems (2:4), December, 1977, pp. 352-369.

"This paper develops algorithms for verifying pointer values in DBTG (Data Base Task Group) type databases. To validate pointer implemented access paths and set structures, two algorithms are developed. The first procedure exploits the 'typed pointer' concept employed in modern programming languages to diagnose abnormalities in directories and set instances. The second algorithm completes pointer validation by examining set instances to ensure that each DBTG set has a unique owner. Sequential processing is used by both algorithms, allowing a straightforward implementation which is efficient in both time and space. As presented, the algorithms are independent of implementation schema and physical structure." (* from the abstract *)

IMPLEMENTATIONS

D. Bates, R. Cailliau, "Experience with Pascal Compilers on Mini-Computers," SIGPLAN Notices (12:11), November, 1977, pp. 10-22.

"This paper relates the history of an implementation of the language Pascal on a mini-computer. The unnecessary difficulties encountered on the way led the authors to reflect on the distribution of "portable" compilers in general and suggest some guidelines for the future. Their experiences described within show that it should be possible to implement a P4 Pascal System on any 16-bit mini-computer in less than two man months, given an implementor already familiar with the target machine." (* From the abstract *)

LANGUAGES

(* This section is sub-divided for this issue. The first section is a set of miscellaneous articles in alphabetical order by author. The second is a semi-complete list of articles on the subject of dynamic arrays in Pascal. The third is a short bibliography on Concurrent Pascal, supplied by Rich Stevens. *)

R. Conradi, "Further Critical Comments on Pascal, Particularly as a Systems Programming Language," SIGPLAN Notices (11:11), November, 1976, pp. 8-25.
There has recently been some controversy between Habermann and Lecarme and Desjardins on Pascal in "Acta Informatica." This paper contains some more comments on Pascal from a systems programmer's point of view. Some undefined points are first treated. Then Pascal's datatypes are critically reviewed. A few remarks on common Pascal constructs are also given. Since the author has experience with the programming language MARY, some comparisons between Pascal and MARY will be made." (* from the author's abstract *)

R. Edwards, "Is Pascal a Logical Subset of ALGOL 68 or Not?" SIGPLAN Notices (12:6), June, 1977, pp. 184-91.

"It is often believed that Pascal is
ALGOL 68 in miniature
well structured

it will be argued that both beliefs are badly founded." (* from the abstract *)

J. Holden and I. C. Wand, "Experience with the Programming Language MODULA," a paper presented to the 1977 IFAC/IFIP Real Time Programming Workshop held at Eindhoven, Netherlands, 77/06/20-22.

"This paper describes a compiler for MODULA, written in the programming language BCPL, which runs on a PDP-11/40 computer under the RSX-11D operating system. The code produced by the compiler is run on PDP-11s under a very small executive (less than 150 words). The quality of the code produced compares well with that of compilers for other high-level languages.

The use of the language is illustrated by the construction of a real-time scheduler similar to that written by Brinch Hansen in CONCURRENT PASCAL. A brief discussion is given of experience gained in the use of the language and comments made about the inclusion and exclusion of certain language features." (* From the abstract *)

W. H. Kaubisch, R. H. Perrott, and C. A. R. Hoare, "Quasiparallel Programming," Software: Practice and Experience (6), 1976, 341-356.

"This paper describes SIMONE, and extension of PASCAL, which provides the quasiparallel programming facility of SIMULA 67, but without classes or references. The language is intended to be suitable for the design, testing and simulation of operating system algorithms. It is illustrated by simple examples, suitable as project material in a course on operating systems." (* from the abstract *)

E. N. Kittlitz, "Block Statements and Synonyms for Pascal," SIGPLAN Notices (11:10), October, 1976, pp. 32-35.

"PYXIS is a language which is the result of (still continuing) modifications to Wirth's PASCAL 1 system. Many of the language concepts are identical to, or slightly evolved from PASCAL 1, others are incompatible with Pascal and its apparent design philosophy. Two new features have been implemented in PYXIS: block statements and synonyms." (* From the abstract *)

O. Lecarme, "Is ALGOL 68 a Logical Subset of Pascal or Not?" SIGPLAN Notices (12:12), December, 1977, pp. 33-35.

"A paper by Roy Edwards uses a comparison of ALGOL 68 and Pascal to make some disputable assertions. The purpose of the present note is simply to correct the most serious errors. It follows exactly the structure of Edwards' paper." (* from the abstract *)

R. D. Tennent, "A Denotational Definition of the Programming Language PASCAL," Technical Report 77-47, Department of Computing and Information Science, Queen's University, Kingston, Ontario, Canada, July 1977.

"This report presents a formal definition of the semantics of the programming language PASCAL, including static aspects such as scope and type checking, using the concepts and notation of denotational semantics. It is suggested that the definition can be used as a standard from which to derive complete informal descriptions, valid proof rules, and correct implementations." (* From the abstract *)

M. Yasumura, "Evolution of Loop Statements," SIGPLAN Notices (12:9), September, 1977, pp. 124-129.

"This paper is motivated by two papers. One is written by Ledgard and Marcotty and the other by Ishihata and Hikita. The former paper is a good summary of control structures but its conclusions are seriously questioned. The latter paper is the report of a new Pascal computer in which Zahn's "event" construct is implemented. That construct is, however, shown to be unsuitable to Pascal." (* From the abstract *)

(* A chronological exchange on dynamic arrays in Pascal *)

B. J. MacLennan, "A Note on Dynamic Arrays in Pascal," SIGPLAN Notices (10:9), September, 1975, pp. 39-40.

"Pascal is frequently criticized for its lack of any variety of dynamic array facility. This lack is particularly unfortunate for systems programs which must manipulate activation records and segments whose sizes are not known at compile time." (* From the abstract. *)

N. Wirth, "Comment on A Note on Dynamic Arrays in Pascal," SIGPLAN Notices (11:1), January, 1976, pp. 37-38.

"A reply to B. J. MacLennan and a suggested alternative." (* From the abstract. *)

J. Steensgaard-Madsen, "More on Dynamic Arrays in Pascal," SIGPLAN Notices (11:5), May, 1976, pp. 63-64.

"A further proposal in reply to Wirth's article." (* From the abstract. *)

C. Jacobi, "Dynamic Array Parameters," Pascal User's Group Newsletter (5), September, 1976, pp. 23-25.

"A proposed description of dynamic array parameters is given in the form of a set of amendments to the book, Pascal User Manual and Report by Jensen and Wirth with syntax diagrams and examples. The extension was implemented successfully in the Pascal-6000 compiler." (* From the abstract. *)

S. Pokrovsky, "Formal Types and Their Application to Dynamic Arrays in Pascal," SIGPLAN Notices (11:10), October 1976, pp. 36-42.

"The formal type concept is presented as a means to uniformly introduce in the Pascal language the dynamic array facility (which may be done as a pure extension) and formal procedure specifications (which would require some changes in the standard language)." (* From the abstract. *)

Edward N. Kittlitz, "Another Proposal for Variable Size Arrays in Pascal," SIGPLAN Notices (12:1), January, 1977, pp. 82-86.

"The syntax, semantics, and some implementation details for a flexible array bound capability in Pascal are discussed. The constructs described are currently implemented as part of the PYXIS system at the University of Calgary. PYXIS is the result of more than two years of modifying [the old Pascal-6000 compiler written by Urs Ammann et. al.]." (* From the introduction. *)

M. Condict, "The Pascal Dynamic Array Controversy and a Method for Enforcing Global Assertions," SIGPLAN Notices (12:11), November, 1977, pp. 23-27.

"In a previous article, Wirth commented that allowing expressions (rather than just constants) as subrange bounds would produce dynamic array capability without significantly complicating the language. . . . This discussion leads directly into a method for obtaining automatic enforcement of assertions about variables throughout their lifetime." (* From the abstract. *)

Brinch Hansen, Per., The Programming Language Concurrent Pascal. IEEE Trans. on Software Engineering 1, 2 (June 1975), 199-207.

Introduces Concurrent Pascal - an abstract language for concurrent programming. It extends the sequential programming language Pascal with modules called processes, monitors, and classes. The language is illustrated by a hierarchical design of a simple spooling system. The main contribution of Concurrent Pascal is to extend the monitor concept with an explicit hierarchy of access rights to shared data structures that can be stated in the program text and checked by a compiler.

Brinch Hansen, Per., The Solo Operating System. Software - Practice & Experience 6, 2 (April-June 1976), 141-205.

Describes the single-user operating system Solo written in Concurrent Pascal. It supports the development of sequential and concurrent Pascal programs for the PDP 11/45 computer. Input/output are handled by concurrent processes. Pascal programs can call one another recursively and pass arbitrary parameters among themselves. This makes it possible to use Pascal as a job control language. Solo is the first major example of a hierarchical concurrent program implemented in terms of abstract data types (classes, monitors, and processes). The paper contains the complete text of the concurrent program. It is a sequence of nearly independent components of less than one page of text each.

Brinch Hansen, Per., Experience with Modular Concurrent Programming, IEEE Trans. on Software Engineering 3, 2 (March 1977), 156-159.

Summarizes the first 2 years of experience with Concurrent Pascal in the design of three model operating systems. A Concurrent Pascal program consists of modules (processes, monitors, and classes). The compiler checks that the data structures of each module are accessed only by the operations defined in the module. The creative aspect of program construction is the initial selection of modules and the connection of them into hierarchical structures. By comparison the detailed implementation of each module is straightforward. The most important result is that it is possible to build a concurrent program of one thousand lines out of one-page modules that can be comprehended at a glance.

Hartmann, A.C., A Concurrent Pascal Compiler for Minicomputers. Lecture Notes in Computer Science 50, Springer-Verlag, New York, NY, 1977.

Describes a seven-pass compiler for Concurrent Pascal. The compiler, written in sequential Pascal, generates virtual code that can be interpreted on any 16-bit minicomputer. The function of each pass is described and the intermediate languages are defined by syntax graphs. Of particular interest is the checking of access rights to data structures within classes, monitors, and processes. This is done exclusively during compilation and is not supported by hardware protection mechanisms. The compiler has been running on a PDP 11/45 computer since January 1975.

Brinch Hansen, P., The Architecture of Concurrent Programs. Prentice-Hall, Englewood Cliffs, New Jersey, July 1977.

Presents a method for developing reliable concurrent programs using Concurrent Pascal. The use of this language is illustrated by three model operating systems for minicomputers; a single-user operating system, a job-stream system, and a real-time scheduler. All of them have been running

successfully on a PDP 11/45 computer. The book includes the complete text of these programs and explains how they are structured, programmed, tested, and described. It also includes the Concurrent Pascal Report and a description of the Concurrent Pascal Machine. The book suggests promising areas of further research in structured concurrent programming.

Brinch Hansen, P., Network: A Multiprocessor Program. IEEE Computer Software & Applications Conference, Chicago, Illinois, Nov. 1977.

Explores the problems of implementing arbitrary forms of process communication on a multiprocessor network. It develops a Concurrent Pascal program that enables distributed processes to communicate on virtual channels. The channels cannot deadlock and will deliver all messages within a finite time. The operation, structure, text, and performance of this program are described. It was written, tested and described in 2 weeks and worked immediately. The program has been running on two PDP 11/45 computers connected by bus links.

TEXTBOOKS

Tony Addyman and I. R. Wilson, A Practical Introduction to Pascal, MacMillan, March-April 1978, 140 pages.
A short and concise introduction to Pascal.

S. Alagic and M. A. Aebib, The Design of Well-structured and Correct Programs, New York: Springer-Verlag, to appear in 1978, 260 pp., \$12.80. An undergraduate text. "Using the Pascal language, both the techniques of top-down program design and verification of program correctness are presented. Many examples of program and proof development as well as an explanation of control and data structures are provided. As a Pascal programming text, it gives not only advanced algorithms, which operate on advanced data structures, but also the full axiomatic definition of Pascal.
"Although an introductory course in programming is presupposed, no particular mathematical background is necessary. An extensive, carefully chosen sample of algorithms, including some examples from business data processing, is presented. Supplementing this collection is an extensive set of exercises." (* From the publisher's ad *)

Kenneth L. Bowles, Microcomputer Problems Solving Using Pascal (* Please note the correct title; it's the first time we've had it right *), New York: Springer-Verlag, 1977, 563 pp., \$9.80.
"This text introduces problem solving and structured programming using the PASCAL (sic) language, extended with built-in functions for graphics. Designed for a one-quarter/semester curriculum at the sophomore/junior level, this book serves a dual purpose: to teach students an organized approach to solving problems, and to introduce them to the computer and its applications, which may be of use later in their chosen professions." (* From the publisher's ad. See also R. Cichelli's review in this section. *)

Peter Grogono, Programming in Pascal, Addison-Wesley, February, 1978, 350 pp., \$10.50.
"An introductory book; assumes no prior knowledge of programming languages or computing techniques." (* Publisher's ad *)

(* We hear rumors of more texts to come from the UK: one by Jim Welsh and John Elder (Dept. of Computer Science, Queen's Univ., Belfast, N. Ireland BT7 1NN), publisher unknown; and one by David Watt and Bill Findlay (Computing Science Dept., Univ. of Glasgow, Glasgow, Scotland G12 8QQ), Pittman Publishers. *)

Book Review - (Microcomputer) Problem Solving Using PASCAL

By Kenneth L. Bowles
 Springer-Verlag, New York, 1977
 ISBN 0-387-90286-4
 563 pp. \$9.80

Microcomputer ... PASCAL, the title sounds like a wish. But it's true. Professor Bowles of the University of California at San Diego (UCSD) has full PASCAL (with extensions for graphics, character string manipulation and direct access files) running on interactive single-user microcomputer systems - Digital Equipment LSI-11's, Zilog Z-80's, and 8080 based machines. He expects to soon have Motorola 6800 and MOS Technology 6502 PASCAL systems as well. Almost all of the software for these systems is written in machine independent PASCAL.

The text, (Microcomputer) Problem Solving Using PASCAL, is an integral part of a revolutionary environment for computing education. As Bowles says, "PASCAL is clearly the best language now in widespread use for teaching ... structured programming at the introductory level". By using PASCAL Bowles is able to introduce algorithm development and problem solving as components of top-down, stepwise design. Procedures are introduced right from the start. Flow of control is presented in terms of modern programming principles (sequence, selection and iteration). Recursion is presented as an obvious extension of the procedure mechanism. Data structures are explained fully and clearly.

Because the text uses graphics and text processing programming examples, unnecessary numericalization of computer science principles is avoided. At UCSD students from the arts, humanities and business disciplines are able to do just as well writing programs for non-numeric applications as are more mathematically sophisticated students. (Bowles' UCSD course is phenomenally successful. More than 650 students from all disciplines registered for it in the Fall of 1977 before registration had to be closed. They use more than 20 single-user micro-systems - each costs about \$5,500 and consists of micro computer, diskette storage, keyboard and graphics display.)

In addition to the compiler/interpreter/editor software, the UCSD system includes a complete computer aided and managed instruction system. The CAI lessons parallel the text and permit easy management of very large introductory classes. The system is simple and complete in and of itself and could also be used effectively by high schools and community colleges to provide low-cost interactive student computing.

Kenneth Bowles has revolutionized the teaching of introductory computing at UCSD. The publication of this book and release of the UCSD PASCAL system will permit other schools to follow his lead.

Reviewed by: Richard J. Cichelli
 Research Manager, Computer Applications
 American Newspaper Publishers Association/
 Research Institute
 Easton, PA and
 Department of Mathematics
 Lehigh University
 Bethlehem, PA

NEWS FROM UNIVERSITY OF COLORADO -- PASCAL DISTRIBUTION CENTER

REPORT REQUESTS

The following is a list of reports currently available:

PASCAL-S: A Subset and its Implementation	\$6.50
On Code Generation in a PASCAL Compiler	\$4.00
The PASCAL <P> Compiler: Implementation Notes	\$5.50
An Axiomatic Definition of the Programming Language Pascal	\$3.00
Concurrent Pascal Implementation Notes	\$3.00
Sequential Pascal Report	\$5.00

*For orders from North America, there is a \$2.50 postage and handling charge. Overseas orders will be billed for appropriate postage.

PASCAL SYSTEM REQUESTS

We are receiving a lot of out-of-date order forms. The prices and options quoted on any form prior to September 1977 are no longer valid. Most important, please note that tapes will no longer be accepted from buyers. Please phone for details or request an up-to-date distribution statement.

Address requests to: PASCAL Distribution
 University of Colorado
 Computing Center
 3645 Marine Street
 Boulder, CO 80309
 U.S.A.
 (303) 492-8131

ERRATA TO PASCAL USER MANUAL AND REPORT second edition

(* Note: These errata were sent to us by Niklaus Wirth in December. The "whole pages" referred to in a couple of places were not sent. Niklaus stated that so far 25,000 copies of the book have been sold! *)

Ø. Keys

p = page
 l = line (l1..l2 means l1 until l2)
 c = code : r = replace
 i = insert (after the line mentioned)
 d = delete

1. Errors (to be corrected mandatorily)

p 1 c

6 5..6 r both lines by
"construct. Enclosure of a sequence of constructs"
"by the meta - brackets { and } implies their repetition"

21 12 r whole line by
"Assignment is possible to variables of any type, except"

35 15 r whole line by
"constant value in the subrange, where the lower bound must not be"
"less than the" by
"greater than the"

50 8 d whole line
50 9 r "implementations" by "Implementations"
50 -4 r "first operand is a scalar type," by
"second operand is of a set"

50 -3..-1 r all the lines by
" type, the first of its associated base type; the
" result is true when the first is an element of the
" second, otherwise false."

59 -11..-10 r both lines by
"Note: The standard procedures reset (rewrite) must not be
"applied to the file input (output)."

69 8 r "a[i]" by "a[1]"
86 17 r "Then," by "Then:"
86 18 d whole line
89 12 r "eof(f)" by "eof(x)"
90 -8 r "textfiles." by "files."
97 7 r "58" by "59"

125 25..26 r both lines by
"read, readln, write, writeln are discussed in chapter 12."
" by
" They must not be changed"
" during execution."
105 -4 r "dispose(p) indicates that storage occupied by the
" variable p| is no longer needed."
105 -1 r " by ". The
105 -1 i " tag field values must be identical to those
" used when allocating the variable."

108 2 r "Operations" by "Operators"
110 12..13 d both lines
116 1 d "PASCAL"
118 r whole page by the corrected one to be found in the enclosure
126..128 r all three pages by the corrected ones to be found in the enclosure
136 -10 r "three" by "four"
136 -9 r "and procedure or" by
" procedure and "
140 15..19 r all the lines by
"packed; this has in general no effect on the meaning of a"
"program (for a restriction see 9.1.2.); but it is a hint to the"
"compiler that storage should be economized even at the price of"
"some loss in efficiency of access, and even if this may expand"
"the code necessary for expressing access to components of the"
"structure."
142 -4 r "test for equality." by
"assignment and the"
142 -4 i "test for equality."
145 -10 r "operands, i.e." by
"operators and"
145 -9 r whole line by
"operands, i.e. variables, constants, and functions."
146 12 r "<adding operator>" by "<sign>"

149 -15 r " hue" by "hug1"
154 -19 r "100" by "63"
154 -15 r "a[i,k]*b[k,j]" by "A[i,k]*B[k,j]"
154 -14 r "c[i,j]" by "C[i,j]"
158 20..21 r both lines by
"Concerning the procedures read, write, readln, writeln, and page"
"see chapter 12."
158 -13 r "procedure" by "procedures"
158 -8 r " by ". The tag"
158 -8 i " field values must be listed contiguously and in the
" order of their declaration and must not be changed"
" during execution."
"dispose(p) indicates that storage occupied by the variable p|"
" is no longer needed. If the second form of new was"
" used to allocate the variable then
"dispose(p,t1,...,tn) with identical tag field values must be"
" used to indicate that storage occupied by this"
" variant is no longer needed."
" + : no line feed (overprinting)"
165 -13 i exchange the two lines
166 -17..-16 r whole page by the corrected one to be found in the enclosure
167 r

2. Printing Errors (to be corrected optionally)

p 1 c

39 6 r "multidimensional" by "multidimensional"
60 -7 r "printeo" by "printed"
61 -1 r "parameter" by "parameter"
64 13 r "ne" by "one"
71 3 r "parameter" by "parameter"
71 -10 r "f" by "of"
91 13 r " and are subsituted" by "and are substituted"
96 -11 r "nf" by "of"
98 17 r "f" by "of"
99 -5 r "ut" by "out"
105 -14 r "procedure" by "procedures"
105 -13 r " by " by "
163 14 r "if" by "If"
163 -10 r "if" by "If"
163 -7 r "if" by "If"

3. Further possible Corrections

p 1 c

32 -16 r "p " by "p ;"
36 -10 r "The general" by "Its"
36 -7 r whole line by
"scalar or subrange type (where types integer and real are"
36 -6 r "index" by "not allowable index"
42 20 r " field identifier is the smallest" by
" field identifier is the innermost"
59 -6 r " by :"
66 2 r " by :"
67 -3 r " ... " by "..."
69 6 r " by :"
76 1 r "(6)" by "[6]"
89 -11 r "schemas" by "schemes"
105 18 r " by :"
163 -6 r " by ", preceded by an appropriate number of"
163 -6 i " blanks as specified by m."

INDEX

When a reference in this index is not a section name (e.g. Appendix A), then the reference may be of the following forms:

x1 x1.x2 x1.x2.x3

x1 is always the chapter number. x2 may be a capital letter in which case it may be followed by x3, a number, and refers to a chapter section. When x2 is a small letter, the reference is a figure; when x2 is a number, the reference is a program.

alpha (PASCAL 6000-3.4) 13.D.1
 array types 6
 assignment statement 4.A
 binary tree 11.A
 block 0
 BNF definitions Appendix D
 Boolean 2.A
 case statement 4.D.2
 char 2.D
 character sets 13.B.3
 comment 1
 compiler error messages 14.C.1, Appendix E
 compiler options (PASCAL 6000-3.4) 13.B
 compound statement 4.B
 conditional statements 4.D
 constant declaration part 3.C
 control statements (PASCAL 6000-3.4) 14.A
 control variable 4.C.3
 data types 2
 declaration part 3
 empty statement 4.B
 equivalence 2.A
 expression 4.A
 field list 7
 figures
 after (list insertion) 10.c
 alternative representation of standard symbols 13.b
 ASCII character set (with CDC's ordering) 13.a
 before (list insertion) 10.b
 binary tree structure 11.b
 block structure 0.b
 CDC scientific character set (with 64 elements) 13.a
 expressions 11.a
 identifier 1.a
 linked list 10.a
 syntax diagram of program structure 0.a
 two sample people 7.a
 unsigned number 1.b
 file types 9
 external files (PASCAL 6000-3.4) 13.B.1
 representation in PASCAL 6000-3.4 13.B.2
 segmented files (PASCAL 6000-3.4) 13.A.1

textfiles 9.A
 for statement 4.C.3
 forward reference 11.C
 functions 11.B
 declaration part 3.F
 designator 11.B
 heading 11.B
 predefined (PASCAL 6000-3.4) 13.D.2
 standard, table of Appendix A
 global variables 11.A
 goto statement 4.E
 identifiers, table of standard Appendix C
 if statement 4.D.1
 implication 2.A
 input 9.B
 integer 2.B
 I/O 12
 labels
 case 7.A
 declaration part 3.B
 goto 3.B, 4.E
 lists (linked) 10
 local variables 11.A
 name precedence 11.A
 notation 1
 numbers 1
 operator precedence 4.A
 operators, summary of Appendix B
 output 9.B
 packed structures 6
 parameters 11.A
 PASCAL 6000-3.4 13, 14
 pointer types 10
 procedures 11.A
 declaration part 3.F
 external procedures (PASCAL 6000-3.4) 13.A.2
 heading 11.A
 predefined (PASCAL 6000-3.4) 13.D.2
 procedure statement 11.A
 standard, table of Appendix A
 program heading 3.A
 (PASCAL 6000-3.4) 13.B.1
 programs and program parts
 beginend 4.1
 bisection 11.6
 complex 7.1
 convert 3.1
 cosine 4.5
 egalfa 13.1
 egfor 4.4
 egrepeat 4.3
 egwhile 4.2
 examples of goto 4.E
 exponentiation 4.8
 expon2 11.8
 forward reference 11.C
 frequency count 9.1

graph1 4.9
 graph2 6.2
 inflation 0.1
 insert 9.2
 matrixmul 6.3
 merge two files 9
 minmax 6.1
 minmax2 11.1
 minmax3 11.2
 parameters 11.3
 pointers, construction via 10
 postfix 11.4
 primes 8.2
 recursivegcd 11.9
 roman 4.7
 setop 8.1
 sideeffect 11.7
 sum file of real numbers 9
 summing 4.6
 tree traversal 11.5
 read, the standard procedure 12.A
 real 2.C
 record types 7
 relational operator 2.A, 4.A
 repeat statement 4.C.2
 repetitive statements 4.C
 reserved words--see word-delimiters
 restrictions (PASCAL 6000-3.4) 13.C
 run-time error messages 14.C.2
 scalar types 5.A
 schemata
 read a text 9.A
 read a text from "input" 9.A
 read and write a segmented file 13.A.1
 reading a segmented file 13.A.1
 reading arbitrary number of numerical
 items from a textfile 12.A
 write a segmented file 13.A.1
 write a text 9.A
 write a text onto "output" 9.A
 write a text x to y 9.A
 scope 0
 separators 1
 set operators 8
 set types 8
 side effect 11.B
 standard identifiers Appendix C
 string 1, 6
 subrange types 5.B
 syntax diagrams Appendix D
 tables
 block structure 0
 default value for field width 13.B.4
 operations on textfiles 9.A
 printer control characters 9.B, 13.B.4
 special symbols 1
 truth values 2.A

REVIEW OF PASCAL NEWSLETTERS 5, 6, 7, AND 8

parameter group	10.
pointer type	6.3
pointer variable	7.3
procedure and function declaration part	10.
procedure declaration	10.
procedure heading	10.
procedure identifier	9.1.2
procedure or function declaration	10.
procedure statement	9.1.2
program	13.
program heading	13.
program parameters	13.
record section	6.2.2
record type	6.2.2
record variable	7.2.2
record variable list	9.2.4
referenced variable	7.3
relational operator	8.1.4
repeat statement	9.2.3.2
repetitive statement	9.2.3
result type	11.
scalar type	6.1.1
scale factor	4.
set	8.
set type	6.2.3
sign	4.
simple expression	8.
simple statement	9.1
simple type	6.1
special symbol	3.
statement	9.
statement part	10.
string	4.
structured statement	9.2
structured type	6.2
subrange type	6.1.3
tag field	6.2.2
term	8.
type	6.
type definition	6.
type definition part	10.
type identifier	6.1
unlabelled statement	9.
unpacked structured type	6.2
unsigned constant	8.
unsigned integer	4.
unsigned number	4.
unsigned real	4.
variable	7.
variable declaration	7.
variable declaration part	10.
variable identifier	7.1
variant	6.2.2
variant part	6.2.2
while statement	9.2.3.1
with statement	9.2.4

Because issues 5, 6, 7, and 8 are now out of print, we ought to lay them to rest properly. This will end a lot of curiosity among new PUG members regarding their contents.

Issues 5-8 were the first to be produced under PUG auspices - see explanation on page 11 of Pascal News 9/10, September, 1977. I suggest you contact PUG members near you and photocopy any issues or parts of issues you really want. - Andy Mickel

Pascal Newsletter #5, September, 1976.

Editor's Contribution: established user group and newsletter policies; recounted the history of the formation of Pascal User's Group over the previous year; described Pascal activities at the University of Minnesota; suggested that all was not well with Pascal because implementations proliferated different features, implementors and critics disregarded Pascal's language design goals, and finally people not having realized the importance of simply making the use of Pascal a respectable activity; acknowledgments to all that helped PUG make a start.

Here and There: 2 conference announcements - a Pascal get-together at ACM '76 in Houston and a preliminary notice of the Pascal Symposium in Southampton in March. a summary of existing or planned textbooks on Pascal; news from Pascalers; errata to the second edition of Pascal User Manual and Report.

Articles:

"Designing Data Structures by Step-wise Refinement"

- Richard J. Cichelli

[Dijkstra and Wirth have defined the principles of systematic programming. They illustrated these principles by designing programs whose control structures reflected hierarchical abstractions of their logic flow. In this paper, systematic programming principles are applied to the design of a program's data structures.]

"In Defense of Formatted Input"

- John Eisenberg

[Formatted input can be useful in many cases, and almost necessary in others. Three examples of "typical" basic computer science problems are presented which would be inconvenient or next to impossible without the use of formatted input.]

"Overlays: A Proposal"

- James F. Miner

[As the availability of Pascal for serious productions grows wider, it will become evident that many implementations will need to cater to features commonly employed in production work which are not currently found in implementations of Pascal. The need to reduce the amount of storage required for a program's object code is such a feature and a proposal for overlays is given here as a remedy.]

"Minor Problems in Pascal"

- Timothy M. Bonham

[A number of syntactic details in Pascal are criticized - not to prove that Pascal is a bad language, but on the contrary to perfect a language which is easily one of the best around, because of its logical clarity, austerity, and the readability of source programs.]

"Dynamic Array Parameters"

- Chris Jacobi

[A description is given of a proposed extension to Pascal-6000 to implement dynamic array parameters. This solves a serious problem in the construction of subprogram libraries written in Pascal. A set of amendments to the User Manual and Report are given, as well as syntax diagrams and an example program.]

Open Forum:

- 75/11/05 John Eisenberg to Andy Mickel: [Pascal-S, University of Illinois PDP-11 Pascal, the User's Group]
- 75/11/22 Urs Ammann to Andy Mickel: [Pascal-6000 extensions (dynamic array parameters value part, constructors); Pascal Day in Switzerland]
- 75/11/25 Andy Mickel to Niklaus Wirth: [questions about the language Pascal, changes to the language, and to features in Pascal-6000]
- 75/12/10 Niklaus Wirth to Andy Mickel: [distinguish clearly between language and implementation; reply to questions about changes; literature about Pascal]
- 75/12/29 Andy Mickel to Niklaus Wirth: [looking at the nature of change - several versions of the Pascal Report, details on Pascal-6000 changes, conventionalized extensions; Pascal Newsletter / Pascal User's Group]
- 76/01/12 Ed Fourt to Andy Mickel: [Lawrence Berkeley Labs and Pascal-6000]
- 76/01/14 Hellmut Golde to Andy Mickel: [Pascal-6000 at the University of Washington Pascal-Fortran subprogram linkage]
- 76/03/08 Wilhelm Burger to Andy Mickel: [Pascal-6000 at the University of Texas, and implementation changes]
- 76/03/09 Richard Cichelli to Andy Mickel: [Soma cube paper, Pascal at Lehigh University DEC PDP 11 Pascal and UNIX]
- 76/03/15 Andy Mickel to George Richmond: [Where is Pascal Newsletter #4?, our plans for the User's Group / Newsletter, plans for the transition]
- 76/04/05 Susan Stallard to Andy Mickel: [IBM 370 Pascal at the University of Southern California]
- 76/05/04 Philip Enslow to Andy Mickel: [Pascal-6000 at Georgia Tech, Brinch Hansen Pascal on the B5700]
- 76/05/23 Harry M. Murphy to Andy Mickel: [Pascal-6000 at Air Force Weapons Laboratory variable dimension array problems]
- 76/05/26 David Elliot Shaw to Andy Mickel: [PDP-11 Pascal employment at Structured Systems Corporation]
- 76/05/27 Charles L. Lawson to Andy Mickel: [Pascal and Numerical Software, adjustable array dimension problems. Pascal on the 1108 at Jet Propulsion Labs]
- 76/06/18 George Richmond to Andy Mickel: [Pascal Newsletter #4, updates on distribution information on Pascal-6000 and Pascal-P]
- 76/06/18 Steve Bellovin to Andy Mickel: [Pascal on the 370 at the University of North Carolina - Chapel Hill, problems with Pascal-P2]
- 76/07/01 James Kendall to Andy Mickel: [Pascal on several machines at the Texas State MHRM Department]
- 76/07/22 Steven Soule to Andy Mickel: [Pascal at the University of Calgary, Pascal's inability to subvert Fortran]
- 76/07/23 Mike Hagerty to Andy Mickel: [Pascal-6000 modifications, Pascal Standards committee, solicit ACM sponsorship, formatted reads]
- 76/07/23 George Richmond to Niklaus Wirth: [Explanation of current situation regarding distribution of compilers, Pascal Newsletter and transfer of duties to other persons]

Implementation Notes: Checklist, Pascal P3 and P4, Pascal Trunk, PascalJ, Pascal-S Machine Dependent Implementations: B6700, CDC-6000, DEC PDP-11, DECsystem 10, Honeywell 6000, IBM 360/370, Univac 1100.

Pascal Newsletter #6, November, 1976.

Editor's Contribution: Standards - course to take very confusing; Pascal User's Group mechanics and finances; feedback from PUG#5; establishment of a PUG outpost in the United Kingdom.

Here and There: News from Pascalers; empirical study of Pascal programs by John Banning at Stanford; agenda for the international Pascal Symposium at the University of Southampton; update on textbooks; update to errata to the Second Edition of Pascal User Manual and Report; listing of contents of Pascal Newsletters 1, 2, 3, and 4; PUG roster of 516 members.

Articles:

- "Indexed Files"
- Svend Knudsen
[In addition to the possibility of dividing sequential files into segments (creating a "segmented file"), it is also possible to construct, read, and modify indexed files. This feature also covers the need for rapid location and modification of segments.]
- "The Need for Hierarchy and Structure in Language Management"
- G. Michael Schneider
[I find it quite ironic that so much concern is being paid to problems of structure and organization of statements within the Pascal language but so little to the structure and organization of the management of the language itself. By this I mean that there is currently lacking a formal administrative hierarchy for the handling of questions relating to language standards, specifications, and extensions]
- "Pascal Potpourri"
- Richard J. Cichelli
[A set of (perhaps ill-formed) topics for the Pascal User is presented for debate: the problem of direct access files, standards and the language Pascal, software tools for the Pascal user]
- "The Case for Extending Pascal's I/O"
- Michael Patrick Hagerty
[With the introduction and subsequent increase in the popularity of Pascal, a number of papers concerning the language, its features and deficiencies, have appeared in various journals and newsletters. Champions of the language have extolled the virtues of its structure and unambiguous grammar using both example and theory as justification of its usefulness. Pascal critics on the other hand, have questioned the claim of the proponents that Pascal will replace FORTRAN, pointing to the inadequacies of the language in several areas. Wirth (1974) defends the absence of certain "favorite features" as necessary to avoid inefficient programming solutions or reliance upon features which are contrary to the aim of clarity and reliability. When the features being debated refer to the flexible input of large amounts of data, the critics hold the stronger hand, and with much justification.]
- "General Thoughts on Pascal Arising out of Correspondence Between Southampton and Tasmania"
- Arthur Sale
[a set of topics of potential interest to the Pascal community: Mixed languages, Portability, Inclusions of Source Text, Files, Standards]

Open Forum:

- 76/07/28 Rich Cichelli to Andy Mickel: [What happened to Pascal-6000 Release 2? Pascal CAI system]
- 76/07/30 Brian Rowswell to Andy Mickel: [Pascal at the University of Sydney Computing Centre]
- 76/08/17 Willett Kempton to Andy Mickel: [Pascal for applications in anthropology, the case for formatted reads.]
- 76/08/31 Henry Ledgard to Andy Mickel: [Pascal Prettyprinter at the University of Massachusetts]
- 76/09/13 Duke Haiduk to Andy Mickel: [Pascal on DEC-10 for teaching at West Texas State University]
- 76/09/16 Olivier Lecarme to Andy Mickel: [European distribution of Pascal Newsletter, PUG session at IFIP '77, book by Bill Atwood, news about Pascal in France (activities and efforts), Pascal language publication notation, comments on Tim Bonham's paper news of translation of the book Systematic Programming into French]
- 76/09/17 Robert Novak to Andy Mickel: [remarks in reply to Eisenberg's article on formatted input.]
- 76/09/22 Stephen Young to Andy Mickel: [Impressed with Pascal - should have a Pascal standards committee]
- 76/09/29 Tony Addyman to Andy Mickel: [Pascal-6000 details, Pascal standards group should be formed.]

Open Forum:

- 76/11/18 David Barron to Andy Mickel: [Against the continuation of Fortran "carriage control character" conventions into Pascal]
 76/12/09 Andy Mickel to Niklaus Wirth: [PUG should have your reaction to the issue of formal standardization of Pascal]
 76/12/10 Andy Mickel to Chris Jacobi: [PUG needs the results of the Pascal-P questionnaire and sites using Pascal-P]
 77/01/02 Arthur Brown to Andy Mickel: [A Pascal standards committee should be set up - standard Pascal should be close to the Revised Report and extensions to standard Pascal compilers should be written in standard Pascal.]
 77/01/03 Jim Miner to Andy Mickel: [Revised Report good in its stated aim; but production uses of Pascal dictate a need for a standard. Fear that a committee will decide what the standard will be - current users are vulnerable because we have an investment in code.]

Implementation Notes: General Information, Checklist, Pascal-P, PascalJ, Software Writing Tools, B6700/7700, CII Iris 80, 10070, DEC PDP-8, DEC PDP-11, Foxboro Fox-1 IBM 360/370, IBM 1130, Interdata 8/32, Univac 90/70, 1100, Xerox Sigma 6/9, Sigma 7.

Pascal Newsletter #8, May, 1977.

Editor's Contribution: Renewal reminder, new developments with microprocessor Pascal, PUG and Pascal Newsletter, Pascal publicity, future, backissues, membership, end of the year acknowledgements.

Here and There: News from Pascalers, 2 Pascal get togethers planned at IFIP '77, and at ACM '77, Report on the University of Southampton Pascal Symposium, large Books and Articles section with new policy announced, first news about DoD-1 (a common language for U.S. Department of Defense use), a book review, several Pascal applications reported.

Articles:

"Development of a Pascal Compiler for the C.I.I. Iris 50, A Partial History"
 - Olivier Lecarme
 [The history which is the subject of the present paper takes place in the University of Nice, a medium-scale University with about fifteen thousand students. The history of the Pascal Compiler development covers several attempts (illustrated by "T" diagrams) and finally a description of the nearly completed successful effort.]

"A Further Defence of Formatted Input"
 - Brian Meekings
 [In PUGN #7, Barron and Mullins attempt to demolish the case for formatted input. Without wishing to blow up the controversy beyond reasonable proportion, I would like to add a voice in favour of formatting. The addition of formatted input to supplement the existing unformatted input facilities, can only enhance an already versatile language.]

"Proposals for Pascal"
 - George H. Richmond
 [A laundry list of idealized proposals are presented making the case for improvement in the areas of: the representation of Pascal for computer input, compile options, internal character set, removal of current restrictions and asymmetries, the program declaration set, variant records, the case statement, boolean expressions, constants, declarations, and constructors, value initialization and own variables, procedure and function types for compile time checking, dynamic array parameters, new basic types and operators, transfer functions, extension of relational operators to structured types, files and text files, formatted input and output, file handling, overlays, and preambles and postamble.]

"A Proposal for Increased Security in the Use of Variant Records"
 - William Barabash, Charles R. Hill, and Richard B. Kieburz
 [The use of variant records in most Pascal Implementations is dangerous because most compilers do not emit a check for conformity with the value of the tagfield

- 76/10/09 Rich Cichelli to Andy Mickel: [Contribution of William Waite's dues to PUG "Which Language?" article in British Computer Society Bulletin]
 76/10/11 Charles Hedrick to Andy Mickel: [Pascal versus SAIL and PL/1 in AI work, Pascal on the DEC-10 at the University of Illinois]
 76/10/15 Niklaus Wirth to Andy Mickel: [disagreement with the policy of printing private letters and letters to the editor]
 76/10/21 Duke Haiduk to Andy Mickel: [liked Pascal Newsletter #5, Brinch Hansen Pascal on the DEC-10?]
 76/10/22 Arthur Sale to Andy Mickel: [series of letters between Southampton and Tasmania may be of interest; Pascal implementation proliferation; B6700 Pascal]
 76/10/04 Judy Mullins to Arthur Sale: [ICL 1900 / 2900 Pascal: I/O, standardization, compiler options, else in case, syntactic sugar]
 76/10/22 Arthur Sale to Judy Mullins: [standards, character sets, B6700 commenting conventions, else in case, mixed languages, files, diagnostics, compiler options sets, bounds checking, B6700s: arrays and off stack storage, pointers speed & space]
 76/10/29 Jonathan Sachs to Andy Mickel: [Interest in the Tokyo 370 compiler at Trans Union Systems Corporation]
 76/11/04 Tim Bonham to PUG membership: [Comments on Jacobi's Dynamic Array Parameters standardization, Pascal on the IBM System 3?, CDC 3200?]

Implementation Notes: General Information, Checklist, Pascal-P, Concurrent Pascal, Pascal prettyprinter, Machine Dependent Implementations: B1700, B4700, B6700, 7700 CII 10070, Iris 50, Iris 80, CDC-6000, CDC-7600, CRAY-1, DECsystem 10, DEC PDP-11 Foxboro FOX-1, HP-2100, 3000, Honeywell H316, level 66, IBM 360/370, IBM 1130 Interdata 7/16, Motorola 6800, Prime P-400, Siemens 4004-157, Univac 1100, Varian V73, Xerox Sigma 6, 7, 9.

Pascal Newsletter #7, February, 1977.

Editor's Contribution: Promoting Pascal Usage (experience at the University of Minnesota); Pascal and Standards - conventionalizing extensions; PUG and Pascal Newsletter mechanics.

Here and There: News from Pascalers, a new textbook, roster update.

Articles:

"Life, Liberty, and the Pursuit of Unformatted Input"
 - David Barron and Judy Mullins
 [In PUGN #5, Eisenberg presents three examples which, he claims, demonstrate the necessity for formatted input. This note attempts to demolish those claims. ...WARNING: FORTRAN CAN IMPAIR YOUR JUDGEMENT.]

"Pascal Printer Plotter"
 - Herb Rubenstein
 [This printer plotter consists of four Pascal callable procedures. A high speed line-printer or a 132 column hard copy interactive terminal is used for output. The plots are X-Y graphs scaled to fit a single sheet of printer paper. Axis labels and axes are automatically set up. Plots may be overlaid and expansions can be performed to blow up tiny pieces of a plot.]

"Yet Another Look at Code Generation for Pascal on CDC 6000 and Cyber Machines"
 - Lawrence A. Liddiard
 [Pascal 2 is amenable to several different methods of compiler length reduction. As a fellow compiler writer (although since MNF is written in machine language it may be compared with the last of the dinosaurs speaking to Homo sapiens), I would rather see the full language specifications and one standard compiler, than to see small subsets such as Pascal-S. For this reason I think it essential to improve Pascal 2 and with the reductions discussed in this article it should be possible to obtain load lengths of approximately 30K octal for the full language rather than the current 44K octal on a CDC 6000 (i. e. a reduction by one-third).

when a variant field is referenced. Indeed, the latest version of the Revised Pascal Report defines a language in which the tagfield may even be absent, making conformity checks impossible! Even so, when the tagfield is present and the compiler does emit conformity checks automatically, the programmer still has the ability to dynamically assign values to the tagfield.]

"Update on UCSD Pascal Activities"
- Ken Bowles

[A potpourri of lively events at the University of California, San Diego is reported including their microprocessor Pascal system on LSI-11, Z-80, 8080, 6502, and 6800 based systems. Also reports on LSI-11 hardware, other micro hardware, a proposal for a manufacturer independent Pascal System, news about an introductory Pascal textbook, the UCSD B6700 compiler, and a sample graphics picture from the LSI-11 system are given.]

"Some Comments on Pascal I/O"
- Chris Bishop

[While admitting that Pascal has I/O specifications involving the concept of files and the GET and PUT statements that are consistent with the flavour of the language and with theoretical manipulation of data, I feel that it is lacking in simple, easy to use I/O and in flexible I/O.]

Open Forum:

- 77/01/14 Nick Solntseff to Andy Mickel: [The nature of standardization efforts on Pascal and perhaps operating system independence as well]
- 77/01/12 Michael Condict to Andy Mickel: [Comment that "slow array" more appropriate than Rich Cichelli's "long array" in PUGN #6 article]
- 77/01/04 Larry Landis to G. Michael Schneider: [An endorsement of Schneider's standards proposals in PUGN #6 article]
- 77/02/14 Robert Fraley to Andy Mickel: [A case for revising Pascal -> 3 mandatory extensions for Pascal so that it can compete with FORTRAN: parametric arrays shared variables in separate compilations, and input formatting.]
- 77/01/24 Mike Hagerty to Andy Mickel: [On the standard, mods to the standard, mods to the implementation, available software, otherwise in case]

Special Topic - Standards

- (* A very important exchange regarding standards and conventionalized extensions follows. At the Southampton Pascal Symposium, Tony Addyman made the case for a formal ISO standard without a standards committee through BSI. Votes were taken which called for standardizing the Revised Report with semantics tightened up, adopting a set of conventionalized extensions, and a list of designated extensions not to be conventionalized. *)
- 77/01/31 Niklaus Wirth to Andy Mickel: [Regarding standards, extending Pascal, Standard Pascal, Recommended set of extensions: dynamic array parameters, array and record constructors; possibility of unnecessary but convenient extensions: default in case lists, and formatted input. Other extensions per se for individual computer systems admissable but they have no place in the Standard language. Various comments on PUGN#6, especially regarding criticism of Pascal.]
- 77/02/09 Jørgen Steensgaard-Madsen to Andy Mickel: [Comments invited by Wirth on initialization of variables, dynamic arrays, exhaustive specification of parameters,

- the case statement, and handling of TEXT variables.]
- 77/03/07 Richard Kieburz to Niklaus Wirth: [Comments invited by Wirth on complete typing of formal procedure parameters, field width specifications in the arguments of the procedure write. On suggested extensions - relax the restriction on the maximum cardinality of set types, typed structured constants, and variable length strings. Also comments on dynamic arrays, array and record constructors, default in case lists, and formatted input.]
- 77/03/29 Andy Mickel to Southampton Symposium: [The Future of Pascal (Extensions and Standardization). A summary of the present state of affairs around Pascal and the desire for a standard; desirable goals for Pascal and current problems; consideration of a standard.]
- 77/04/07 Tony Addyman to Andy Mickel: [News on BSI / ISO standardization effort. A three page "attention list" of problems in the Pascal Report.]
- 77/04/24 Andy Mickel to Tony Addyman: [When in the BSI Working Group, don't forget the principles: "don't confuse the language with the implementation" and "some aspects are intentionally left undefined in Pascal and must be defined by implementation", possible meanings for omissions in the Revised Report.]
(* * * End of Special Topic: Standards * * *)
- 77/01/28 Arthur Sale to Andy Mickel: [Pascal has more to fear from its friends than its enemies, defense of editorial attack in PUGN6; Pascal Files - are Pascal's files inadequate?, are files variables?, is the best way to random access through slow array of...? What relation is there between Pascal files and our operating system files? Pascal's two greatest dangers are from naive extensions and Pascal fanaticism. The language has defects; it has strengths. Let's be a bit more cautious.]
- 77/02/14 Arthur Sale to Andy Mickel: [3 criticisms: 1) Sea mail distribution of PUGN overseas unacceptable, 2) Editorial sniping, 3) Pascal Support - Bill Waite's criteria. Distribution of software; we don't need crusaders yet; despite bits of rubbish PUGN serves a very useful purpose, publishing Arthur Sale - Judy Mullins correspondence.]
- 77/04/26 Andy Mickel to Arthur Sale: [Apologies for editorial sniping, reasons for seemail distribution of PUGN, Pascal files, CDC bias, PUGN's non-academic membership Pascal usage at Minnesota.]
- 77/03/04 Olivier Lecarme to Andy Mickel: [PUG produces PUGNs faster than I can read them. CII Iris 50 news, Pascal Subgroup formed in AFCET (French counterpart of ACM), compiler writing system.]
- 77/03/28 Nick Fiddian to Andy Mickel: [Plea to recognize the value to others of the software products we originate; invest accordingly in faithful standardization - down with backstreet implementors.]

Implementation Notes: Checklist, General Information, Microprocessors, Software Writing Tools, Pascal-P4 corrections, Pascal Trunk Compiler, PascalJ, Modula, Feature Implementation Notes: Reading and Writing Scalars (Arthur Sale), Pointer Values (Arthur Sale), Pointer Tests (Andy Mickel). Machine Dependent Implementations: B3700 / B4700, B6700, Computer Automation LSI-2, CDC Cyber 18, CDC 6000,7000, Cyber 70,170, Data General Nova, DEC-10, DEC PDP-11, HP-2100, Honeywell H66, IBM 360/370, IBM 1130, ICL 1900 / 2900, Intel 8080, Motorola 6800, Nanodata QM-1, Norsk Data Nord 10, SEMS T1600/Solar, Siemens 4004, 7000, TI ASC, TI 990/9900, Univac 90/70, U1100, Varian V70, Zilog Z-80.

ROSTER INCREMENT (77/12/31)

The names listed below represent people who have renewed, changed address or joined PUG since the roster was printed in PUGN #9/10.

01002	HENRY F. LEDGARD/ COMPUTER AND INFO. SCI./ U OF MASSACHUSETTS/ AMHERST MA 01002/ (413) 545-2744/ (413) 545-1332
01451	RALPH S. GOODELL/ HILLCREST DRIVE/ HARVARD MA 01451/ (617) 456-8090
01545	JOHN DE ROSA JR./ 32-G BRANDYWINE DRIVE/ SHREWSBURY MA 01545
01581	JOHNNY STOVALL/ 15 TURNPIKE RD./ WESTBORO MA 01581/ (617) 366-8911
01609	STEPHEN R. ALPERT/ COMP. SCI. DEPT./ WORCESTER POLYTECHNIC INSTITUTE/ WORCESTER MA 01609/ (617) 753-1411 X416
01701	MARGARETTA HOMMEL/ 43 ADAMS ROAD/ FRAMINGHAM MA 01701/ (617) 879-6848/ (617) 890-8460 X208 X351
01701	BARRY F. MARGOLIUS/ DEPT. OF COMPUTER SCI./ FRAMINGHAM STATE COLLEGE/ FRAMINGHAM MA 01701/ (617) 872-3501 X224/ (617) 266-6648 (HOME)
01701	ROBERT J. OBERG/ DEPT. OF MATH AND COMPUTER SCIENCE/ FRAMINGHAM STATE COLLEGE/ FRAMINGHAM MA 01701/ (617) 852-3501

01730 ROGER D. ROLES/ COMPUTERVISION CORP./ 201 BURLINGTON RD/ BEDFORD MA 01730/ (617) 275-1800 X212
 01741 STEPHEN KLEIN/ 188 JUDY FARM ROAD/ CARLISLE MA 01741
 01752 CARL W. SCHWARCZ/ MR 1-2/E27/ DIGITAL EQUIPMENT CORP./ 200 FOREST STREET/ MARLBORO MA 01752/ (617) 481-9511
 01754 ROBERT TROCCHI/ EDUCATIONAL PRODUCTS GROUP/ DIGITAL EQUIPMENT CORP./ PARKER STREET - PK3-1/M40/ MAYNARD MA 01754/ (617) 493-3475
 01776 RANDY ENGER/ 28 BRIAR PATCH LANE/ SUDBURY MA 01776
 01778 PAUL BARR/ EQUIPMENT DIVISION J9/ RAYTHEON CO./ BOSTON POSTROAD/ WAYLAND MA 01778/ (617) 358-2721 X2825
 01810 ROBERT I. DEMROW/ 11 LINDA DRIVE/ ANDOVER MA 01810
 01852 EDWARD STEEN/ 119 SHERMAN STREET/ LOWELL MA 01852/ (617) 454-9320
 01886 RICHARD KRASIN/ FIRST DATA CORP./ 1 MAIN STREET/ WESTFORD MA 01886
 01960 SAM CARPENTER/ 22 PULASKI ST. APT. B-7/ PEABODY MA 01960/ (617) 532-0669
 02035 WARREN R. BROWN/ D.330/ THE FOXBORO COMPANY/ 38 NEPONSET AVE./ FOXBORO MA 02035/ (617) 543-8750 X2023
 02114 RICHARD PITKIN/ COMPUTER NETWORK/ MASS. STATE COLLEGE/ 150 CAUSEWAY ST./ BOSTON MA 02114/ (617) 727-2530
 02132 BILL SOUTHWORTH/ 30 POTOMAC ST./ W. ROXBURY MA 02132/ (617) 323-4537
 02138 FRED LUHMANN/ ABT ASSOCIATES INC./ 55 WHEELER ST./ CAMBRIDGE MA 02138/ (617) 492-7100 X424
 02138 JAMES S. MILLER/ INTERMETRICS INC./ 701 CONCORD AVE./ CAMBRIDGE MA 02138/ (617) 661-1840
 02138 DENNIS J. MURPHY/ ABT ASSOCIATES INC./ 55 WHEELER ST/ CAMBRIDGE MA 02138/ (617) 492-7100
 02138 ROBERT E. WELLS/ BOLT BERANEK AND NEWMAN INC./ 50 MOULTON STREET/ CAMBRIDGE MA 02138/ (617) 491-1850 X694
 02139 CHARLES L. BROOKS/ ABT ASSOCIATES INC./ 55 WHEELER STREET/ CAMBRIDGE MA 02139/ (617) 492-7100
 02142 JAMES STEINBERG/ 23/ DOT/TSC/ KENDALL SQUARE/ CAMBRIDGE MA 02142
 02154 BRYAN HOPKINS/ EKS/ 200 TRAPELO ROAD/ WALTHAM MA 02154/ (617) 893-3500 X277
 02159 LAWRENCE F. CRAM/ 64 BOWEN STREET/ NEWTON MA 02159
 02165 THOMAS M. ATWOOD/ 70 BARNSTABLE RD./ W. NEWTON MA 02165/ (617) 235-8171 X131
 02165 JOHN C. MILLER/ 105 CHERRY STREET/ W. NEWTON MA 02165/ (617) 272-7070 X160
 02172 FRED ELLENSTEIN/ 68 SPRING STREET/ WATERTOWN MA 02172/ (617) 924-2248
 02840 DAVID TAFES/ 42 THIRD STREET/ NEWPORT RI 02840/ (401) 847-3770
 02912 ATTN: L.A.M.B.D.A./ BROWN UNIVERSITY/ BOX G/ PROVIDENCE RI 02912/ (401) 863-3162
 02912 READ T. FLEMING/ PROGRAM IN COMPUTER SCIENCE/ BROWN UNIVERSITY/ BOX F/ PROVIDENCE RI 02912/ (401) 863-3088
 03060 BETTY BUXTON/ NCA 1-3220/ SANDERS ASSOCIATES INC./ 95 CANAL STREET/ NASHUA NH 03060/ (603) 885-5314
 03755 MICHAEL MCKENNA/ TIME SHARE CORP./ BOX 683/ HANOVER NH 03755/ (603) 448-3838
 03824 WILLIAM J. VASILIOU JR./ COMPUTER SERVICES/ KINGSBURY HALL/ U OF NEW HAMPSHIRE/ DURHAM NH 03824/ (603) 862-2323
 06106 A. E. SAPEGA/ ENGINEERING DEPT./ TRINITY COLLEGE/ HARTFORD CT 06106/ (203) 527-3151 X202
 06268 TIM RAND/ P.O. BOX 98/ STORRS CT 06268
 06437 PAUL KOHLBRENNER/ 261 DUNK ROCK ROAD/ GUILFORD CT 06437/ (203) 453-9540
 06477 MICHAEL BEHAR/ 428 WINDY HILL RD./ ORANGE CT 06477/ (203) 878-7141
 06520 DICK OSGOOD/ YALE COMPUTER CENTER/ 175 WHITNEY CENTER/ NEW HAVEN CT 06520/ (203) 432-4080
 06810 RONA GURKEWITZ/ 181 WHITE STREET/ DANBURY CT 06810
 06901 DOUGLAS M. GRANT/ NATIONAL CSS/ 500 SUMMER STREET/ STAMFORD CT 06901/ (203) 327-9100
 07054 ROBERT KAST/ 350 BALDWIN ROAD APT. F4/ PARSIPPANY NJ 07054
 07470 HAL PACE/ KEARFOTT DIV. - DEPT. 5760/ SINGER CO./ 150 TOTOWA ROAD/ WAYNE NJ 07470/ (201) 256-4000 X3503
 07724 CHRISTOPHER J. HENRICH/ SOFTWARE DEVELOPMENT/ INTERDATA INC./ 106 APPLE STREET/ TINTON FALLS NJ 07724/ (201) 229-4040
 08077 FOREST VAN SISE SHAFER/ COMCON INC./ 504 U.S. ROUTE 130 AT HIGHLAND AVE./ CINNAMINSON NJ 08077
 08512 WILLIAM G. HUTCHISON JR./ N 191 PRINCETON ARMS/ CRANBURY NJ 08512/ (609) 443-6631
 08618 WILLIAM J. K. HARRINGTON/ 70 MAIN BOULEVARD/ TRENTON NJ 08618
 08826 GEORGE B. DIAMOND/ DIAMOND AEROSOL CORP./ RD #1/ GLEN GARDNER NJ 08826
 08854 ATTN: CCIS LIBRARY HILL CENTER/ BUSCH CAMPUS/ RUTGERS UNIV./ P.O. BOX 879/ PISCATAWAY NJ 08854/ (201) 932-2296
 08854 JIM STEWART/ 195B PLEASANT VIEW ROAD/ PISCATAWAY NJ 08854
 08876 RODERICK MONTGOMERY/ HEALTH PRODUCTS RESEARCH INC./ 3520 U.S. ROUTE 22/ SOMERVILLE NJ 08876/ (201) 534-4148
 08903 CHARLES HEDRICK/ COMPUTER SCIENCE DEPT./ RUTGERS UNIV./ HILL CENTER/ NEW BRUNSWICK NJ 08903
 09098 DOUG FORSTER/ P.O. BOX 1027 UNIT AA/ APO NY 09098
 10003 WILLIAM HENRY/ 117 E. TENTH ST./ NEW YORK NY 10003/ (212) 673-6944
 10009 NORMAN D. WHALAND/ 430 EAST 9TH STREET - APT. 15/ NEW YORK NY 10009
 10010 ROBERTO MINIO/ SPRINGER-VERLAG INC./ 175 FIFTH AVE/ NEW YORK NY 10010/ (212) 477-8316
 10011 JON A. SOLWORTH/ 7 WEST 14TH ST APT 15A/ NEW YORK NY 10011/ (212) 243-2183
 10012 EDWARD R. FRIEDMAN/ CIMS/ NEW YORK UNIVERSITY/ 251 MERCER ST./ NEW YORK NY 10012/ (212) 460-7100/ (212) 460-7293
 10012 ANDREW P. VALENTI/ COURANT INST. OF MATHEMATICAL SCIENCE/ NEW YORK UNIV./ 251 MERCER ST./ NEW YORK NY 10012/ (212) 641-0274
 10016 RAMON TAN/ 305 E. 40TH ST. APT. 12W/ NEW YORK NY 10016/ (212) 682-1013
 10019 MARK STAHLMAN/ COMPUTRON INC./ 888 7TH AVENUE - 25TH FLOOR/ NEW YORK NY 10019
 10024 PAUL SPRECHER/ 241 WEST 77TH STREET/ NEW YORK NY 10024
 10024 DONALD WARREN/ 130 WEST 81ST STREET APT 7/ NEW YORK NY 10024
 10025 HOWARD D. ESKIN/ CENTER FOR COMPUTING ACTIVITIES/ ROOM 712/ COLUMBIA UNIVERSITY/ 612 W. 115TH ST./ NEW YORK NY 10025/ (212) 280-2874
 10027 LARRY ARONSON/ CENTER FOR COMPUTING ACTIVITIES/ COLUMBIA UNIVERSITY/ 612 W 115TH ST./ NEW YORK NY 10027/ (212) 280-2698
 10510 JERRY S. SULLIVAN/ PHILIPS LABORATORIES/ 345 SCARBOROUGH ROAD/ BRIARCLIFF MAN NY 10510/ (914) 762-0300
 10573 TIMOTHY P. ROBERTS/ KERN INSTRUMENTS INC./ 111 BOWMAN AVE./ PORT CHESTER NY 10573/ (914) 939-0200
 11210 PAUL S. KLARREICH/ 2809 BEDFORD AVE./ BROOKLYN NY 11210/ (212) 859-1408
 11439 LYNN S. MARTIN/ DEPT. OF ENGLISH/ ST. JOHN'S UNIV./ GRAND CENTRAL AND UTOPIA PARKWAYS/ JAMAICA NY 11439/ (212) 969-8000 X387
 11740 M. WAITE/ HAZELTINE CORP./ GREENLAWN NY 11740/ (516) 261-7000 X687

11756 ROBERT SCHUTZ/ 93 MERIDIAN ROAD/ LEVITTOWN NY 11756
 11794 RICHARD B KIEBURTZ/ DEPT. OF COMPUTER SCI./ SUNY AT STONY BROOK/ STONY BROOK NY 11794/ (516) 246-5987/ (516) 246-7146
 11794 GENE ROLLINGS/ COMPUTER SCIENCE DEPT./ SUNY - STONY BROOK/ STONY BROOK NY 11794/ (516) 246-4383
 11797 PAUL ZILBER/ ONTEL CORP./ 250 CROSSWAYS PARK DRIVE/ WOODBURY NY 11797/ (516) 364-2121
 12210 GARYO O'SCHENECTADY/ 144 LANCASTER ST./ ALBANY NY 12210
 13201 J. DANIEL GERSTEN/ COMPUTED IMAGE ENG. - CSP 3-21/ GENERAL ELECTRIC CO./ SYRACUSE NY 13201/ (315) 456-7366
 13440 STEPHEN B. WATERS/ ROME SENTINEL COMPANY/ 333 W. DOMINICK STREET/ ROME NY 13440/ (315) 337-4000
 14072 LEO CHRZANOWSKI/ 67 WARD PARK ROAD/ GRAND ISLAND NY 14072
 14127 F. DOUGLAS ROBINSON/ 57 TANGLEWOOD WEST/ ORCHARD PARK NY 14127/ (716) 843-7142 (WORK)/ (716) 662-4093 (HOME)
 14226 STUART W. ROWLAND/ COMPUTER SCIENCE DEPT./ SUNY - BUFFALO/ 4226 RIDGE LEA ROAD/ AMHERST NY 14226/ (716) 831-1351
 14454 ANTHONY E. HOFFMAN/ MATHEMATICS DEPT./ SUNY-CAS/ GENESEE NY 14454/ (716) 243-3833
 14850 WILLIAM LYCZKO/ SOFTWARE DEVELOPMENT/ NCR CORPORATION/TERMINAL SYSTEMS/ 950 DANBY ROAD/ ITHACA NY 14850/ (607) 273-5310/ X251 X254
 15213 ATTN: EARL L. MOUNTS-COMP. SCI. LIBRAR/ E & S LIBRARY/ SCIENCE HALL/ CARNEGIE-MELLON UNIVERSITY/ PITTSBURGH PA 15213/ (412) 578-2426
 15213 DAVID B. GROUSE/ GRAPHIC ARTS TECHNICAL FOUNDATION/ 4615 FORBES AVE/ PITTSBURGH PA 15213
 15213 KEVIN WEILER/ SCHOOL OF URBAN AND PUBLIC AFFAIRS/ INSTITUTE OF PHYSICAL PLANNING/ CARNEGIE MELLON UNIV/ SCHENLEY PARK/ PITTSBURGH PA 15213
 (412) 578-2177
 15461 RON MAHON/ VIDEO LINK/ P.O. BOX 688/ MASONTOWN PA 15461/ (412) 583-7786
 15701 HOWARD E. TOMPKINS/ COMPUTER SCIENCE DEPT/ INDIANA UNIVERSITY OF PA/ INDIANA PA 15701/ (412) 357-2524
 16057 PETER RICHETTA/ 129A WEST WATER STREET/ SLIPPERY ROCK PA 16057/ (412) 794-3531
 17019 E. R. BEAUREGARD/ P.O. BOX 357/ DILLSBURG PA 17019
 17837 ATTENTION: MARJORIE HEINE/ FREAS-ROOKE COMPUTER CENTER/ BUCKNELL UNIVERSITY/ LEWISBURG PA 17837/ (717) 524-1436
 18015 DAVID B. ANDERSON/ DEPT. OF MATHEMATICS/ 14 CHRISTMAS-SAUCON/ LEHIGH UNIVERSITY/ BETHLEHEM PA 18015/ (215) 683-5086
 18015 CRAIG PAYNE/ LEHIGH UNIV./ P.O. BOX 22A/ BETHLEHEM PA 18015/ (215) 867-6367
 18017 ROBERT COLE/ 782 BARRYMORE LANE/ BETHLEHEM PA 18017
 18042 JOHN A. WEAVER/ ANPA - RESEARCH INSTITUTE/ P.O. BOX 598/ EASTON PA 18042/ (215) 867-1085
 18049 JOHN W. IOBST/ 22 N. KEYSTONE AVE./ EMMAUS PA 18049/ (215) 965-4677
 18938 BILL CHESWICK/ DARIEN 15B / VILLAGE 2/ NEW HOPE PA 18938/ (215) 866-4491
 19010 DONALD B. KLEIN/ 145 LOWRY'S LANE/ ROSEMONT PA 19010
 19044 JAMES P. MCILVAINE IV/ BRIDGEPORT-TEXTRON/ 200 PRECISION RD./ HORSHAM PA 19044/ (215) 674-2700
 19046 DAN MORTON/ 701 WASHINGTON LANE/ JENKINTOWN PA 19046/ (215) 885-2443/ (215) 895-2259
 19085 STEPHEN W. CHING/ DEPT OF ELECTRICAL ENGINEERING/ COMPUTER SCIENCE/ VILLANOVA UNIVERSITY/ VILLANOVA PA 19085/ (215) 527-2100 X631
 19087 MARK HIPPE/ GEOMETRIC DATA CORP./ 999 WEST VALLEY ROAD/ WAYNE PA 19087/ (215) 687-6550
 19101 G. KEVIN DOREN/ P.O. BOX 8191/ PHILADELPHIA PA 19101/ (215) 963-0465/ (215) 963-0551
 19122 EARL RALEY/ COMPUTER ACTIVITY/ ACADEMIC SERVICES/ TEMPLE UNIV./ PHILADELPHIA PA 19122/ (215) 787-8527
 19151 CLIFTON CHANG-CHAO TING/ 879 WYNNEWOOD ROAD - 1ST FLOOR/ PHILADELPHIA PA 19151/ (215) 878-7231
 19301 DAVID M. ADAMS/ CSG/T/ BURGESS CORPORATION/ P.O. BOX 203/ PAOLI PA 19301/ (215) 648-2000
 19335 TOM KELLY/ 58-B MEADOWLAKE DRIVE/ DOWNINGTOWN PA 19335/ (215) 269-3626
 19341 JEFFREY D. STROOMER/ 224 HERITAGE LANE/ EXTON PA 19341/ (216) 363-1948
 19438 DONALD A. KEFFER/ 252 MANOR ROAD/ HARLEYSVILLE PA 19438
 19440 V. LALITA RAO/ HATFIELD VILLAGE APARTMENTS #T1-5/ HATFIELD PA 19440/ (215) 865-6448
 19711 JOHN D. EISENBERG/ COMPUTING CENTRE/ SMITH HALL/ U OF DELAWARE/ NEWARK DE 19711/ (302) 738-8441 X57 (OFFICE)/ (302) 453-9059 (HOME)
 19711 WILLIAM S. PAGE/ 23 OLD MANOR ROAD/ NEWARK DE 19711/ (302) 731-5988
 19898 C. E. BRIDGE/ ENGINEERING DEVELOPMENT LAB/ E. I. DU PONT DE NEMOURS AND CO./ 101 BEECH STREET/ WILMINGTON DE 19898/ (302) 774-1731
 19898 STEPHEN C. SCHWARM/ E.I. DU PONT DE NEMOURS CO./ 101 BEECH ST./ WILMINGTON DE 19898/ (302) 774-1669
 20006 KENNETH R. JACOBS/ FEDERAL SYSTEMS DIVISION/ ADP NETWORK SERVICES/ 2011 EYE STREET NW/ WASHINGTON DC 20006/ (202) 872-0580
 20014 KEITH E. GORLEN/ 2017 BLDG.12A/ NATIONAL INST. OF HEALTH/ BETHESDA MD 20014/ (301) 496-5361
 20014 TERRY P. MEDLIN/ SCIENTIFIC RESEARCH UNIT - DPSA/ B23 BLDG 30/ NATIONAL INSTITUTE OF DENTAL HEALTH/ BETHESDA MD 20014/ (301) 496-1621
 20014 JOHN M. SHAW/ BLDG 36 / ROOM 2A29/ NATIONAL INSTITUTES OF HEALTH/ BETHESDA MD 20014/ (301) 496-3204
 20016 JOSEPH P. JOHNSON/ 3520 QUEBEC ST. NW/ WASHINGTON DC 20016/ (202) 362-8523
 20036 MARGERY AUSTIN/ THE URBAN INSTITUTE/ 2100 M STREET NW/ WASHINGTON DC 20036/ (202) 223-1950 X486
 20041 DAVID AULT/ COMPUTER SCIENCE/ VPI AND SU/ P.O. BOX 17186/ WASHINGTON DC 20041/ (703) 471-4600
 20229 STEVE O'KEEFE/ 7328/ U.S. CUSTOMS SERVICE/ 1301 CONSTITUTION AVE. N.W./ WASHINGTON DC 20229/ (202) 566-2974
 20375 PETER A. RIGSBEE/ CODE 5494/ NAVAL RESEARCH LABORATORY/ WASHINGTON DC 20375/ (202) 767-3318
 20770 LEO DAVIS/ 40 LAKESIDE DRIVE/ GREENBELT MD 20770
 20770 CAROL B. HOWELL/ P.O. BOX 326/ GREENBELT MD 20770
 20784 EDWARD D. ROTHE/ 7101 VARNUM ST/ LANDOVER HILLS MD 20784
 20855 BOB ROGERS/ 18625 AZALEA DRIVE/ DERWOOD MD 20855
 21010 PAUL H. BROOME/ BIOPHYSICS BRANCH/ CHEMICAL SYSTEMS LAB/ RESEARCH DIVISION/ PROVING GROUND/ ABERDEEN MD 21010/ (301) 671-3489
 21045 JOHN FRINK/ 5304 THUNDER HILL ROAD/ COLUMBIA MD 21045/ (202) 394-2396
 21045 RICHARD LLEWELLYN/ 5355 RED LAKE/ COLUMBIA MD 21045/ (301) 765-4570
 21045 RAINER F. MCCOWN/ MCCOWN COMPUTER SERVICES/ 9537 LONG LOOK LANE/ COLUMBIA MD 21045/ (301) 730-0379
 21235 LESTER SACHS/ OPERATIONS/ MS 3-0-25/ SOCIAL SECURITY ADMINISTRATION/ 6401 SECURITY BOULEVARD/ BALTIMORE MD 21235
 21793 PAUL C. BERGMAN/ DIGITAL SYSTEMS CORP./ P.O. BOX 396/ WALKERSVILLE MD 21793/ (301) 845-4141
 22003 ROBERT C. JANKU/ 5112 ALTHEA DRIVE/ ANNANDALE VA 22003/ (703) 978-8384
 22042 ROBERT LEE SHARP/ P.O. BOX 2170/ FALLS CHURCH VA 22042
 22090 HENRY DAVIS/ ACUITY SYSTEMS INC./ 11413 ISAAC NEWTON SQUARE/ RESTON VA 22090/ (703) 471-4700 X243

22091 JAMES K. MOORE/ 12345 COLERAINE COURT/ RESTON VA 22091/ (703) 437-2338
 22101 H. F. HESSON/ ADVANCED RECORD SYSTEMS ENGINEERING/ WESTERN UNION/ 7916 WEST PARK DRIVE/ MCLEAN VA 22101
 22101 ROBERT L. STEELE II/ INCO. INC./ 7916 WEST PARK DRIVE/ MCLEAN VA 22101/ (703) 893-4330
 22152 MARK S. WATERBURY/ 8358 L DUNHAM CT./ SPRINGFIELD VA 22152/ (703) 451-8255
 22180 TRUMAN C. PEWITT/ 8507 COTTAGE STREET/ VIENNA VA 22180/ (703) 821-6321/ (703) 573-3192
 22209 WILLIAM A. WHITTAKER/ DARPA/ 1400 WILSON BLVD./ ARLINGTON VA 22209/ (202) 694-1139
 22901 ROBERT A. GIBSON/ WEST LEIGH/ 2380 KINGSTON RD/ CHARLOTTESVILLE VA 22901/ (804) 977-3233
 22901 STEPHEN P. MERSON/ SCHOOL OF ENGINEERING--DAMACS/ AERO-MATH BLDG./ UNIV. OF VIRGINIA/ CHARLOTTESVILLE VA 22901/ (804) 924-3917
 22903 ATTN: J. F. MCINTYRE - LIBRARIAN/ COMPUTING CENTER/ GILMER HALL/ UNIV OF VIRGINIA/ CHARLOTTESVILLE VA 22903/ (804) 924-3731
 23185 MICHAEL K. DONEGAN/ DEPT. OF MATH. & COMP. SCIENCE/ COLLEGE OF WILLIAM & MARY/ WILLIAMSBURG VA 23185/ (804) 253-4481
 23508 FRANCES L. VAN SCOY/ DEPT. OF MATH AND COMPUTING SCIENCES/ OLD DOMINION UNIV./ NORFOLK VA 23508/ (804) 489-6525
 23665 JOHN C. KNIGHT/ LANGLEY RESEARCH CENTER/ M/S 125A/ NASA/ HAMPTON VA 23665/ (804) 827-3875
 23669 JOHN CLARSON/ 303 TENDERFOOT COURT/ HAMPTON VA 23669
 27409 TOM TYSON/ COMPUTER LABS/ 505 EDWARDIA DRIVE/ GREENSBORO NC 27409/ (919) 292-5427
 27607 DONALD L. PARCE/ BUSINESS APPLICATIONS SYSTEMS INC./ 7334 CHAP EL HILL ROAD/ RALEIGH NC 27607/ (919) 851-8512
 27709 W. J. MEYERS/ DATA GENERAL CORP./ RESEARCH TRIANGLE PARK/ TRIANGLE PARK NC 27709
 30303 ROBERT N. MACDONALD/ INFORMATION SYSTEMS DEPT./ GEORGIA STATE UNIV./ UNIVERSITY PLAZA/ ATLANTA GA 30303/ (404) 658-3880
 30303 DARRELL PREBLE/ COMPUTER CENTER USER SERVICES/ GEORGIA STATE UNIVERSITY/ ATLANTA GA 30303
 30303 MORRIS W. ROBERTS/ DEPT. OF INFORMATION SYSTEMS/ GEORGIA STATE UNIV./ UNIVERSITY PLAZA/ ATLANTA GA 30303/ (404) 658-3882
 30310 JOE CELKO/ P.O. BOX 11023/ ATLANTA GA 30310/ (404) 753-7993
 30339 DOUGLAS MANN/ SCIENCE APPLICATIONS INC./ 2028 POWERS FERRY ROAD - SUITE 260/ ATLANTA GA 30339/ (404) 955-2663
 32303 C. EDWARD REID/ RT. 7 BOX 1257/ TALLAHASSEE FL 32303/ (904) 488-2451
 32306 JOHN H. BOLSTAD/ DEPT. OF MATHEMATICS/ FLORIDA STATE UNIV./ TALLAHASSEE FL 32306/ (904) 644-2580
 32901 GEORGE E. HAYNAM/ SYSTEMS DIVISION/ HARRIS CORP./ P.O. BOX 2080/ MELBOURNE FL 32901/ (904) 378-8118
 32901 TOM SPURRIER/ ELECTRONICS SYSTEMS DIVISION/ HARRIS CORP./ P.O. BOX 37/ MELBOURNE FL 32901/ (305) 727-4000
 32905 DENIS KERMICLE/ WOODLAKE DRIVE EAST APT. D-130/ PALM BAY FL 32905/ (305) 725-2417
 32935 ROBERT L. CHEEZEM JR./ 2192 CHERYL CT./ MELBOURNE FL 32935/ (305) 254-6522
 33313 S. HAYES/ DEVELOPMENT ENGR. LIBRARY/ SYSTEMS ENGR. LABS/ 6901 W. SUNRISE BLVD./ FT.LAUDERDALE FL 33313/ (305) 587-2900
 33313 STEVE MATUS/ MARKET PLANNING AND RESEARCH/ SYSTEMS ENGINEERING LABS/ 6901 W. SUNRISE BLVD./ FT.LAUDERDALE FL 33313/ (305) 587-2900
 35486 DONALD B. CROUCH/ DEPT.OF COMPUTER SCIENCE/ UNIV OF ALABAMA/ P.O. BOX 6316/ UNIVERSITY AL 35486/ (205) 348-6363
 35805 MIKE D. PESSONEY/ ANALYSTS INTERNATIONAL CORP./ 2317 BOB WALLACE AVE. SE/ HUNTSVILLE AL 35805/ (205) 533-4220
 37130 SAMUEL T. BAKER/ 1310 STONEWALL BLVD./ MURFREESBORO TN 37130/ (615) 896-3362 (HOME)/ (615) 741-3531 (OFFICE)
 37916 ATTENTION: CHARLES PFLEEGER/ COMP. SCI. DEPT./ U OF TENNESSEE/ KNOXVILLE TN 37916/ (615) 974-5067
 40205 A. CHARLES BUCKLEY/ DATA/INFORMATION SYSTEMS/ URBAN STUDIES CENTER/ ALTA VISTA ROAD - GARDENCOURT/ LOUISVILLE KY 40205/ (502) 588-6626
 40217 MICHAEL P. ROBINSON/ AMERICAN DATA MANAGEMENT SYSTEMS/ 2434 CRITTENDEN DRIVE - SUITE 200/ LOUISVILLE KY 40217/ (502) 637-9765
 43403 FRANK J. BATES JR./ OFFICE OF COMPUTATIONAL SERVICES/ BOWLING GREEN STATE UNIV./ BOWLING GREEN OH 43403/ (419) 372-2911
 43403 JOHN M. HEMPHILL/ DEPT. OF COMPUTER SCI./ BOWLING GREEN STATE UNIV./ BOWLING GREEN OH 43403/ (419) 372-2337
 43403 RICHARD T. THOMAS/ DEPT. OF COMPUTER SCIENCE/ BOWLING GREEN STATE UNIV./ BOWLING GREEN OH 43403/ (419) 372-2339
 44106 R. B. LAKE/ BIOMETRY/ WEARN BUILDING/ CASE WESTERN UNIV HOSPITALS/ CLEVELAND OH 44106/ (216) 444-3491
 44106 TOM NUTE/ SYS. & COMPUTER ENG./ CRAWFORD HALL/ CASE WESTERN RESERVE UNIV./ CLEVELAND OH 44106/ (216) 368-2800
 44107 STEVEN B. HALL/ 1599 ORCHARD GROVE/ LAKEWOOD OH 44107/ (216) 521-4178
 44115 ARTHUR C. DARTT/ CHEMISTRY DEPT./ CLEVELAND STATE UNIV./ EUCLID AT EAST 24TH STREET/ CLEVELAND OH 44115/ (216) 687-2473
 44116 BILL SHANNON/ 21345 HILLIARD/ ROCKY RIVER OH 44116/ (216) 331-8733
 44119 DAVID PESEC/ 21030 MILLER/ EUCLID OH 44119/ (216) 486-4716
 44691 ANN C. JOHNSTON/ RD 6/ HAPPY VALLEY RD./ WOOSTER OH 44691
 45036 ATTN: BETTE BOLLING-LIBRARIAN/ TECHNICAL INFORMATION CTR-ELECTRONICS/ CINCINNATI MILACRON INC./ LEBANON OH 45036/ (513) 494-1200
 45036 TOM MORAN/ PROCESS CONTROLS DIVISION/ CINCINNATI MILACRON/ MASON RD & RT #48/ LEBANON OH 45036
 46375 PHILIP T. HODGE/ 346 KENNEDY/ SCHERERVILLE IN 46375
 46530 JOE TORZEWSKI/ 51625 CHESTNUT ROAD/ GRANGER IN 46530/ (219) 272-4670
 47401 ANNA BUCKLEY/ WRUBEL COMPUTING CENTER/ INDIANA UNIV./ BLOOMINGTON IN 47401/ (812) 337-1911
 47401 ANTHONY J. SCHAEFFER/ 3510 DUNSTAN DR./ BLOOMINGTON IN 47401/ (812) 337-9137
 47805 ROBERT L. ARGUS/ 2603 THOMAS AVE. APT 4/ TERRE HAUTE IN 47805
 47907 ALLAN M. SCHWARTZ/ DEPT. OF COMPUTER SCIENCES/ MATH SCIENCES BUILDING/ PURDUE UNIVERSITY/ WEST LAFAYETTE IN 47907/ (317) 743-2473
 47907 EDWARD F. GEHRINGER/ DEPT. OF COMPUTER SCIENCE/ MATH SCIENCES BUILDING/ PURDUE UNIVERSITY/ W. LAFAYETTE IN 47907
 47907 SAUL ROSEN/ COMPUTING CENTER/ PURDUE UNIV./ W. LAFAYETTE IN 47907/ (317) 494-8235
 48100 MARK HERSEY/ 1114 MAIDEN LANE COURT APT. 112/ ANN ARBOR MI 48100/ (313) 994-3934/ (517) 355-1764 (OFFICE)
 48104 GREG WINTERHALTER/ HORIBA INSTRUMENTS/ 3901 VARSITY DRIVE/ ANN ARBOR MI 48104/ (313) 973-2171
 48104 KARL L. ZINN/ CTR. FOR RESEARCH ON LEARNING & TEACHI/ UNIV. OF MICHIGAN/ 109 EAST MADISON STREET/ ANN ARBOR MI 48104/ (313) 763-4410/ 763-0158
 48106 CHARLES G. MOORE/ ADP NETWORK SERVICES/ 175 JACKSON PLAZA/ ANN ARBOR MI 48106/ (313) 769-6800
 48824 JOHN B. EULENBERG/ COMP. SCI. DEPT./ MICHIGAN STATE U/ EAST LANSING MI 48824/ (517) 353-0831
 48824 HARRY G. HEDGES/ DEPT. OF COMP. SCI./ 400 COMPUTER CENTER/ MICHIGAN STATE UNIV/ EAST LANSING MI 48824/ (517) 353-6484
 48824 STEVEN L. HUYSER/ USER INFO. CENTER/ 313 COMPUTER CENTER/ MICHIGAN STATE U/ EAST LANSING MI 48824/ (517) 353-1800
 48824 MARK RIORDAN/ USER SERVICES/ COMPUTER LABORATORY/ MICHIGAN STATE UNIVERSITY/ EAST LANSING MI 48824/ (517) 353-1800
 49008 JACK R. MEAGHER/ COMPUTER SCIENCE AND MATHEMATICS/ WESTERN MICHIGAN UNIV./ KALAMAZOO MI 49008/ (616) 383-0095
 50011 AHMED KASSEM/ COMPUTATION CENTER/ 104 COMPUTER SCIENCE/ IOWA STATE UNIV./ AMES IA 50011/ (515) 294-8424
 52240 G. STEPHEN HRST/ 930 FAIRCHILD/ IOWA CITY IA 52240/ (319) 351-5253 (HOME)/ (319) 353-3935 (WORK)

52242 DONALD L. EPLEY/ DEPT. OF COMPUTER SCIENCE/ UNIV. OF IOWA/ IOWA CITY IA 52242/ (319) 353-5605
 52302 DENNIS SUTHERLAND/ 2835 25TH AVE./ MARION IA 52302/ (319) 395-4728
 52402 JAMES C. COZZIE/ 254 NORTHPOINTE N.E. - APT. 322/ CEDAR RAPIDS IA 52402
 53149 MICHAEL A. BEAVER/ ROUTE 3 BOX 271B/ MUKWONAGO WI 53149/ (414) 728-5531 X249
 53201 RICHARD E. NEUBAUER/ JOHNSON CONTROLS INC./ P.O. BOX 423/ MILWAUKEE WI 53201/ (414) 276-9200
 53219 JOHN G. DOBNICK/ 3171 S. 83 ST./ MILWAUKEE WI 53219/ (414) 963-5727
 53705 EDWARD K. REAM/ 508 FARLEY AVE. - APT. 5/ MADISON WI 53705
 53706 LARRY E. TRAVIS/ COMPUTER SCIENCE DEPT./ UNIV OF WISCONSIN - MADISON/ 1210 WEST DAYTON STREET/ MADISON WI 53706/ (608) 262-7971/ (608) 262-1204
 53719 LEN LINDSAY/ 5150 ANTON DR. #212/ MADISON WI 53719
 55057 CARL HENRY/ COMPUTER CENTER/ CARLETON COLLEGE/ NORTHFIELD MN 55057/ (507) 645-4431 X504
 55057 TIMOTHY W. HOEL/ ACADEMIC COMPUTER CENTER/ ST. OLAF COLLEGE/ NORTHFIELD MN 55057/ (507) 663-3097
 55066 TERRY MYHRER/ 1324 EAST AVENUE/ RED WING MN 55066
 55105 ATTN: COMPUTING SERVICES/ MACALESTER COLLEGE/ 1600 GRAND AVE/ ST. PAUL MN 55105/ (612) 647-6171
 55108 JAMES KREILICH/ 1408 ALBANY AVE./ ST. PAUL MN 55108/ (612) 644-1375
 55112 ED KATZ/ 3564 N. SNEILING/ ARDEN HILLS MN 55112/ (612) 636-3472
 55112 W. B. CHAPIN/ ARH 242/ CONTROL DATA CORP./ 4201 N. LEXINGTON/ ST. PAUL MN 55112/ (612) 483-4673
 55112 MARK RUSTAD/ 585 HARRIET AVE #213/ ST. PAUL MN 55112/ (612) 483-0589/ (612) 376-1143 (WORK)
 55165 ROBERT A. LAWLER/ MS U2M23/ UNIVAC/ P.O. BOX 3525/ ST. PAUL MN 55165/ (612) 456-3109
 55337 LARRY W. SMITH/ 125 RIVER WOODS LN./ BURNSVILLE MN 55337
 55343 ROBERT G. LANGE/ MN 11-2120/ HONEYWELL INC./ 600 2ND ST. NE/ HOPKINS MN 55343/ (612) 542-4925
 55343 ROSS D. SCHMIDT/ MS MN11-2120/ HONEYWELL INC./ 600 2ND STREET NE/ HOPKINS MN 55343/ (612) 542-6741
 55391 GREG KEMNITZ/ 1539 CLARE LANE/ WAYZATA MN 55391/ (612) 473-6123
 55404 JON G. KLASSEN/ 911 22ND AV. SO. #375/ MINNEAPOLIS MN 55404/ (612) 339-4170
 55404 RICK L. MARCUS/ 1609 11TH AVE. S./ MINNEAPOLIS MN 55404/ (612) 339-1638
 55413 BELLE P. SHENOY/ MS MN17-3670/ HONEYWELL INC./ 2600 RIDGEWAY ROAD/ MINNEAPOLIS MN 55413/ (612) 378-5418
 55414 ATTN: KHK/ 330 11TH AVE. S.E./ MINNEAPOLIS MN 55414/ (612) 331-2133
 55414 WALT PERKO/ 727 15TH AVE. S.E./ MINNEAPOLIS MN 55414/ (612) 331-6984
 55418 BOB PETERSON/ 2415 POLK ST. NE/ MINNEAPOLIS MN 55418/ (612) 789-2393
 55420 JOHN ALSTRUP/ INTERDATA CORP./ 10800 LYNDALE AVE. SOUTH - SUITE 130/ BLOOMINGTON MN 55420/ (612) 884-1757
 55427 HUGO MEISSER/ 3021 WISCONSIN AVE. N./ CRYSTAL MN 55427/ (612) 482-3052
 55427 DAVID PERLMAN/ 8309 NORTHWOOD PKWY./ NEW HOPE MN 55427/ (612) 546-2154
 55429 DAVID L. PETERSON/ 6301 UNITY AVE. NO./ BROOKLYN CTR MN 55429
 55435 DANIEL E. GERMANN/ 6813 BROOK DRIVE/ EDINA MN 55435/ (612) 941-1082
 55436 JOHN FITZSIMMONS/ 5025 YVONNE TERRACE/ EDINA MN 55436/ (612) 926-8954
 55437 DENNIS NICKOLAI/ MNAOZA/ CONTROL DATA CORPORATION/ 5001 W. 80TH ST./ BLOOMINGTON MN 55437/ (612) 830-6903
 55455 ATTENTION: BOB JARVIS/ SCH. OF DENTISTRY/CLINICAL SYS. DIV./ 8-440 HEALTH SCIENCE UNIT A/ U OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455
 (612) 376-4131
 55455 KEVIN FJELSTED/ UNIVERSITY COMPUTER CENTER/ 227 EXP ENGR/ U OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-4181
 55455 PAULETTE D. GENES/ UNIVERSITY COMPUTER CENTER/ 227 EXP. ENGR./ UNIV. OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 376-5262
 55455 SARA K. GRAFFUNDER/ UNIVERSITY COMPUTER CENTER/ 227 EXP. ENGR./ U OF MINNESOTA/ MINNEAPOLIS MN 55455/ (612) 376-1637
 55455 THEA D. HODGE/ UNIVERSITY COMPUTER CENTER/ 227 EXP. ENGR./ UNIV OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-4599
 55455 DAN LALIBERTE/ UNIVERSITY COMPUTER CENTER/ 227 EXP. ENGR./ U OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-4181
 55455 MICHAEL PRIETULA/ MGMT. SCIENCES DEPT./ 773 BA/ U OF MINNESOTA/ WEST BANK/ MINNEAPOLIS MN 55455/ (612) 376-7506
 55455 STEVEN A. REISMAN/ UNIVERSITY COMPUTER CENTER/ 227 EX/ UNIV OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 376-1762
 55455 J. B. ROSEN/ 114 LIND HALL/ U OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-0133
 55455 TIMOTHY J SALO/ UNIVERSITY COMPUTER CENTER/ LAUDERDALE/ U OF MINNESOTA/ MINNEAPOLIS MN 55455/ (612) 376-5607
 55455 G. MICHAEL SCHNEIDER/ C.SCI. DEPT./ 114 LIND HALL/ U OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-7582
 56267 ANDY LOPEZ/ COMPUTER CENTER/ U OF MINNESOTA - MORRIS/ MORRIS MN 56267/ (612) 589-1665 X321
 56301 R. WARREN JOHNSON/ DEPT. OF MATH AND COMP. SCI./ MS-149/ ST. CLOUD STATE U/ ST. CLOUD MN 56301/ (612) 255-2147
 56569 PAUL J. WOZNIAK/ R. R. 1/ OGEMA MN 56569/ (612) 376-1137
 57069 JOHN LUSHBOUGH/ COLLEGE OF ARTS & SCIENCES/ UNIV. OF SOUTH DAKOTA/ VERMILLION SD 57069/ (605) 677-5221
 58201 CATHLINE S. HILLEY/ COMPUTER CENTER/ UNIV. OF NORTH DAKOTA/ P.O. BOX 8218 UNIVERSITY STATION/ GRAND FORKS ND 58201/ (701) 777-3171
 60148 ROBERT E. NOVAK/ 21 W 551 NORTH AVE. APT. 123/ LOMBARD IL 60148/ (312) 629-3512
 60181 STEVEN A. VERE/ 1635 S. MICHIGAN AVE. APT. 307/ VILLA PARK IL 60181/ (312) 627-2965
 60201 ARNOLD LAU/ COMPUTER SCIENCE DEPT./ NORTHWESTERN UNIV./ EVANSTON IL 60201/ (312) 463-2694
 60439 ATTENTION: J. M. KNOCK/ BLDG 203-C110/ ARGONNE NATIONAL LABORATORY/ 9700 SOUTH CASS AVENUE/ ARGONNE IL 60439/ (312) 739-7711
 60532 TERRY E. WEYMOUTH/ 4702 BEAU BIEN LANE EAST/ LISLE IL 60532
 60652 TONY CHMIEL/ 3900 WEST 84TH PLACE/ CHICAGO IL 60652
 60657 ROBERT D. GUSTAFSON/ SIMULATION SPECIALISTS INC./ 609 WEST STRATFORD PLACE/ CHICAGO IL 60657
 60660 JON SINGER/ 1540 W. ROSEMONT #3E/ CHICAGO IL 60660/ (312) 262-8545
 61801 ATTENTION: MIKE WILDE - CONSULTING OFF/ COMPUTING SERVICES OFFICE/ 138 DIGITAL COMPUTER LAB/ U OF ILLINOIS/ URBANA IL 61801/ (217) 333-6133
 61801 BOB LIDRAL/ 406 EAST GREEN STREET - APT. 104/ URBANA IL 61801/ (217) 367-5372
 61801 M. D. MICKUNAS/ 297 DCL/ U OF ILLINOIS/ URBANA IL 61801/ (217) 333-6351
 61820 ATTN: L. LAWRIE/ CERL - SOC/ U.S. ARMY/ P.O. BOX 4005/ CHAMPAIGN IL 61820/ (217) 352-6511
 61820 PETER DEWOLF/ 310 W. EUREKA/ CHAMPAIGN IL 61820/ (217) 356-1548 (HOME)/ (217) 333-8252 (WORK)
 62704 MIKE HARRIS/ APT. 4/ 309 WEST EDWARDS/ SPRINGFIELD IL 62704/ (217) 789-7669 (HOME)/ (217) 782-0014 (WORK)

62901 JAMES W. BUTLER/ COMPUTER SERVICES/ WHAM BLDG./ SOUTHERN ILLINOIS UNIV./ CARBONDALE IL 62901/ (618) 453-4361
63188 JOHN K. MCCANDLISS/ ATTN: DRXAL-TF(JOHN K. MCCANDLISS)/ ALMSA/ P.O. BOX 1578/ ST. LOUIS MO 63188/ (314) 268-5361/ (314) 268-5362
64108 LARRY D. LANDIS/ UNITED COMPUTING SYSTEMS/ 2525 WASHINGTON/ KANSAS CITY MO 64108/ (816) 942-6063
64108 JEFFERY M. RAZAFSKY/ UNITED COMPUTING SYSTEMS INC./ 500 W. 26TH STREET/ KANSAS CITY MO 64108/ (816) 221-9700
64108 ROBERT R. TEISBERG/ UNITED COMPUTING SYSTEMS/ 2525 WASHINGTON/ KANSAS CITY MO 64108/ (816) 221-9700 X431
65201 DIANE L. KRAMER/ CAMPUS COMPUTING CENTER/ UNIV. OF MISSOURI-COLUMBIA/ 100 LEFEVRE/ COLUMBIA MO 65201/ (314) 882-6382
65401 HOWARD D. PYRON/ 312 MATH - COMP. SCIENCE/ UNIV OF MISSOURI - ROLLA/ ROLLA MO 65401/ (314) 341-4495
66045 JIM ARNOLD/ COMPUTER SCIENCE DEPT./ 18 STRONG HALL/ KANSAS UNIV./ LAWRENCE KS 66045/ (913) 864-4482
66045 CHARLES J. BANGERT/ ACADEMIC COMPUTER CENTER/ UNIVERSITY OF KANSAS/ P.O. DRAWER 2007/ LAWRENCE KS 66045/ (913) 864-4291
66045 STEVEN S. MUCHNICK/ DEPARTMENT OF COMPUTER SCIENCE/ 18 STRONG HALL/ UNIV OF KANSAS/ LAWRENCE KS 66045/ (913) 864-4482
66045 GREGORY F. WETZEL/ DEPARTMENT OF COMP. SCI./ 18 STRONG HALL/ UNIVERSITY OF KANSAS/ LAWRENCE KS 66045/ (913) 864-4482
66506 ALAN E. SKIDMORE/ COMPUTER SCIENCE DEPT./ KANSAS STATE UNIV./ MANHATTAN KS 66506/ (913) 532-6350
67401 KEARNEY HILL/ 857 NAVAJO/ SALINA KS 67401/ (913) 825-2971
68154 JERRY L. RAY/ 4540 SOUTH 84TH STREET/ OMAHA NE 68154/ (402) 592-3520
70118 DANIEL B. KILLEEN/ COMPUTER LAB/ RICHARDSON BLDG./ TULANE UNIVERSITY/ 6823 ST. CHARLES AVE/ NEW ORLEANS LA 70118/ (504) 865-5631
70504 DAVID LANDSKOV/ UNIV OF SOUTHWESTERN LOUISIANA/ USL BOX 4-4154/ LAFAYETTE LA 70504/ (318) 233-7949
70803 K. P. LEE/ DEPT. OF COMP. SCI./ 102 NICHOLSON/ LOUISIANA STATE UNIVERSITY/ BATON ROUGE LA 70803/ (504) 388-1495
71201 KENNETH R. DUCKWORTH/ FORD BACON AND DAVIS/ 3901 JACKSON STREET/ MONROE LA 71201
72554 DAN REED/ BOX 22/ MAMMOTH SPRING AR 72554
73019 RICHARD V. ANDREE/ MATH DEPT./ UNIV OF OKLAHOMA/ NORMAN OK 73019/ (405) 325-3410
73019 R. A. MORRIS/ MATH DEPT/ UNIV OF OKLAHOMA/ NORMAN OK 73019/ (405) 325-3391
73019 JAMES D. WHITE/ COMPUTING SERVICES/ UNIV. OF OKLAHOMA/ 1610 NEWTON DRIVE/ NORMAN OK 73019/ (405) 325-1882
74102 KENNETH R. DRIESSEL/ AMOCO RESEARCH/ P.O. BOX 591/ TULSA OK 74102
74145 CONRAD SUECHTING/ DATA GENERAL CORP./ 9726 E. 42ND ST. SUITE 200/ TULSA OK 74145
75023 ROGER R. BATE/ 3428 MISSION RIDGE/ PLANO TX 75023/ (214) 238-3052
75062 GRANT COLVIN/ MANAGEMENT SHARES INC./ 2121 WEST AIRPORT FREEWAY - SUITE 660/ IRVING TX 75062/ (214) 255-7121
75080 E. J. SAMMONS/ M/S 406-246/ ROCKWELL INTERNATIONAL/ 1200 N. ALMA/ RICHARDSON TX 75080/ (214) 690-5802
75081 FRANK DUNN/ 1912 E. SPRING VALLEY ROAD/ RICHARDSON TX 75081/ (214) 231-3423
75081 GEORGE LIGLER/ 626 GOODWIN DR./ RICHARDSON TX 75081/ (214) 238-5311
75229 DAVID E. BREEDING/ HARRIS DATA COMM DIV/ 11262 INDIAN TRAIL/ DALLAS TX 75229/ (214) 620-4294
75235 W. J. PERVIN/ REGIONAL COMPUTER CENTER/ UNIV. OF TEXAS-DALLAS/ 5601 MEDICAL CENTER DRIVE/ DALLAS TX 75235/ (214) 688-2383
75275 JANET TAYLOR/ USER SERVICES/ COMPUTER CENTER/ SOUTHERN METHODIST UNIVERSITY/ P.O. BOX 262/ DALLAS TX 75275/ (214) 692-2900
75961 JESSE D. MIXON/ DEPT. OF ACCOUNTING/ STEPHEN F. AUSTIN STATE UNIV/ P.O. BOX 3005 SFA STATION/ NACOGDOCHES TX 75961/ (713) 569-3105/ 569-2508
76012 ROSS F. HOUSHOLDER/ 1725 BROOKS DRIVE/ ARLINGTON TX 76012/ (817) 461-1149
77005 JOHNNIE BUZEK JR./ SOFTWARE RESOURCES/ P.O. BOX 25210/ HOUSTON TX 77005/ (713) 521-0366
77005 THOMAS E. SHIELDS/ SOFTWARE RESOURCES/ 2715 BISSONNET - SUITE 212/ HOUSTON TX 77005/ (713) 521-0366
77027 CHARLES L. HETHCOAT III/ C/O PIPELINE TECHNOLOGISTS INC./ 5251 WESTHEIMER P.O. BOX 22146/ HOUSTON TX 77027/ (713) 622-3456 X334 (WORK)
(713) 626-7737 (HOME)
77043 JOHN EARL CRIDER/ 2918 KEVIN LANE/ HOUSTON TX 77043/ (713) 241-4501
77341 EASTON BEYMER/ SAM HOUSTON STATE UNIV./ P.O. BOX 2821/ HUNTSVILLE TX 77341
77801 CHARLES MATTAIR/ AGENCY RECORDS CONTROL/ P.O. BOX 1009/ BRYAN TX 77801/ (713) 693-6122 X253
77843 UDO POOCH/ INDUSTRIAL ENGINEERING DEPT./ TEXAS A & M UNIV./ COLLEGE STATION TX 77843/ (713) 845-5531
78284 MIKE GREEN/ DATAPOINT CORPORATION/ 9725 DATAPOINT DRIVE/ SAN ANTONIO TX 78284/ (512) 699-7345
78712 TOM KEEL/ COMPUTATION CENTER/ UNIV. OF TEXAS - AUSTIN/ AUSTIN TX 78712/ (512) 471-3242
78731 DAVID W. HOGAN/ 4312 FAR WEST BLVD/ AUSTIN TX 78731/ (512) 258-7837
78758 WILLIAM L. COHAGAN/ SUITE 211/ S/B/P & C ASSOCIATES/ 8705 SHOAL CREEK BLVD./ AUSTIN TX 78758/ (512) 458-2276
79409 ATTN: COMPUTER CENTER/ TEXAS TECH UNIV./ P.O. BOX 4519/ LUBBOCK TX 79409
79409 LEONARD H. WEINER/ DEPT. OF MATH AND COMP. SCI./ TEXAS TECH. UNIV/ P.O. BOX 4319/ LUBBOCK TX 79409/ (806) 742-2571
79601 JOHN TUCKER/ 628 E. N. 16TH ST./ ABILENE TX 79601/ (915) 698-1605
80004 RAYNER K. ROSICH/ 7031 PIERSON ST./ ARVADA CO 80004/ (303) 499-1000 X3109 (WORK)/ (303) 421-0425 (HOME)
80204 GERHARDT C. CLEMENTSON/ DEPT. OF COMP. AND MGMT SCIENCE/ METRO POLITAN STATE COLLEGE/ 1006 11TH STREET BOX 13/ DENVER CO 80204/ (303) 629-3009
80303 P. K. GOVIND/ 850 WEST MOORHEAD CIRCLE - APT. 2-L/ BOULDER CO 80303
80303 R. KEITH NICKEY/ 3580 EVERETT DRIVE/ BOULDER CO 80303/ (303) 494-2847
80309 WILLIAM M. WAITE/ ELECTRICAL ENGINEERING DEPT./ SOFTWARE ENGINEERING GROUP/ UNIVERSITY OF COLORADO/ BOULDER CO 80309/ (303) 492-7204
83316 GEORGE W. ANTHONY/ ANTHONY FARMS/ BOX 632/ BUHL ID 83316/ (208) 326-5703/ (208) 543-5233
83639 JAY WOODS/ BOX 297/ MARSING ID 83639/ (208) 896-4462
84112 ATTN: COMPUTER SCIENCE DEPARTMENT/ 3160 MEB/ UNIV OF UTAH/ SALT LAKE CITY UT 84112/ (801) 581-8224
84112 MIKE LEMON/ COMPUTER SCIENCE DEPT./ 3160 MEB/ UNIVERSITY OF UTAH/ SALT LAKE CITY UT 84112/ (801) 581-8378
84112 GARY LINDSTROM/ COMPUTER SCIENCE DEPT./ UNIV OF UTAH/ SALT LAKE CITY UT 84112/ (801) 581-8224
84112 ED SHARP/ COMPUTER CENTER/ 3116 M.E.B./ U OF UTAH/ SALT LAKE CITY UT 84112/ (801) 581-6575
85001 RICHARD M. WILSON/ BIOMEDICAL ELECTRONICS/ ST. JOSEPH'S HOSPITAL/ P.O. BOX 2071/ PHOENIX AZ 85001/ (602) 277-6611 X3257
85016 ROBERT FULKS/ SUITE 253/ OMNICO MP INC./ 5150 N. 16TH ST./ PHOENIX AZ 85016/ (602) 264-2475
85282 KIRK D. THOMPSON/ 2321 EAST LOYOLA DRIVE/ TEMPE AZ 85282/ (602) 965-3716(WORK)
87106 DAVE PEERCY/ BDM CORP./ 2600 YALE BLVD. SE/ ALBUQUERQUE NM 87106
87106 BOB WALSH/ 817 LAFAYETTE DR. NE/ ALBUQUERQUE NM 87106/ (505) 268-1654
87115 BRUCE LINK/ DIVISION 1712/ SAN DIA LABORATORIES/ ALBUQUERQUE NM 87115/ (505) 264-1281

87115 NANCY RUIZ/ ORG. 5166/ SANDIA LABS/ ALBUQUERQUE NM 87115/ (505) 264-3690
88003 JOSEPH EINWECK/ COMPUTER CENTER/ NEW MEXICO STATE UNIVERSITY/ BOX 3AT/ LAS CRUCES NM 88003/ (505) 646-1443
88047 JOHN TUCKER/ P.O. BOX 2122/ MESILLA PARK NM 88047/ (505) 526-5 544 X64
89154 JOHN WERTH/ DEPT. OF MATH/ UNIV OF NEVADA - LAS VEGAS/ LAS VEGAS NV 89154/ (702) 739-3567
90007 PER BRINCH HANSEN/ COMPUTER SCIENCE DEPT./ UNIV. OF SOUTHERN CALIFORNIA/ UNIVERSITY PARK/ LOS ANGELES CA 90007/ (213) 741-5501
90007 JORGEN STAUNSTRUP/ COMPUTER SCIENCE DEPT./ UNIV. OF SOUTHERN CALIFORNIA/ UNIVERSITY PARK/ LOS ANGELES CA 90007/ (213) 741-5501
90020 KENNETH YOUNG/ 3311 WEST 3RD ST. APT. 1-319/ LOS ANGELES CA 90 020/ (213) 383-9666
90036 WILLIAM MOSKOWITZ/ INSTRUCTIONAL SUPPORT GROUP/ CALIFORNIA STATE UNIVERSITY/ 5670 WILSHIRE BOULEVARD/ LOS ANGELES CA 90036/ (213) 852-5789
90066 LEROY E. NELSON/ 11925 AVON WAY/ LOS ANGELES CA 90066/ (213) 3 97-7390
90250 CHARLES SISK JR./ TRW DATA SYSTEMS/ 12911 SIMMS AVENUE/ HAWTHORNE CA 90250/ (213) 535-3777
90266 HERBERT E. MORRISON/ 1257 2ND STREET/ MANHATTAN BEACH CA 90266
90274 JIM HIGHTOWER/ 4947 BROWNDEER LANE/ RANCHO PALOS VERDES CA 90274/ (213) 541-4662
90278 JAMES L. AGIN/ 2178 BLD. 90/ TRW-DSSG/ ONE SPACE PARK/ REDONDO BEACH CA 90278/ (213) 535-0313
90278 JOHN R. DEALY/ BLDG. R3/1072/ TRW DSSG/ ONE SPACE PARK/ REDONDO BEACH CA 90278/ (213) 535-0833
90278 WILEY GREINER/ 90/2178/ TRW DSSG/ ONE SPACE PARK/ REDONDO BEACH CA 90278/ (213) 535-0313
90278 ROGER A. VOSSLER/ BLDG. 90-2178/ TRW/DSSG/ ONE SPACE PARK/ REDONDO BEACH CA 90278/ (213) 535-0312
90402 WILLIAM S. COOKE/ 503 22ND STREET/ SANTA MONICA CA 90402/ (213) 451-2615
90403 MICHAEL TEENER/ TECHNOLOGY SERVICE CORP./ 2811 WILSHIRE BLVD./ SANTA MONICA CA 90403/ (213) 829-7411 X244
90405 TERRY J. LAYMAN/ 2039 4TH ST. #106/ SANTA MONICA CA 90405/ (213) 357-2121 X279
90406 ATTN: ELAINE DENTON (41-41)/ SOFTWARE INFORMATION SERVICES/ SYSTEM DEVELOPMENT CORP./ 2500 COLORADO AVE./ SANTA MONICA CA 90406
90501 WILLIAM E. FISHER/ 2074 SANTA FE AVENUE/ TORRANCE CA 90501/ (213) 328-7193
90733 PAUL RUSSELL/ LOGICON INC./ 255 WEST 5TH ST. P.O. BOX 471/ SAN PEDRO CA 90733/ (213) 831-0611
90801 DAVE SKINNER/ COMMUNICATION MFG. COMPANY/ P.O. BOX 2708/ LONG BEACH CA 90801/ (213) 426-8345
90803 STEVEN L. BRECHER/ 5235 MARINA PACIFICA DR. NORTH KEY 19/ LONG BEACH CA 90803
91101 GURUPREM SINGH KHALSA/ BYTE SHOP OF PASADENA/ 496 SOUTH LAKE AVENUE/ PASADENA CA 91101/ (213) 684-3311
91101 E. E. SIMMONS/ 455 SOUTH OAKLAND AVE/ PASADENA CA 91101/ (213) 687-7047
91104 MILAN KARSPECK/ 1149 NORTH MICHIGAN/ PASADENA CA 91104
91126 LARRY SEILER/ 1-55 RUDDOCK/ CALIFORNIA INST. OF TECHNOLOGY/ PASADENA CA 91126/ (213) 449-9886
91203 RICHARD DIEVENDORFF/ DEPARTMENT 84F/ IBM/ 620 NORTH BRAND BLVD./ GLENDALE CA 91203
91307 RHODA P. NOVAK/ 6736 RANDIWOOD LANE/ CANOGA PARK CA 91307/ (213) 346-1135
91330 JOHN M. MOTIL/ DEPT. OF COMPUTER SCIENCE/ CALIFORNIA STATE UNIVERSITY/ NORTHRIDGE CA 91330/ (213) 885-2193
91342 STERLING WILSON/ 11621 RINCON AVE./ SYLMAR CA 91342
91360 PAULO S. CASTILLO JR./ 385 QUEENSBURY ST./ THOUSAND OAKS CA 91360/ (213) 670-1515 X3100
91367 GEORGE MASSAR/ 6225-102 SHOUP AVE./ WOODLAND HILLS CA 91367/ (213) 346-1883/ (213) 970-5021 (NORTHROP)
91405 L. F. MELLINGER/ 13622 HART ST./ VAN NUYS CA 91405/ (213) 354-2505
91601 LARRY ROBERTSON/ 5651 CASE AVE./ N. HOLLYWOOD CA 91601/ (213) 762-8068
91604 JERRY POURNELLE/ 12051 LAUREL TERRACE DRIVE/ STUDIO CITY CA 91604/ (213) 762-2256
91711 STANLEY E. LUNDE/ 890 HOOD DRIVE/ CLAREMONT CA 91711/ (714) 626-9977
91775 TOM GREER/ 224 N. ALABAMA ST./ SAN GABRIEL CA 91775/ (213) 286-8226
92014 JOEL MCCORMACK/ 507 CAMINO DEL MAR/ DEL MAR CA 92014/ (714) 755-8135
92093 KEN BOWLES/ APIS DEPT./ C-021/ U OF CALIFORNIA - SAN DIEGO/ LA JOLLA CA 92093/ (714) 755-7288/ 452-4526
92093 BOB HOFKIN/ APIS DEPT. C-014/ UNIV. OF CALIFORNIA-SAN DIEGO/ LA JOLLA CA 92093
92110 BILL VELMAN/ SCHOOL OF LAW/ UNIV. OF SAN DIEGO/ ALCALA PARK/ SAN DIEGO CA 92110/ (714) 291-6480
92127 WALT FEESER/ MS 401/ BURROUGHS CORP./ 16701 W. BERNARDO DR./ SAN DIEGO CA 92127
92152 MICHAEL S. BALL/ CODE 632/ NAVAL OCEAN SYSTEMS CENTER/ SAN DIEGO CA 92152/ (714) 225-2365
92324 DAVID H. WELCH/ P.O. BOX 721/ COLTON CA 92324
92335 WALLY SCHNITGER/ C/O INSTRUMENT RESEARCH CO./ P.O. BOX 666/ FONTANA CA 92335/ (714) 546-4474
92507 JOHN DE PILLIS/ 2931 WALDORF DRIVE/ RIVERSIDE CA 92507/ (714) 686-0534 (HOME)/ (714) 787-5002 (WORK)
92626 ATTN: A. S. WILLIAMS - LIBRARIAN/ LIBRARY/ TECHNOLOGY MARKETIN G INC./ 3170 RED HILL AVE./ COSTA MESA CA 92626/ (714) 979-1100
92627 TIM LOWERY/ 2653 SANTA ANA AVE./ COSTA MESA CA 92627
92634 WALTER R. RYPER/ BLDG 604 M/S E212/ HUGHES - GSG/ P.O. BOX 3310/ FULLERTON CA 92634/ (714) 871-3232 X3318
92634 SEYMOUR SINGER/ BLDG 606/M.S. K110/ HUGHES AIRCRAFT CO./ P.O. BOX 3310/ FULLERTON CA 92634/ (714) 871-3232 X1167
92644 MARK JUNGWIRTH/ 13318 NEWLAND ST/ GARDEN GROVE CA 92644
92646 BARCLAY R. KNERR/ 9061 CHRISTINE DRIVE/ HUNTINGTON BEACH CA 92646
92669 TED SHAPIN/ 5110 E. ELSINORE AV./ ORANGE CA 92669/ (714) 633-0922
92680 ROBERT W. ANDERSON/ 345 W. FIRST APT 38/ TUSTIN CA 92680
92680 GARY DUNCAN/ P.O. BOX 930/ TUSTIN CA 92680
92704 DAVID C. FITZGERALD/ CONTROL DATA CORP./ 3519 W. WARNER AVE./ SANTA ANA CA 92704/ (714) 754-4244
92704 JIM FONTANA/ CONTROL DATA CORPORATION/ 3519 W. WARNER AVE./ SANTA ANA CA 92704/ (714) 754-4244
92705 MARK M. SCHNEGG/ PERTEC COMPUTER CORP./ 17112 ARMSTRONG AVE./ IRVINE CA 92705/ (714) 540-8340
92713 RUDY L. FOLDEN/ OPERATING SYSTEMS DEVELOPMENT/ P.O. BOX C-1950 4/ SPERRY UNIVAC/ 2722 MICHELSON DRIVE/ IRVINE CA 92713/ (714) 833-2400
92713 BOB HUTCHINS/ COMPUTER AUTOMATION INC./ 18651 VON KARMAN/ IRVINE CA 92713/ (714) 833-8830 X335
92714 ALEX BRADLEY/ STANDARD SOFTWARE SYSTEMS/ 17931 'J' SKY PARK/ IRVINE CA 92714
92714 WILLIAM E. DROBISH/ ADVANCED DEVELOPMENT/ SILICON SYSTEMS INC. / 16692 HALE AVENUE/ IRVINE CA 92714/ (714) 979-0941
92714 PAUL KELLY/ EDUCATIONAL DATA SYSTEMS/ 1682 LANGLEY AVENUE/ IRVINE CA 92714/ (714) 556-4242
92714 JOHN S. CONERY/ PERTEC COMPUTER CORP./ 17112 ARMSTRONG AVE./ SANTA ANA CA 92714/ (714) 540-8340

92715 WILLIAM J. EARL/ 10 BANYAN TREE LANE/ IRVINE CA 92715/ (714) 552-1543
 92803 C. L. HORNEY/ MICROELECTRONIC DEVICE DIV./ D/832-RC27/ ROCKWELL INTERNATIONAL/ P.O. BOX 3669/ ANAHEIM CA 92803
 92807 DAN MARCUS/ GTE INFORMATION SYSTEMS/ 5300 E. LA PALMA AVE./ ANAHEIM CA 92807/ (714) 524-3131
 93017 BILL DODSON/ BURROUGHS CORP./ 6300 HOLLISTER/ GOLETA CA 93017/ (805) 964-6881
 93105 DENNIS R. AUSTIN/ 26 WEST JUNIPERO STREET/ SANTA BARBARA CA 93105/ (805) 962-7320
 93111 ANDY HARRINGTON/ 72 SOUTH PATTERSON #207/ SANTA BARBARA CA 93111/ (805) 964-6881 (WORK)/ (805) 967-4235 (HOME)
 93407 NEIL W. WEBRE/ DEPT. OF COMP. SCI. AND STAT./ CALIF. POLY. STATE UNIV./ SAN LUIS OBISPO CA 93407/ (805) 546-2986
 94010 D. B. ANDERSON/ 280 BELLA VISTA DRIVE/ HILLSBOROUGH CA 94010
 94022 ATTN: NEWBERRY MICROSYSTEMS/ 24225 SUMMERHILL AVE/ LOS ALTOS CA 94022/ (415) 948-8007
 94025 BOB ALBRECHT/ DYMEX/ P.O. BOX 310/ MENLO PARK CA 94025/ (415) 323-6117
 94035 CARL S. ROSENBERG/ AMES RESEARCH CENTER/ MAIL STOP 239-19/ MOFFETT FIELD CA 94035/ (415) 965-6436 (WORK)/ (415) 967-7000 (HOME)
 94086 GEORGE LEWIS/ R & D/ BASIC TIMESHARING INC./ 870 WEST MAUDE AVENUE/ SUNNYVALE CA 94086/ (408) 733-1122
 94086 FLEMING M. OLIVER/ 213 WEDDELL - APT. 12/ SUNNYVALE CA 94086/ (408) 734-8771
 94086 GERALD STEINBACK/ SYSTEMS PROGRAMMING/ SIGNETICS CORP./ 811 EA ST ARQUES AVE./ SUNNYVALE CA 94086/ (408) 739-7700 X2055
 94086 ANDREW HARRIS ZIMMERMAN/ 550 NORTH FAIR OAKS AVE. APT. 14/ SUNNYVALE CA 94086/ (408) 732-8109
 94088 ROY E. BOLLINGER/ DEPT. 1965/ BLD 529/ LOCKHEED/ P.O. BOX 504/ SUNNYVALE CA 94088/ (408) 742-3182
 94109 ERIC SMALL/ ERIC SMALL AND ASSOCIATES/ 680 BEACH STREET/ SAN FRANCISCO CA 94109/ (415) 441-0666
 94112 TURNEY C. STEWARD/ 201 DRAKE ST./ SAN FRANCISCO CA 94112
 94301 JAN DEDEK/ 505 LOWELL AVE./ PALO ALTO CA 94301/ (415) 321-9298
 94303 C. E. DUNCAN/ 865 THORNWOOD DRIVE/ PALO ALTO CA 94303
 94304 LINDA E. CROLEY/ BNR INC./ 3174 PORTER DR./ PALO ALTO CA 94304 / (415) 494-3942 X40 OR 61
 94304 ROBERT A. FRALEY/ HEWLETT PACKARD LABS/ 3500 DEER CREEK RD./ PALO ALTO CA 94304/ (415) 494-1444
 94306 DAVID ELLIOT SHAW/ SUITE 605/ STRUCTURED SYSTEMS CORPORATION/ 2600 EL CAMINO REAL/ PALO ALTO CA 94306/ (415) 321-8111
 94404 BRUCE BARRETT/ 930 LIDO LANE/ FOSTER CITY CA 94404/ (415) 349-8724
 94521 PAUL GODFREY/ 5545 MARYLAND DR./ CONCORD CA 94521
 94546 DAVID P. FRICK/ 5964 HIGHWOOD RD./ CASTRO VALLEY CA 94546/ (415) 537-9536
 94550 S. T. HEIDELBERG/ DIVISION 8323/ SANDIA LABORATORIES/ LIVERMORE CA 94550/ (415) 455-2179
 94563 DAVID CLINGERMAN/ 63 LOST VALLEY DRIVE/ ORINDA CA 94563/ (415) 834-3030
 94611 ROBERT C. NICKERSON/ 6966 COLTON BLVD/ OAKLAND CA 94611
 94701 JOSEPH N. JOHNSON/ P.O. BOX 92/ BERKELEY CA 94701
 94701 JOHN P. MCGINITIE/ P.O. BOX 655/ BERKELEY CA 94701
 94702 TIMOTHY DAVID MCCREERY/ P.O. BOX 2423/ BERKELEY CA 94702/ (415) 527-2774
 94704 MARK ZIMMER/ #10 2750 DWIGHT AVE/ BERKELEY CA 94704
 94707 DAVE BAASCH/ 1011 FOUNTAIN WALK/ BERKELEY CA 94707/ (415) 525-3458
 94708 MAURICE MCEVOY/ 1212 QUEEN'S ROAD/ BERKELEY CA 94708/ (415) 843-8260
 94709 PETER LINHARDT/ 1890 ARCH STREET - APT. 202/ BERKELEY CA 94709 / (415) 841-6917
 94720 ATTN: LIBRARY/ LAWRENCE BERKELEY LAB/ 134 BLDG 50/ UNIV OF CALIFORNIA/ BERKELEY CA 94720
 94720 WILLETT KEMPTON/ LANGUAGE BEHAVIOR RESEARCH LAB/ UNIVERSITY OF CALIFORNIA/ 2220 PIEDMONT AVE./ BERKELEY CA 94720
 94801 HARRY R. CHESLEY/ 400 WEST RICHMOND/ POINT RICHMOND CA 94801/ (415) 233-8220
 94804 CARROLL E. BUTTERFIELD/ ELECTRONICS DIVISION/ BADGER METER INC. / 150 EAST STANDARD AVENUE/ RICHMOND CA 94804/ (415) 233-8220
 94901 ERNEST W. JONES/ 59 BILLOU STREET/ SAN RAFAEL CA 94901
 95005 WARREN VAN CAMP/ 485 PARK DR./ BEN LOMOND CA 95005/ (408) 336-5654
 95014 ATTN: ENGINEERING LIBRARY/ FOUR-PHASE SYSTEMS INC./ 10700 NORTH DE ANZA BLVD./ CUPERTINO CA 95014/ (408) 255-0900 X2694
 95014 ARDON R. CORD/ DATREX CORP./ 21050 MCCLELLAN ROAD - SUITE 1/ CUPERTINO CA 95014/ (408) 996-2551
 95014 BOB PUETTE/ DATA SYSTEMS DIVISION/ HEWLETT-PACKARD CO./ 11000 WOLFE ROAD/ CUPERTINO CA 95014/ (408) 257-7000
 95030 STAN HEAD/ 19270 RAINIERI LANE/ LOS GATOS CA 95030/ (408) 353-3055
 95030 TERRY THOMAS/ 24740 MILLER HILL RD./ LOS GATOS CA 95030/ (415) 965-6436
 95035 ROB MEANS/ 1421 YELLOWSTONE/ MILPITAS CA 95035/ (408) 262-0420
 95050 JOHN DENNIS COUCH/ GSD/ HEWLETT-PACKARD/ 5303 STEVENS CREEK BLVD./ SANTA CLARA CA 95050/ (408) 249-7020 X2949
 95050 E. HAROLD WILLIAMS/ SYSCOM/ 3058-B SCOTT BLVD./ SANTA CLARA CA 95050/ (408) 246-2437
 95051 JOHN DOERR/ INTEL CORPORATION MCD/ 3065 BOWERS AVENUE/ SANTA CLARA CA 95051/ (408) 987-7351
 95051 R. STEVEN GLANVILLE/ 100 BUCKINGHAM DR. APT 238/ SANTA CLARA CA 95051/ (408) 241-6294
 95064 MICHAEL FAY/ INFORMATION SCI. DEPT./ UNIV. OF CALIFORNIA - SANTA CRUZ/ SANTA CRUZ CA 95064/ (408) 429-4043
 95120 THOMAS A. ROLANDER/ 21950 MCKEAN ROAD/ SAN JOSE CA 95120/ (408) 378-2014
 95131 D. H. SPRINGER/ COMPUTER SYSTEMS DIVISION/ ANDERSON JACOBSON INC./ 521 CHARCOT AVENUE/ SAN JOSE CA 95131/ (408) 263-8520
 95133 JOHN H. SPANTON/ 2351 RAVINE DRIVE/ SAN JOSE CA 95133/ (408) 734-7145
 95153 TOM PITTMAN/ ITTY BITTY COMPUTERS/ P.O. BOX 23189/ SAN JOSE CA 95153
 95404 GARY LOWELL/ 2625 HIDDEN VALLEY/ SANTA ROSA CA 95404/ (707) 544-6373
 95521 ATTN: DIRECTOR / INST. RESEARCH/ ADP/ HUMBOLDT STATE UNIVERSITY/ ARCATA CA 95521
 97077 ATTN: MANUFACTURING COMPUTER GROUP/ DS 60-171/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 638-3411 X2710
 97077 TERRY HAMM/ M.S. 60-456/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 638-3411 X2579
 97077 RANDY HODNETT/ MS 39-007/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 644-0161
 97077 GREG JOHNS/ MS 39-007/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 644-0161
 97077 LYNN SAUNDERS/ MS 39-135/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 644-0161 X6640
 97077 WAYNE VYROSTEK/ MS 74-329/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 644-0161
 97201 ROBERT D. PERRY JR./ 255 SW HARRISON ST. APT. 5-D/ PORTLAND OR 97201/ (503) 222-6654

97213 BRIAN HANSEN/ 2426 N.E. 57TH AVE. APT #3/ PORTLAND OR 97213/ (503) 284-3537
 97221 ATTN: OREGON MINI-COMPUTER SOFTWARE IN/ 4015 SW CANYON ROAD/ PORTLAND OR 97221/ (503) 226-7760
 97229 DAVID ROWLAND/ ELECTRO SCIENTIFIC INDUSTRIES/ 13900 N.W. SCIENCE PARK DRIVE/ PORTLAND OR 97229/ (503) 641-4141
 97330 DAVID F. CAUTLEY/ DEPT. OF COMPUTER SCIENCE/ GENERAL INFORMATION SYSTEMS INC./ 983 N.W. SPRUCE ST./ CORVALLIS OR 97330/ (503) 754-1171
 97330 GARY OLIVER/ P.O. BOX 826/ CORVALLIS OR 97330/ (503) 753-1770
 97401 H. MARC LEWIS/ REGIONAL INFORMATION SYSTEMS/ 125 E 8TH/ EUGENE OR 97401/ (503) 687-4559
 97440 KENT LOOBY/ FARWEST STEEL CORP./ P.O. BOX 889/ EUGENE OR 97440/ (503) 686-2000
 98008 KEITH MITCHELL/ 16213 SE 28 PL/ BELLEVUE WA 98008
 98033 KENNETH E. CHARLTON/ 12929 11TH PLACE N.E./ KIRKLAND WA 98033 / (206) 822-9348
 98040 DONNAFAYE FINGER/ 4108 78TH AVE SE/ MERCER ISLAND WA 98040/ (206) 232-2428
 98043 GARY S. ANDERSON/ JOHN FLUKE MFG. CO. INC./ P.O. BOX 43210 - M.S. 29/ MOUNTLAKE TERR WA 98043/ (206) 774-2296
 98109 PETER A. ARMSTRONG/ DIGITAL DATA SYSTEMS/ 1113 DEXTER AVE. N./ SEATTLE WA 98109/ (206) 282-2323
 98195 HELLMUT GOLDE/ DEPT. OF COMP. SCI./ FR-35/ U OF WASHINGTON/ SEATTLE WA 98195/ (206) 543-9264
 98225 ROBERT B. FINCH/ 910 N. LAKE SAMISH DR.-APT. 30/ BELLINGHAM WA 98225/ (206) 734-0781
 98401 ATTN: ENGINEERING LIBRARY/ PT 1-14/ WEYERHAUSER CO./ TACOMA WA 98401/ (206) 572-9000 X495
 98632 BUZZ HILL/ EYEDENTIFY INC./ P.O. BOX 2006/ LONGVIEW WA 98632/ (206) 423-3281
 98926 F. STANLEY/ COMPUTER SERVICES/ CENTRAL WASHINGTON UNIV./ ELLENBURG WA 98926
 99210 FRED J. MILLER/ BUSINESS COMPUTER SYSTEMS/ DATACOMP/ P.O. BOX 1087/ SPOKANE WA 99210/ (509) 456-6908
 99258 HOUSTON P. LOWRY/ GONZAGA UNIVERSITY/ STUDENT BOX 816/ SPOKANE WA 99258
 2006 AUSTRALIA TONY J. GERBER/ BASSER DEPT. OF COMPUTER SCIENCE/ UNIVERSITY OF SYDNEY/ SYDNEY N.S.W. 2006/ AUSTRALIA/ 692 3216
 2006 AUSTRALIA CARROLL MORGAN/ BASSER DEPT. OF COMPUTER SCIENCE/ UNIV. OF SYDNEY/ SYDNEY N.S.W. 2006/ AUSTRALIA/ 692 3216
 2006 AUSTRALIA BRIAN G. ROWSWELL/ UNIVERSITY COMPUTING CENTRE H08/ UNIVERSITY OF SYDNEY/ SYDNEY N.S.W. 2006/ AUSTRALIA/ (02) 692-3491
 2308 AUSTRALIA ATTN: SERIALS DEPARTMENT/ AUCHMUTY LIBRARY/ UNIVERSITY OF NEWCASTLE/ NEWCASTLE N.S.W. 2308/ AUSTRALIA/ 685391
 2308 AUSTRALIA J. A. CAMPBELL/ MATHEMATICS DEPT./ UNIVERSITY OF NEWCASTLE/ NEWCASTLE N.S.W. 2308/ AUSTRALIA/ 685 657
 2308 AUSTRALIA P. SIMON/ DEPT OF MATHEMATICS/ UNIV. OF NEWCASTLE/ NEWCASTLE N.S.W. 2308/ AUSTRALIA
 2600 AUSTRALIA N. D. H. HAMMOND/ FLEET MAINTENANCE/ NAVY OFFICE OF UNDERWATER WEAPONS/ CANBERRA A.C.T. 2600/ AUSTRALIA
 3000 AUSTRALIA Q. VAN ABBE/ COMPUTER CENTRE/ ROYAL MELBOURNE INSTITUTE OF TECHNOLOGY/ 124 LATROBE STREET/ MELBOURNE VICTORIA 3000/ AUSTRALIA/ (03) 341-2467
 (03) 341-2292
 3052 AUSTRALIA ATTN: LIBRARIAN/ SCHOOL OF MATHEMATICAL SCIENCES/ RICHARD BERRY BUILDING/ UNIVERSITY OF MELBOURNE/ PARKVILLE VICTORIA 3052/ AUSTRALIA
 3052 AUSTRALIA PETER RICHARDSON/ COMPUTER SCIENCE DEPT./ UNIV. OF MELBOURNE/ PARKVILLE VICTORIA 3052/ AUSTRALIA/ (03) 3415225
 3072 AUSTRALIA M. RAHILLY/ 2 RITA STREET/ EAST PRESTON VICTORIA 3072/ AUSTRALIA
 3165 AUSTRALIA GEOFFREY A. CLEAVE/ 18 NEILL COURT/ E. BENTLEIGH VICTORIA 3165/ AUSTRALIA
 4001 AUSTRALIA W. J. G. FISHER/ QUEENSLAND INST. OF TECHNOLOGY/ G.P.O. BOX 24 34/ BRISBANE QUEENSLAND 4001/ AUSTRALIA/ 221-2411 X423
 4067 AUSTRALIA D. B. JOHNSTON/ DEPT. OF COMPUTER SCIENCE/ UNIV. OF QUEENSLAND/ ST. LUCIA QUEENSLAND 4067/ AUSTRALIA/ 07/3706930
 5001 AUSTRALIA B. KIDMAN/ DEPT OF COMPUTER SCIENCE/ UNIVERSITY OF ADELAIDE/ GPO BOX 498/ ADELAIDE S.A. 5001/ AUSTRALIA/ 223 4333
 5063 AUSTRALIA I. N. BLAVINS/ 40 WOODFIELD AVENUE/ FULLARTON S.A. 5063/ AUSTRALIA
 5098 AUSTRALIA CHRIS A. RUSBRIDGE/ SAENET/ SOUTH AUSTRALIA INSTITUTE OF TECHNOLOGY/ P.O. BOX 1/ INGLE FARM S.A. 5098/ AUSTRALIA/ AUSTRALIA 08-260-2055
 6102 AUSTRALIA D. P. HODGSON/ DEPT. OF MATHEMATICS AND COMPUTER STUD/ WESTERN AUSTRALIAN INSTITUTE OF TECHNOLOGY/ HAYMAN ROAD/ SOUTH BENTLEY W.A. 6102/ AUSTRALIA
 7001 AUSTRALIA A. H. J. SALE/ DEPT. OF INFORMATION SCIENCE/ UNIVERSITY OF TASMANIA/ BOX 252C/ HOBART TASMANIA 7001/ AUSTRALIA/ (002) 23 0561
 A-1040 AUSTRIA ATTN: INST. FUER INFORMATIONSSYSTEME/ TECHNISCHE UNIVERSITAT WIEN/ WIEN A-1040/ AUSTRIA
 A-1150 AUSTRIA KONRAD MAYER/ REICHSAPFELG 13/8/ VIENNA A-1150/ AUSTRIA/ (0225 4) 201 781
 B-1050 BELGIUM O. BEAUFAYS/ MATHEMATIQUES APPLIQUEES/ C P I 165/ UNIVERSITE LIBRE DE BRUXELLES/ AVENUE F.-D. ROOSEVELT 50/ BRUXELLES B-1050/ BELGIUM
 B-1170 BELGIUM ALAIN PIROTTE/ MBL/RESEARCH LABORATORY/ AVENUE EM. VAN BECELAERE 2/ BRUSSELS B-1170/ BELGIUM/ 673.41.90/ 673.41.99
 B-1180 BELGIUM MARTINE DE GERLACHE/ CENTRE DE CALCUL/ INSTITUTE ROYAL METEOROLOGIQUE/ 3 AVENUE CIRCULAIRE/ BRUXELLES B-1180/ BELGIUM
 H3C 3J7 CANADA PATRICK WARD/ CENTRE DE CALCUL/ UNIVERSITE DE MONTREAL/ C.P. 6128/ MONTREAL QUEBEC H3C 3J7/ CANADA/ (514) 343-6866
 H3S 1J6 CANADA FRANCOIS PINARD/ 2780 BARCLAY - APARTEMENT 1/ MONTREAL QUEBEC H3S 1J6/ CANADA/ (514) 342-3450
 K1S 5B6 CANADA DAVID A. THOMAS/ COMPUTING SERVICES/ 401 ADMIN. BLDG./ CARLETON UNIV./ OTTAWA ONTARIO K1S 5B6/ CANADA/ (613) 231-6770
 K2A 1T2 CANADA F. S. WINTERSPRING/ P.O. BOX 6115 STATION J/ OTTAWA ONTARIO K2A 1T2/ CANADA
 L4W 1T0 CANADA CARLO LOCICERO/ 3501 GLEN ERIN DRIVE #401/ HISSISSAUGA ONTARIO L4W 1T0/ CANADA/ (416) 826-8640
 L5N 1K7 CANADA PETER HAYNES/ CONTROL DATA CANADA LTD./ 1855 MINNESOTA COURT-STREETSVILLE/ MISSISSAUGA ONTARIO L5N 1K7/ CANADA/ (416) 826-8640 X238
 L5N 1K7 CANADA DAVID JONES/ CONTROL DATA CANADA LTD./ 1855 MINNESOTA COURT-STREETSVILLE/ MISSISSAUGA ONTARIO L5N 1K7/ CANADA/ (416) 826-8640 X262
 L5N 1K7 CANADA HENRY MCGILTON/ CONTROL DATA CANADA/ 1855 MINNESOTA COURT/ MISSISSAUGA ONTARIO L5N 1K7/ CANADA/ (416) 826-8640 X295
 M4B 2E5 CANADA MARK GREEN/ #1001 - 390 DAWES ROAD/ TORONTO ONTARIO M4B 2E5/ CANADA/ (416) 755-8607
 M4N 1A4 CANADA NORMAN A. JULL/ 56A BLYTHWOOD ROAD/ TORONTO ONTARIO M4N 1A4/ CANADA
 M4Y 1P9 CANADA RON BAECCKER/ HUMAN COMPUTING RESOURCES/ 10 ST. MARY STREET SUITE 401/ TORONTO ONTARIO M4Y 1P9/ CANADA/ (416) 922-1937
 M5P 2C3 CANADA ROBERIO DIAS/ 134 COLIN AVE./ TORONTO ONTARIO M5P 2C3/ CANADA
 M5S 1A4 CANADA LEONARD VANEK/ COMPUTER SYSTEMS RESEARCH GROUP/ SANFORD FLEMING BLDG/ UNIV. OF TORONTO/ TORONTO ONTARIO M5S 1A4/ CANADA/ (416) 978-6219
 M9W 5R8 CANADA RON MCKERRON/ COMSHARE LTD./ 230 GALAXY ROAD/ REDDALE ONTARIO M9W 5R8/ CANADA
 N2J 4H2 CANADA CHARLES H. FORSYTH/ APT. 2-304/ 300 REGINA ST. N./ WATERLOO ONTARIO N2J 4H2/ CANADA/ (519) 884-7531/ (519) 885-1211 X3055
 N2L 3G1 CANADA JOHN C. BEATTY/ DEPT. OF COMP. SCIENCE/ UNIV OF WATERLOO/ WATERLOO ONTARIO N2L 3G1/ CANADA/ (519) 885-1211 X2241
 N2L 3G1 CANADA W. MORVEN GENTLEMAN/ MATHEMATICS COMPUTING FACILITY/ UNIVERSITY OF WATERLOO/ WATERLOO ONTARIO N2L 3G1/ CANADA/ (519) 578-8866/ (519) 885-1211
 N6A 4N5 CANADA R. A. ALLAN/ DIESEL DIVISION/ GENERAL MOTORS OF CANADA LTD./ P.O. BOX 5160/ LONDON ONTARIO N6A 4N5/ CANADA
 QUEBEC CANADA DANIEL THALMANN/ DEPARTEMENT D'INFORMATIQUE ET RECHERCHE/ UNIVERSITE DE MONTREAL/ CASE POSTALE 6128 - SUCC A/ MONTREAL H3C 3J7 QUEBEC/ CANADA
 (514) 343-7477
 V6T 1W5 CANADA BARY W. POLLACK/ DEPT OF COMP. SCI./ UNIV. OF BRITISH COLUMBIA/ 2075 WESBROOK PLACE/ VANCOUVER B.C. V6T 1W5/ CANADA/ (604) 228-6794
 V6X 2Z9 CANADA I. GANAPATHY/ NOOTKA BUILDING/ MACDONALD DETTWILER & ASSOCIATES/ 10280 SHELLBRIDGE WAY/ RICHMOND B.C. V6X 2Z9/ CANADA/ (604) 278-3411 X32

DK-1051 DENMARK MOGENS LINDHARD/ NYHAVN 20/ KOBENHAVN K DK-1051/ DENMARK/ (01) 11 12 04
DK-2200 DENMARK ATTN: BIBLIOTEKET/ DATALOGISK INSTITUT./ KOBENHAVN UNIVERSITET/ SIGURDSGADE 41/ KOBENHAVN N DK-2200/ DENMARK
DK-2650 DENMARK NIELS WINTHER/ REBAEK SOPARK 5-544/ HVIDOVRE DK-2650/ DENMARK
DK-2800 DENMARK GUNNAR JOHANSEN/ DEPT. OF CHEMISTRY AND CHEM. ENG./ DANISH ENGINEERING ACADEMY/ BYGNING 375/ LYNGBY DK-2800/ DENMARK
DK-9220 DENMARK UFFE MOLLER/ DATANOMUDDANNELSEN/ LANGAGERVEJ 16/ AALBORG OST DK-9220/ DENMARK/ (08) 15 81 00
SF-00100 FINLAND ATTN: DEPT. OF COMPUTER SCIENCE/ UNIVERSITY OF HELSINKI/ TOOLONKATU 11/ HELSINKI 10 SF-00100/ FINLAND
SF-20500 FINLAND MARKKU SUNI/ COMPUTER CENTRE/ UNIVERSITY OF TURKU/ TURKU 50 SF-20500/ FINLAND/ 921-335599 X280
SF-33101 FINLAND MATTI KARINEN/ COMPUTING CENTRE/LPR-PROJECT/ TAMPERE UNIV. OF TECHNOLOGY/ PL 527/ TAMPERE 10 SF-33101/ FINLAND/ 931-652802(HOME)/ 931-162125 (WORK)
SF-33101 FINLAND JUUKA KESO/ COMPUTING CENTRE/LPR-PROJECT/ TAMPERE UNIV. OF TECHNOLOGY/ PL 527/ TAMPERE 10 SF-33101/ FINLAND/ 931-33727 (HOME)/ 931-162125 (WORK)
SF-33101 FINLAND JYRKI TUOMI/ COMPUTING CENTRE/LPR-PROJECT/ TAMPERE UNIV. OF TECHNOLOGY/ PL 527/ TAMPERE 10 SF-33101/ FINLAND/ 931-50000/570 (HOME)/ 931-162125 (WORK)
SF-33200 FINLAND JORMA SINNAMO/ PYYNIKINT. 3/ TAMPERE 20 SF-33200/ FINLAND
SF-33340 FINLAND VEIKKO VISALA/ KUOKKIJANTIE 10/ TAMPERE 34 SF-33340/ FINLAND/ 917-712338
SF-33410 FINLAND TIMO HAMMAR/ JANISLAHDENKATU 3 A 7/ TAMPERE 41 SF-33410/ FINLAND
SF-33720 FINLAND ERKKI LEHTIMAKI/ OPISKELIJANK. 4A 20/ TAMPERE 72 SF-33720/ FINLAND/ 963-37141 (HOME)
F-31077 FRANCE MICHEL GALINIER/ INFORMATIQUE/ UNIVERSITE P. SABATIER/ 118 ROUTE DE NARBONNE/ TOULOUSE CEDEX F-31077/ FRANCE/ 16-61-53 11 20
F-31077 FRANCE PIERRE MAURICE/ INFORMATIQUE/ UNIVERSITE PAUL SABATIER/ 118 ROUTE DE NARBONNE/ TOULOUSE CEDEX F-31077/ FRANCE/ (61) 53 11 20 X300
F-34075 FRANCE ATTN: CENTRE DE RECHERCHE/ INFORMATIQUE ET GESTRON/ UNIV. DES SCIENCES ET TECH. DU LANCUED/ AVENUE D'OCCIREINE/ MONTPELLIER CEDEX F-34075/ FRANCE
633886
F-38000 FRANCE ATTN: A. D. R./ CHAMBRE DE COMMERCE ET D'INDUSTRIE/ 6 BOULEVARD GAMBETTA/ GRENOBLE F-38000/ FRANCE
F-54042 FRANCE ALAIN TISSERANT/ DEPARTEMENT INFORMATIQUE/ ECOLE DES MINES/ PARC DE SAURUPT/ NANCY CEDEX F-54042/ FRANCE/ (28) 51 42 32
F-75230 FRANCE JACQUES FARRE/ INSTITUT DE PROGRAMMATION/ T 55-65/ UNIVERSITE P. ET M. CURIE/ 4 PLACE JUSSIEU/ PARIS CEDEX 05 F-75230/ FRANCE/ 336 25 25 X58 77
D-1000 GERMANY ATTN: NIXDORF COMPUTER GMBH/ BERICHT ENTWICKLUNG/ ABT DOKUMENTATION/ KAISERIN-AUGUSTA-ALLEE 21/ BERLIN 21 D-1000/ GERMANY
D-2000 GERMANY H.-H. NAGEL/ INSTITUT FUER INFORMATIK/ UNIVERSITAT HAMBURG/ SCHLUTERSTRASSE 66-72/ HAMBURG 13 D-2000/ GERMANY/ 040-4123-4151
D-2000 GERMANY BERNHARD NEBEL/ CLASINGSTRASSE 8/ HAMBURG 19 D-2000/ GERMANY/ 040/4913613
D-2000 GERMANY ATTN: INSTITUT FUER INFORMATIK/ UNIVERSITAT HAMBURG/ SCHLUETER STRASSE 13/ HAMBURG 70 D-2000/ GERMANY
D-3000 GERMANY ROLF SONNTAG/ RICHARD WAGNER STR. 27/ HANNOVER 1 D-3000/ GERMANY
D-3000 GERMANY G. MARQUARDT/ REGIONALES RECHENZENTRUM/ WUNSTORFER STR. 14/ HANNOVER 91 D-3000/ GERMANY
D-4400 GERMANY HORST STENZEL/ RECHENZENTRUM/ UNIVERSITAT MUNSTER/ ROXELER STRASSE 60/ MUNSTER D-4400/ GERMANY
D-5000 GERMANY HENK JANSSEN/ DIGITAL EQUIPMENT GMBH/ STOBERGER STR. 90/ KOLN 41 D-5000/ GERMANY/ (0221) 5486-1
D-6236 GERMANY GERHARD BLANKE/ POSTBOX 5107/ ESCHBORN D-6236/ GERMANY/ (06196) 403267
D-6300 GERMANY A. GEUBE/ ABTEILUNG BIOMATHEMATIK/ UNIVERSITAT GIESSEN/ FRANKFURTER STRASSE 100/ GIESSEN D-6300/ GERMANY/ 06 41-702/4855/56/57
D-6750 GERMANY HANS-WILM WIPPERMANN/ INFORMATIK/ F13/ UNIV. OF KAISERSLAUTERN/ PFAFFENBERGSTR. 95/ KAISERSLAUTERN D-6750/ GERMANY/ (0631) 854 2635
D-7000 GERMANY KLAUS LAGALLY/ INSTITUT FUR INFORMATIK/ UNIVERSITAT STUTTGART/ AZENBERGSTRASSE 12/ STUTTGART 1 D-7000/ GERMANY/ (0711) 2078-373/329
D-7408 GERMANY ASHOK N. ULLAL/ GOETHESTR. 10/ KUSTERDINGEN D-7408/ GERMANY/ 07121/271446
D-7500 GERMANY KARLHEINZ KAPP/ ANGEW. INFORMATIK/ UNIVERSITAET KARLSRUHE/ TRANSPORT-U. VERKEHRSSYSTEME/ KARLSRUHE D-7500/ GERMANY/ (0721) 608-3170/3898
(07247) 823 928
D-7500 GERMANY GERHARD T. GOOS/ INSTITUT FUER INFORMATIK II/ UNIVERSITAT KARLSRUHE/ POSTFACH 6380/ KARLSRUHE 1 D-7500/ GERMANY/ (0721) 751-176
D-7500 GERMANY ULRICH KULISCH/ INSTITUT FUR ANGEWANDTE MATHEMATIK/ UNIVERSITAT KARLSRUHE (TH)/ KAISERSTR. 12 - POSTFACH 6380/ KARLSRUHE 1 D-7500/ GERMANY
(0721) 608-2680
D-7900 GERMANY AXEL T. SCHREINER/ SEKTION INFORMATIK/ UNIVERSITAET ULLU/ ULLU D-7900/ GERMANY/ 0711/176 2523
D-8000 GERMANY ATTN: BIBLIOTHEK/ LEIBNIZ - RECHENZENTRUM/ BARERSTRASSE 21/ MUENCHEN 2 D-8000/ GERMANY/ (089) 2105-8489
D-8000 GERMANY MANFRED LUCKMANN/ ALEMANNENSTR. 24/ MUENCHEN 90 D-8000/ GERMANY/ (089) 2105-8276
D-8000 GERMANY PETER RAUSCHMAYER/ HERZOG-CARIBALD-STRASSE 15/ MUENCHEN 90 D-8000/ GERMANY/ 647504
D-8000 GERMANY JAN WITT/ ZFE FL SAR/ SIEMENS AG/ POSTFACH 832729/ MUNCHEN 83 D-8000/ GERMANY/ (089) 722-22651
D-8012 GERMANY BERNHARD H. BEITINGER/ INDUSTRIEANLAGEN-BETRIEBSGESELLSCHAFT/ EINSTEINSTRASSE/ OTTOBRUN D-8012/ GERMANY/ 089/60082363
D-8031 GERMANY RAINER R. LATKA/ AN DER GRUNDBREITE 1/ WESSLING D-8031/ GERMANY/ 08153-1063
500762 INDIA ATTENTION: N. V. KOTESWARA RAO/ COMPUTER TRG. UNIT/ ELECTRONICS CORPORATION OF INDIA/ HYDERABAD (AP) 500762/ INDIA/ 71611
IRELAND DIARMUID MCCARTHY/ KILMACUD/ 7 ST. KEVIN'S PARK/ BLACKROCK CO. DUBLIN/ IRELAND
IRELAND JOHN W. FINNEGAN/ SCHOOL OF ENGINEERING/ UNIVERSITY COLLEGE/ GALWAY/ IRELAND
2 IRELAND DAVID M. ABRAHAMSON/ DEPT. OF COMPUTER SCIENCE/ TRINITY COLLEGE/ 200 PEARSE ST./ DUBLIN 2/ IRELAND/ 772941 X1716 X1765
ISRAEL ATTN: THE LIBRARY/ MINISTRY OF DEFENCE/ P.O.BOX 962/ HAIFA/ ISRAEL
ISRAEL ATTN: PERIODICALS DEPT./ JEWISH NATIONAL AND UNIV. LIBRARY/ P.O. BOX 503/ JERUSALEM/ ISRAEL
ISRAEL GIDEON YUVAL/ COMPUTER SCIENCE/ THE HEBREW UNIVERSITY/ JERUSALEM/ ISRAEL
I-40033 ITALY M. E. RONCHI/ CENTRO DI CALCOLO/INTERUNIVERSITARIO/ VIA MAGNAN ELLI 6/3/ LECCHIO DIRENO BOLOGNA I-40033/ ITALY/ 576541/ 576542
I-40122 ITALY MAURO MONTESI/ TEMA S.P.A./ VIA MARCONI 29/1/ BOLOGNA I-40122/ ITALY/ 051-267285
I-40122 ITALY GUISEPPE SELVA/ TEMA S.P.A./ VIA MARCONI 29/1/ BOLOGNA I-40122/ ITALY/ 051-267285
JAPAN NOBUO WAKABAYASHI/ DEPT. OF MANAGEMENT SCIENCE/ OTARU UNIVERSITY OF COMMERCE/ 3-5-21 MIDORI/ OTARU HOKKAIDO/ JAPAN/ 0134-33-7227 (HOME)
113 JAPAN HARUHISA ISHIDA/ COMPUTER CENTRE/ UNIVERSITY OF TOKYO/ 2-11-16 YAYOI/ BUNKYOKU TOKYO 113/ JAPAN/ 03-812-2111 X2871
151 JAPAN KOHEI NOSHITA/ 1-52-4 YOYOGI/ Khibuya-KU TOKYO 151/ JAPAN/ 03-370-8031
182 JAPAN MASATO TAKEICHI/ DEPT. OF COMPUTER SCIENCE/ THE UNIV. OF ELECTRO-COMMUNICATIONS/ 1-5-1 CHOFUGAOKA/ CHOFU-SHI TOKYO 182/ JAPAN/ JAPAN 0424-83-2161 X525
560 JAPAN NOBUKI TOKURA/ DEPT. OF INFORMATION AND COMP. SCIENCE/ OSAKA UNIV./ 1-1 MACHIKANAYAMA/ TOYONAKA 560/ JAPAN/ 06 (856) 1151 X3245
NEW ZEALAND G. A. VIGNAUX/ DEPARTMENT OF INFORMATION SCIENCE/ VICTORIA UNIVERSITY OF WELLINGTON/ PRIVATE BAG/ WELLINGTON/ NEW ZEALAND/ 721-000
NORWAY HARALD ELDE/ NORSK DATA A.S./ LORENVN. 57/ OSLO 5/ NORWAY/ 47 2 21 73 71/ 47 2 22 80 90
0001 SOUTH AFRICA ATTENTION: E. N. VAN DEVENTER/ COMPUTING CENTRE/ NATIONAL RESEARCH INST FOR MATH SCIENC/ P O BOX 395/ PRETORIA 0001/ SOUTH AFRICA/ 74-9111
14 SPAIN MARTIN VERGES TRIAS/ CENTRO DE CALCULO UPB/ AV. DR. GREGORIO MARANON S/N/ BARCELONA 14/ SPAIN/ (93) 334.35.00
34 SPAIN LUIS A. GARCIA-RAMOS/ E.S.A.D.E./ AV. VICTORIA 60/ BARCELONA 34/ SPAIN/ (93) 203 7800

5-000 00 SWEDEN JOHN TIMOTHY FRANKLIN/ KRUKMAKERGATTAN #6/ STOCKHOLM S-000 00/ SWEDEN
5-100 44 SWEDEN STAFFAN ROMBERGER/ COMPUTER SCIENCE/ ROYAL INSTITUTE OF TECHNOLOGY/ STOCKHOLM S-100 44/ SWEDEN/ 08-787 7194
3-100 44 SWEDEN LARS-ERIK THORELLI/ DEPT. OF COMPUTER SYSTEMS/ THE ROYAL INSTITUTE OF TECHNOLOGY/ STOCKHOLM 70 S-100 44/ SWEDEN/ SWEDEN-08-236520
5-161 54 SWEDEN CLAES RICKEY/ HEDEBYVAGEN 5/ BROMMA S-161 54/ SWEDEN/ 08/37 6 5 37
5-175 62 SWEDEN NEIL T. KEANE/ SYSTEM DEVELOPMENT/ STANSAAB ELEKTRONIK AB/ VED DESTAVALLEN 13/ JAARFAALLA S-175 62/ SWEDEN/ 08/36 28 00
5-220 07 SWEDEN LENNERT BENSRYD/ LUNDS DATACENTRAL/ LUND UNIVERSITY/ BOX 783/ LUND S-220 07/ SWEDEN/ 046/12 46 20
5-402 20 SWEDEN AKE WIKSTROM/ DEPT. OF COMPUTER SCIENCES/ CHALMERS UNIV. OF TECHNOLOGY/ FACK/ GOTHENBURG 5 S-402 20/ SWEDEN
5-431 39 SWEDEN KURT FREDRIKSSON/ RINGLEKEN 7/ MOLNDAL S-431 39/ SWEDEN/ 4631- 41-04-15 (HOME)/ 4631-27 50 00-491 (OFFICE)
5-461 01 SWEDEN LARS G. MOSSBERG/ VOLVO FLYGMOTOR AB/ BOX 136/ TROLLHATTAN S-4 61 01/ SWEDEN/ (0520) 30900-287
5-752 23 SWEDEN OLLE OLSSON/ DEPT. OF COMPUTER SCIENCE/ADP/ UPPSALA UNIVERSITY / STUREGATAN 4B 1 TR/ UPPSALA S-752 23/ SWEDEN/ 018-138650
5-901 87 SWEDEN HANS WALLBERG/ UMDAL/ UMEA S-901 87/ SWEDEN/ 46-90 12 56 00
5-951 45 SWEDEN LARS LYSEN/ HAVSORINGSGRAND 11/ LULEA S-951 45/ SWEDEN/ (0920) 65515
5-951 87 SWEDEN JOHNNY WIDEN/ UNIV. OF LULEA/ FACK/ LULEA S-951 87/ SWEDEN/ (0 920) 98000
5-951 87 SWEDEN HANS-KURT JOHANSEN/ UNIV. OF LULEA/ LULEA S-951 87/ SWEDEN/ (0920) 98000 X386
CH-1200 SWITZERLAND DAVID BATES/ 12 CHEMIN DE TAVERNAY/ 1218 GRAND SACONNEX/ GENEVA CH-1200/ SWITZERLAND/ 98-55-44/ 41-98-11
CH-8005 SWITZERLAND RAFAEL E. EGLOFF/ HONEYWELL BULL/SCHWEIZ AG/ HARDTURMSTRASSE 2 53/ ZURICH CH-8005/ SWITZERLAND/ (01) 44 49 40
CH-8008 SWITZERLAND URS R WYSS/ BLEULERSTRASSE 2/ ZURICH CH-8008/ SWITZERLAND/ 004 1-22-28.79.61
CH-8092 SWITZERLAND NIKLAUS WIRTH/ INSTITUT FUER INFORMATIK/ ETH - ZENTRUM/ ZUERICH CH-8092/ SWITZERLAND
CH-9470 SWITZERLAND HELMUT SANDMAYR/ NEU-TECHNIKUM BUCHS/ BUCHS CH-9470/ SWITZERLAND/ CH-085/6 45 24
THE NETHERLANDS D. GOSMAN/ ZEMAN LABORATORIUM/ UNIVERSITEIT VAN AMSTERDAM/ PLANTAGE MUIDERGRACHT 4/ AMSTERDAM/ THE NETHERLANDS/ 020-522177
THE NETHERLANDS H. S. M. KRUYER/ DEPT. MSE/ KONINKLIJKE/ SHELL-LABORATORIUM/ BAD HUISWEG 3/ AMSTERDAM/ THE NETHERLANDS
THE NETHERLANDS ATTN: I.W.I.S.-TNO/ POSTBUS 297/ KON. MARIA LAAN 21/ DEN HAAG/ THE NETHERLANDS
THE NETHERLANDS ATTN: DSM/ CENTRAL BIBLIOTHEEK 4213.001/ CENTRAL LABORATORIUM/ P.O. BOX 18/ GELEEN/ THE NETHERLANDS
THE NETHERLANDS H. PAAS/ DEPT. OF SPACE RESEARCH/ UNIV. OF GRONINGEN/ P.O. BOX 800/ GRONINGEN/ THE NETHERLANDS/ 050-116662
THE NETHERLANDS J. D. ALANEN/ FREDERIK HENDRIKSTRAAT 112/ UTRECHT/ THE NETHERLANDS/ 030 + 520548
UNITED KINGDOM MAURICE O'FLAHERTY/ ANTRIM/ 444 MEVILLE GARDEN VILLAGE/ NEWTOWNABBEEY N. IRELAND/ UNITED KINGDOM
UNITED KINGDOM B. L. MARKS/ U.K. LABORATORIES/ IBM/ HURSLEY/ N.WINCHESTER ENGLAND/ UNITED KINGDOM
UNITED KINGDOM STEPHEN L. BREIBART/ EASTCOTE/ 12 ELM AVENUE/ PINNER MIDDLESEX/ UNITED KINGDOM
UNITED KINGDOM PETER J. BURNS DE BONO/ NEWFOUNDLAND HOUSE/ HUGH PUSHMAM COMPILERS/ THE QUAY/ POOLE ENGLAND/ UNITED KINGDOM/ (02013) 70510
AB9 2UB UNITED KINGDOM DENIS M. WILSON/ DEPARTMENT OF COMPUTING SCIENCE/ UNIVERSITY OF ABERDEEN/ KING'S COLLEGE/ OLD ABERDEEN SCOTLAND AB9 2UB/ UNITED KINGDOM
0224 40241 X6418
AL10 9AB UNITED KINGDOM JOHN W. LEWIS/ SCHOOL OF INFORMATION SCIENCES/ HATFIELD POLYTECHNIC/ P.O. BOX 109/ HATFIELD HERTS AL10 9AB/ UNITED KINGDOM/ 68100 X237
BB7 4DZ UNITED KINGDOM J. N. HARRISON/ MOSNA COTTAGE/ NEWBY - RIMINGTON/ N. CLITHEROE LANCBS BB7 4DZ/ UNITED KINGDOM/ GIBBURN 329
BN1 2GJ UNITED KINGDOM STEVEN PEMBERTON/ DEPT. OF COMPUTING AND CYBERNETICS/ BRIGHTON POLYTECHNIC/ MOULSECOOMB/ BRIGHTON ENGLAND BN1 2GJ/ UNITED KINGDOM/ 693655 X2273
BN1 9QT UNITED KINGDOM J. R. W. HUNTER/ SCHOOL OF ENGR. AND APPL. SCI./ UNIV. OF SUSSEX/ BRIGHTON SUSSEX BN1 9QT/ UNITED KINGDOM/ (0273) 66755 X146
BT7 1NN UNITED KINGDOM JIM WELSH/ DEPARTMENT OF COMPUTER SCIENCE/ QUEEN'S UNIVERSITY/ BELFAST N.IRELAND BT7 1NN/ UNITED KINGDOM/ (0232) 45133 X3221
CM17 9NA UNITED KINGDOM DAVID FLOOD PAGE/ STANDARD TELECOMMUNICATIONS LABORATORY/ LOND ON ROAD/ HARLOW ESSEX CM17 9NA/ UNITED KINGDOM/ 0279 29531 X345
CV4 7AL UNITED KINGDOM RUTH RAYMOND/ COMPUTER UNIT/ UNIV. OF WARWICK/ COVENTRY ENGLAND CV4 7AL/ UNITED KINGDOM
EH10 6NH UNITED KINGDOM DAVID A. COOPER/ FAIRMILEHEAD/ 52 SWAN SPRING AVENUE/ EDINBURGH SCOTLAND EH10 6NH/ UNITED KINGDOM
EH5 2XT UNITED KINGDOM T. M. SPENCE/ SYSTEMSHARE LTD/ PILTON DRIVE/ EDINBURGH SCOTLAND EH5 2XT/ UNITED KINGDOM
EX4 4Q6 UNITED KINGDOM D. R. ALLUM/ DEPT. OF PHYSICS/ UNIVERSITY OF EXETER/ EXETER ROAD/ EXETER ENGLAND EX4 4Q6/ UNITED KINGDOM
E14 6JG UNITED KINGDOM ISAMU HASEGAWA/ LUKE HOUSE/ 7 CANTON STREET/ LONDON ENGLAND E14 6JG/ UNITED KINGDOM/ 01/987-4612
HA4 9DP UNITED KINGDOM ROBERT KIRKBY/ RUISLIP MANOR/ 44 WHITBY ROAD/ MIDDLESEX ENGLAND HA4 9DP/ UNITED KINGDOM
LA1 1BA UNITED KINGDOM CHI-KEUNG YIP/ DEPT. OF COMPUTER STUDIES/ GILLOW HOUSE/ UNIVERSITY OF LANCASTER/ LANCASTER ENGLAND LA1 1BA/ UNITED KINGDOM/ (0524) 65201 EXT.4123
LA1 4YN UNITED KINGDOM BOB E. BERRY/ DEPT. OF COMPUTER STUDIES/ UNIVERSITY OF LANCASTER/ BAILRIGG/ LANCASTER ENGLAND LA1 4YN/ UNITED KINGDOM/ (0524) 65201 X4134
LA1 4YW UNITED KINGDOM ATTN: USER SERVICES MANAGER/ COMPUTER SERVICES/ UNIV OF LANCASTER/ LANCASTER ENGLAND LA1 4YW/ UNITED KINGDOM
LA1 4YX UNITED KINGDOM BRIAN A. E. MEEKINGS/ DEPT. OF COMPUTER STUDIES/ UNIVERSITY OF LANCASTER/ BAILRIGG/ LANCASTER ENGLAND LA1 4YX/ UNITED KINGDOM/ (0524) 65201
L69 3BX UNITED KINGDOM K. C. MANDER/ DEPT. OF COMP. AND STAT. SCIENCE/ VICTORIA BUILDING/ UNIV. OF LIVERPOOL/ BROWNLOW HILL/ LIVERPOOL ENGLAND L69 3BX/ UNITED KINGDOM
051-709-6022 X2022
M13 9PL UNITED KINGDOM SRISAK WATHANASIN/ DEPT. OF COMPUTER SCIENCE/ THE UNIVERSITY OF MANCHESTER ENGLAND M13 9PL/ UNITED KINGDOM
M60 1QD UNITED KINGDOM ATTN: THE LIBRARIAN/ DEPT. OF COMPUTATION/ UMIST/ P.O. BOX 88/ MANCHESTER ENGLAND M60 1QD/ UNITED KINGDOM/ 061-2363311 X2178
M60 1QD UNITED KINGDOM DEREK COLEMAN/ DEPT. OF COMPUTATION/ UNIV. OF MANCH. INST. OF SCI. & TECH/ P.O. BOX 88/ MANCHESTER ENGLAND M60 1QD/ UNITED KINGDOM/ 061-236-3311 X2341
NW3 UNITED KINGDOM H. J. ZELL/ 14 KEMPLAY ROAD/ LONDON ENGLAND NW3/ UNITED KINGDOM/ (01) 435-9396
SG2 9UT UNITED KINGDOM G. J. FREEMAN/ 125 WIGRAM WAY/ STEVENAGE HERTS SG2 9UT/ UNITED KINGDOM/ (0442) 42291 X330
S09 5NH UNITED KINGDOM J. E. AHERN/ MATHS DEPT/ THE UNIVERSITY/ SOUTHAMPTON ENGLAND S09 5NH/ UNITED KINGDOM
S09 5NH UNITED KINGDOM M. EL-NAHAS/ MATHS. DEPT/ UNIV. OF SOUTHAMPTON/ SOUTHAMPTON ENGLAND S09 5NH/ UNITED KINGDOM
SW7 2AZ UNITED KINGDOM ATTN: COMPUTING AND CONTROL COLLECTION/ LYON PLAYFAIR LIBRARY/ IMPERIAL COLLEGE/ 180 QUEENSGATE/ LONDON ENGLAND SW7 2AZ/ UNITED KINGDOM
(01) 589-5111 X2115
SW7 2AZ UNITED KINGDOM DAVID SLATER/ DEPT OF COMPUTING AND CONTROL/ IMPERIAL COLLEGE/ 180 QUEENSGATE/ LONDON ENGLAND SW7 2AZ/ UNITED KINGDOM/ 589-5111 X2722
WC1H 0PY UNITED KINGDOM I. D. GRAHAM/ INSTITUTE OF ARCHAEOLOGY/ 31-34 GURROON SQUARE/ LONDON ENGLAND WC1H 0PY/ UNITED KINGDOM
Y01 5DD UNITED KINGDOM D. G. BURNETT-HALL/ DEPARTMENT OF COMPUTER SCIENCE/ UNIVERSITY OF YORK/ HESLINGTON/ YORK ENGLAND Y01 5DD/ UNITED KINGDOM/ (0904) 59861 EXT.5641
630 090 USSR S. POKROVSKY/ COMPUTING CENTRE/ USSR ACADEMY OF SCIENCES/ NOVO SIBIRSK 630 090/ USSR
YU-61000 YUGOSLAVIA BOSTJAN VILFAN/ FAKULTETA ZA ELEKTROTEHNIKO/ UNIVERZA V LJUBLJANI/ TRZASKA 25/ LJUBLJANA YU-61000/ YUGOSLAVIA
61001 YUGOSLAVIA ROBERT REINHARDT/ INSTITUT JOZEF STEFAN/ UNIV. V LJUBLJANI/ JAMOVA 39/ LJUBLJANA 61001/ YUGOSLAVIA/ 63-261

DAVID M. ABRAHAMSON	2 IRELAND	MARGERY AUSTIN	20036	DAVID A. COOPER	EH10 6NH UNITED KINGDOM
DAVID M. ADAMS	19301	DAVE BAASCH	94707	ARDON R. CORD	95014
JAMES L. AGIN	90278	RON BAECKER	M4Y 1P9 CANADA	JOHN DENNIS COUCH	95050
J. E. AHERN	S09 5NH UNITED KINGDOM	SAMUEL T. BAKER	37130	JAMES C. COZZIE	52402
J. D. ALANEN	THE NETHERLANDS	MICHAEL S. BALL	92152	LAWRENCE F. CRAM	02159
BOB ALBRECHT	94025	CHARLES J. BANGERT	66045	JOHN EARL CRIDER	77043
R. A. ALLAN	N6A 4N5 CANADA	PAUL BARR	01778	LINDA E. CROLEY	94304
D. R. ALLUM	EX4 4Q6 UNITED KINGDOM	BRUCE BARRETT	94404	DONALD B. CROUCH	35486
STEPHEN R. ALPERT	01609	ROGER R. BATE	75023	ARTHUR C. DARTT	44115
JOHN ALSTRUP	55420	DAVID BATES	CH-1200 SWITZERLAND	HENRY DAVIS	22090
DAVID B. ANDERSON	18015	FRANK J. BATES JR.	43403	LEO DAVIS	20770
D. B. ANDERSON	94010	JOHN C. BEATTY	N2L 3G1 CANADA	ITINE DE GERLACHE	B-1180 BELGIUM
GARY S. ANDERSON	98043	O. BEAUFAYS	B-1050 BELGIUM	JOHN DE PILLIS	92507
ROBERT W. ANDERSON	92680	E. R. BEAUREGARD	17019	JOHN DE ROSA JR.	01545
RICHARD V. ANDREE	73019	MICHAEL A. BEAVER	53149	JOHN R. DEALY	90278
GEORGE W. ANTHONY	83316	MICHAEL BEHAR	06477	JAN DEDEK	94301
ROBERT L. ARGUS	47805	BERNHARD H. BEITINGER	D-8012 GERMANY	ROBERT I. DEMROW	01810
PETER A. ARMSTRONG	98109	LENNERT BENSRYD	S-220 07 SWEDEN	PETER DEWOLF	61820
JIM ARNOLD	66045	PAUL C. BERGMAN	21793	GEORGE B. DIAMOND	08826
LARRY ARONSON	10027	BOB E. BERRY	LAI 4YN UNITED KINGDOM	ROBERIO DIAS	M5P 2C3 CANADA
ATTENTION: BOB JARVIS	55455	EASTON BEYMER	77341	RICHARD DIEVENDORFF	91203
ATTENTION: CHARLES PFLEEGER	37916	GERHARD BLANKE	D-6236 GERMANY	JOHN G. DOBNICK	53219
ATTENTION: E. N. VAN DEVENTER	0001 SOUTH AFRICA	I. N. BLAVINS	5063 AUSTRALIA	BILL DODSON	93017
ATTENTION: J. M. KNOCK	60439	ROY E. BOLLINGER	94088	JOHN DOERR	95051
ATTENTION: MARJORIE HEINE	17837	JOHN H. BOLSTAD	32306	MICHAEL K. DONEGAN	23185
ATTENTION: MIKE WILDE - CONSULTING OFF.	61801	KEN BOWLES	92093	G. KEVIN DOREN	19101
ATTENTION: N. V. KOTESWARA RAO	500762 INDIA	ALEX BRADLEY	92714	KENNETH R. DRIESSEL	74102
ATTN: A. D. R.	F-38000 FRANCE	STEVEN L. BRECHER	90803	WILLIAM E. DROBISH	92714
ATTN: A. S. WILLIAMS - LIBRARIAN	92626	DAVID E. BREEDING	75229	KENNETH R. DUCKWORTH	71201
ATTN: BETTE BOLLING-LIBRARIAN	45036	STEPHEN L. BREIBART		C. E. DUNCAN	94303
ATTN: BIBLIOTEKET	DK-2200 DENMARK	C. E. BRIDGE	19898	GARY DUNCAN	92680
ATTN: BIBLIOTHEK	D-8000 GERMANY	PER BRINCH HANSEN	90007	FRANK DUNN	75081
ATTN: CCIS LIBRARY HILL CENTER	08854	CHARLES L. BROOKS	02139	WILLIAM J. EARL	92715
ATTN: CENTRE DE RECHERCHE	F-34075 FRANCE	PAUL H. BROOME	21010	RAFAEL E. EGLOFF	CH-8005 SWITZERLAND
ATTN: COMPUTER CENTER	79409	WARREN R. BROWN	02035	HARALD EIDE	NORWAY
ATTN: COMPUTER SCIENCE DEPARTMENT	84112	ANNA BUCKLEY	47401	FRED ELLENSTEIN	02172
ATTN: COMPUTING AND CONTROL COLLECTION	SW7 2AZ UNITED KINGDOM	A. CHARLES BUCKLEY	40205	JOSEPH EINWECK	88003
ATTN: COMPUTING SERVICES	55105	D. G. BURNETT-HALL	Y01 5DD UNITED KINGDOM	JOHN D. EISENBERG	19711
ATTN: DEPT. OF COMPUTER SCIENCE	SF-00100 FINLAND	PETER J. BURNS DE BONO	UNITED KINGDOM	M. EL-NAHAS	S09 5NH UNITED KINGDOM
ATTN: DIRECTOR / INST. RESEARCH	95521	JAMES W. BUTLER	62901	RANDY ENGER	01776
ATTN: DSM	THE NETHERLANDS	CARROLL E. BUTTERFIELD	94804	DONALD L. EPLEY	52242
ATTN: EARL L. MOUNTS-COMP. SCI. LIBRARIAN	15213	BETTY BUXTON	03060	HOWARD D. ESKIN	10025
ATTN: ELAINE DENTON (41-41)	90406	JOHNNIE BUZEK JR.	77005	JOHN B. EULENBERG	48824
ATTN: ENGINEERING LIBRARY	98401	J. A. CAMPBELL	2308 AUSTRALIA	JACQUES FARRE	F-75230 FRANCE
ATTN: ENGINEERING LIBRARY	95014	SAM CARPENTER	01960	MICHAEL FAY	95064
ATTN: INSTITUT FUER INFORMATIK	D-2000 GERMANY	PAULO S. CASTILLO JR.	91360	WALT FEESER	92127
ATTN: INST. FUER INFORMATIONSSYSTEME	A-1040 AUSTRIA	DAVID F. CAUTLEY	97330	ROBERT B. FINCH	98225
ATTN: I.W.I.S.-TNO	THE NETHERLANDS	JOE CELKO	30310	DONNAFAYE FINGER	98040
ATTN: J. F. MCINTYRE - LIBRARIAN	22903	W. B. CHAPIN	55112	JOHN W. FINNEGAN	IRELAND
ATTN: KHK	55414	KENNETH E. CHARLTON	98033	WILLIAM E. FISHER	90501
ATTN: LIBRARIAN	3052 AUSTRALIA	ROBERT L. CHEEZEM JR.	32935	W. J. G. FISHER	4001 AUSTRALIA
ATTN: LIBRARY	94720	HARRY R. CHESLEY	94801	DAVID C. FITZGERALD	92704
ATTN: L. LAWRIE	61820	BILL CHESWICK	18938	JOHN FITZSIMMONS	55436
ATTN: L.A.M.B.D.A.	02912	STEPHEN W. CHING	19085	KEVIN FJELSTED	55455
ATTN: MANUFACTURING COMPUTER GROUP	97077	TONY CHMIEL	60652	READ T. FLEMING	02912
ATTN: NEWBERRY MICROSYSTEMS	94022	LEO CHRZANOWSKI	14072	RUDY L. FOLDEN	92713
ATTN: NIXDORF COMPUTER GMBH	D-1000 GERMANY	JOHN CLARSON	23669	JIM FONTANA	92704
ATTN: OREGON MINI-COMPUTER SOFTWARE INC.	97221	GEOFFREY A. CLEAVE	3165 AUSTRALIA	DOUG FORSTER	09098
ATTN: PERIODICALS DEPT.	ISRAEL	GERHARDT C. CLEMENTSON	80204	CHARLES H. FORSYTH	N2J 4H2 CANADA
ATTN: SERIALS DEPARTMENT	2308 AUSTRALIA	DAVID CLINGERMAN	94563	ROBERT A. FRALEY	94304
ATTN: THE LIBRARIAN	M60 1QD UNITED KINGDOM	WILLIAM L. COHAGAN	78758	JOHN TIMOTHY FRANKLIN	S-000 00 SWEDEN
ATTN: THE LIBRARY	ISRAEL	ROBERT COLE	18017	KURT FREDRIKSSON	S-431 39 SWEDEN
ATTN: USER SERVICES MANAGER	LAI 4YW UNITED KINGDOM	DEREK COLEMAN	M60 1QD UNITED KINGDOM	G. J. FREEMAN	SG2 9UT UNITED KINGDOM
THOMAS M. ATWOOD	02165	GRANT COLVIN	75062	DAVID F. FRICK	94546
DAVID AULT	20041	JOHN S. CONERY	92714	EDWARD R. FRIEDMAN	10012
DENNIS R. AUSTIN	93105	WILLIAM S. COOKE	90402	JOHN FRINK	21045

ROBERT FULKS	85016		BOB HOFKIN	92093		RAINER R. LATKA	D-8031	GERMANY
MICHEL GALINIER	F-31077	FRANCE	DAVID W. HOGAN	78731		ARNOLD LAU	60201	
I. GANAPATHY	V6X 229	CANADA	MARGARETTA HOMMEL	01701		ROBERT A. LAWLER	55165	
LUIS A. GARCIA-RAMOS	34	SPAIN	BRYAN HOPKINS	02154		TERRY J. LAYMAN	90405	
EDWARD F. GEHRINGER	47907		C. L. HORNEY	92803		HENRY F. LEDGARD	10002	
PAULETTE D. GENES	55455		ROSS F. HOUSHOLDER	76012		K. P. LEE	70803	
W. MORVEN GENTLEMAN	N2L 361	CANADA	CAROL B. HOWELL	20770		ERKKI LEHTIMAKI	SF-33720	FINLAND
TONY J. GERBER	2006	AUSTRALIA	J. R. W. HUNTER	BN1 9QT	UNITED KINGDOM	MIKE LEMON	84112	
DANIEL E. GERMANN	55435		BOB HUTCHINS	92713		GEORGE LEWIS	94086	
J. DANIEL GERSTEN	13201		WILLIAM G. HUTCHISON JR.	08512		H. MARC LEWIS	97401	
A. GEUBE	D-6300	GERMANY	STEVEN L. HUYSER	48824		JOHN W. LEWIS	AL10 9AB	UNITED KINGDOM
ROBERT A. GIBSON	22901		JOHN W. IOBST	18049		BOB LIDRAL	61801	
R. STEVEN GLANVILLE	95051		HARUHISA ISHIDA	113	JAPAN	GEORGE LIGLER	75081	
PAUL GODFREY	94521		KENNETH R. JACOBS	20006		MOGENS LINDHARD	DK-1051	DENMARK
HELLMUT GOLDE	98195		ROBERT C. JANKU	22003		LEN LINDSAY	53719	
RALPH S. GOODSELL	01451		HENK JANSSEN	D-5000	GERMANY	GARY LINDSTROM	84112	
GERHARD T. GOOS	D-7500	GERMANY	GUNNAR JOHANSEN	DK-2800	DENMARK	PETER LINHARDT	94709	
KEITH E. GORLEN	20014		HANS-KURT JOHANSEN	S-951 87	SWEDEN	BRUCE LINK	87115	
D. GOSMAN		THE NETHERLANDS	GREG JOHNS	97077		RICHARD LLEWELLYN	21045	
P. K. GOVIND	80303		JOSEPH N. JOHNSON	94701		CARLO LOCICERO	L4W ITO	CANADA
SARA K. GRAFFUNDER	55455		JOSEPH P. JOHNSON	20016		KENT LOOBEY	97440	
I. D. GRAHAM	WC1H OPY	UNITED KINGDOM	R. WARREN JOHNSON	56301		ANDY LOPEZ	56267	
DOUGLAS M. GRANT	06901		ANN C. JOHNSTON	44691		GARY LOWELL	95404	
MARK GREEN	M4B 2E5	CANADA	D. B. JOHNSTON	4067	AUSTRALIA	TIM LOWERY	92627	
MIKE GREEN	78284		DAVID JONES	L5N 1KT	CANADA	HOUSTON P. LOWRY	99258	
TOM GREER	91775		ERNEST W. JONES	94901		MANFRED LUCKMANN	D-8000	GERMANY
WILEY GREINER	90278		NORMAN A. JULL	M4N 1A4	CANADA	FRED LUHMANN	02138	
DAVID B. GROUSE	15213		MARK JUNGWIRTH	92644		STANLEY E. LUNDE	91711	
RONA GURKEWITZ	06810		KARLHEINZ KAPP	D-7500	GERMANY	JOHN LUSHBOUGH	57069	
ROBERT D. GUSTAFSON	60657		MATTI KARINEN	SF-33101	FINLAND	WILLIAM LYCZKO	14850	
STEVEN B. HALL	44107		MILAN KARSPECK	91104		LARS LYSEN	S-951 45	SWEDEN
TERRY HAMM	97077		AHMED KASSEM	50011		ROBERT N. MACDONALD	30303	
TIMO HAMMAR	SF-33410	FINLAND	ROBERT KAST	07054		RON MAHON	15461	
N. D. H. HAMMOND	2600	AUSTRALIA	ED KATZ	55112		K. C. MANDER	L69 3BX	UNITED KINGDOM
BRIAN HANSEN	97213		NEIL T. KEANE	S-175 62	SWEDEN	DOUGLAS MANN	30339	
ANDY HARRINGTON	93111		TOM KEEL	78712		DAN MARCUS	92807	
WILLIAM J. K. HARRINGTON	08618		DONALD A. KEFFER	19438		RICK L. MARCUS	55404	
MIKE HARRIS	62704		PAUL KELLY	92714		BARRY F. MARGOLIUS	01701	
J. N. HARRISON	BB7 4DZ	UNITED KINGDOM	TOM KELLY	19335		B. L. MARKS		UNITED KINGDOM
ISAMU HASEGAWA	E14 6JG	UNITED KINGDOM	GREG KEMNITZ	55391		G. MARQUARDT	D-3000	GERMANY
S. HAYES	33313		WILLETT KEMPTON	94720		LYNN S. MARTIN	11439	
GEORGE E. HAYNAM	32901		DENIS KERMICLE	32905		GEORGE MASSAR	91367	
PETER HAYNES	L5N 1KT	CANADA	JUKKA KESO	SF-33101	FINLAND	CHARLES MATTAIR	77801	
STAN HEAD	95030		GURUPREM SINGH KHALSA	91101		STEVE MATUS	33313	
HARRY G. HEDGES	48824		B. KIDMAN	5001	AUSTRALIA	PIERRE MAURICE	F-31077	FRANCE
CHARLES HEDRICK	08903		RICHARD B KIEBURTZ	11794		KONRAD MAYER	A-1150	AUSTRIA
S. T. HEIDELBERG	94550		DANIEL B. KILLEEN	70118		JOHN K. MCCANDLISS	63188	
JOHN M. HEMPHILL	43403		ROBERT KIRKBY	HA4 9DP	UNITED KINGDOM	DIARMUID MCCARTHY		IRELAND
CHRISTOPHER J. HENRICH	07724		PAUL S. KLARREICH	11210		JOEL MCCORMACK	92014	
CARL HENRY	55057		JON G. KLASSEN	55404		RAINER F. MCCOWN	21045	
WILLIAM HENRY	10003		DONALD B. KLEIN	19010		TIMOTHY DAVID MCCREERY	94702	
MARK HERSEY	48100		STEPHEN KLEIN	01741		MAURICE MCEVOY	94708	
H. F. HESSION	22101		BARCLAY R. KNERR	92646		HENRY MCGILLTON	L5N 1K7	CANADA
CHARLES L. HETHCOAT III	77027		JOHN C. KNIGHT	23665		JOHN P. MCGINITIE	94701	
JIM HIGHTOWER	90274		PAUL KOHLBRENNER	06437		JAMES P. MCILVAINE IV	19044	
BUZZ HILL	98632		DIANE L. KRAMER	65201		MICHAEL MCKENNA	03755	
KEARNEY HILL	67401		RICHARD KRASIN	01886		RON MCKERRON	M9W 5R8	CANADA
CATHLINE S. HILLEY	58201		JAMES KREILICH	55108		JACK R. MEAGHER	49008	
MARK HIPPE	19087		H. S. M. KRUYER		THE NETHERLANDS	ROB MEANS	95035	
G. STEPHEN HIRST	52240		ULRICH KULISCH	D-7500	GERMANY	TERRY P. MEDLIN	20014	
PHILIP T. HODGE	46375		KLAUS LAGALLY	D-7000	GERMANY	BRIAN A. E. MEEKINGS	LAI 4YX	UNITED KINGDOM
THEA D. HODGE	55455		R. B. LAKE	44106		HUGO MEISSER	55427	
D. P. HODGSON	6102	AUSTRALIA	DAN LALIBERTE	55455		L. F. MELLINGER	91405	
RANDY HODNETT	97077		LARRY D. LANDIS	64108		STEPHEN F. MERSHON	22901	
TIMOTHY W. HOEL	55057		DAVID LANDSKOV	70504		W. J. MEYERS	27709	
ANTHONY E. HOFFMAN	14454		ROBERT G. LANGE	55343		M. D. MICKUNAS	61801	

FRED J. MILLER	99210		DARRELL PREBLE	30303		BILL SHANNON	44116				
JAMES S. MILLER	02138		MICHAEL PRIETULA	55455		TED SHAPIN	92669				
JOHN C. MILLER	02165		BOB PUETTE	95014		ED SHARP	84112				
ROBERTO MINIO	10010		HOWARD D. PYRON	65401		ROBERT LEE SHARP	22042				
KEITH MITCHELL	98008		M. RAHILLY	3072	AUSTRALIA	DAVID ELLIOT SHAW	94306			TOM TYSON	27409
JESSE D. MILXON	75961		EARL RALEY	19122		JOHN M. SHAW	20014			ASHOK N. ULLAL	D-7408 GERMANY
UFFE MOLLER	DK-9220	DENMARK	TIM RAND	06268		BELLE P. SHENOY	55413			ANDREW P. VALENTI	10012
MAURO MONTESI	I-40122	ITALY	V. LALITA RAO	19440		THOMAS E. SHIELDS	77005			Q. VAN ABBE	3000 AUSTRALIA
RODERICK MONTGOMERY	08876		PETER RAUSCHMAYER	D-8000	GERMANY	E. E. SIMMONS	91101			WARREN VAN CAMP	95005
CHARLES G. MOORE	48106		JERRY L. RAY	68154		P. SIMON	2308	AUSTRALIA		FRANCES L. VAN SCOY	23508
JAMES K. MOORE	22091		RUTH RAYMOND	CV4 7AL	UNITED KINGDOM	JON SINGER	60660			LEONARD VANEK	M5S 1A4 CANADA
TOM MORAN	45036		JEFFERY M. RAZAFSKY	64108		SEYMOUR SINGER	92634			WILLIAM J. VASILIOU JR.	03824
CARROLL MORGAN	2006	AUSTRALIA	EDWARD K. REAM	53705		JORMA SINNAMO	SF-33200	FINLAND		BILL VELMAN	92110
R. A. MORRIS	73019		DAN REED	72554		CHARLES SISKA JR.	90250			STEVEN A. VERE	60181
HERBERT E. MORRISON	90266		C. EDWARD REID	32303		ALAN E. SKIDMORE	66506			G. A. VIGNAUX	NEW ZEALAND
DAN MORTON	19046		ROBERT REINHARDT	61001	YUGOSLAVIA	DAVE SKINNER	90801			BOSTJAN VILFAN	YU-61000 YUGOSLAVIA
WILLIAM MOSKOWITZ	90036		STEVEN A. REISMAN	55455		DAVID SLATER	SW7 2AZ	UNITED KINGDOM		VEIKKO VISALA	SF-33340 FINLAND
LARS G. MOSSBERG	S-461 01	SWEDEN	PETER RICHARDSON	3052	AUSTRALIA	ERIC SMALL	94109			ROGER A. VOSSLER	90278
JOHN M. MOTIL	91330		PETER RICHETTA	16057		LARRY W. SMITH	55337			WAYNE VYROSTEK	97077
STEVEN S. MUCHNICK	66045		CLAES RICKBEY	S-161 54	SWEDEN	JON A. SOLWORTH	10011			M. WAITE	11740
DENNIS J. MURPHY	02138		PETER A. RIGSBEE	20375		ROLF SONNTAG	D-3000	GERMANY		WILLIAM M. WAITE	80309
TERRY MYHRER	55066		MARK RIORDAN	48824		BILL SOUTHWORTH	02132			NOBUO WAKABAYASHI	JAPAN
H.-H. NAGEL	D-2000	GERMANY	MORRIS W. ROBERTS	30303		JOHN H. SPANTON	95133			HANS WALLBERG	S-901 87 SWEDEN
BERNHARD NEBEL	D-2000	GERMANY	TIMOTHY P. ROBERTS	10573		T. M. SPENCE	EH5 2XT	UNITED KINGDOM		BOB WALSH	87106
LEROY E. NELSON	90066		LARRY ROBERTSON	91601		PAUL SPRECHER	10024			PATRICK WARD	H3C 3J7 CANADA
RICHARD E. NEUBAUER	53201		F. DOUGLAS ROBINSON	14127		D. H. SPRINGER	95131			DONALD WARREN	10024
ROBERT C. NICKERSON	94611		MICHAEL P. ROBINSON	40217		TOM SPURRIER	32901			MARK S. WATERBURY	22152
R. KEITH NICKKEY	80303		BOB ROGERS	20855		MARK STAHLMAN	10019			STEPHEN B. WATERS	13440
DENNIS NICKOLAI	55437		THOMAS A. ROLANDER	95120		F. STANLEY	98926			SRISAK WATHANASIN	M13 9PL UNITED KINGDOM
KOHEI NOSHITA	151	JAPAN	ROGER D. ROLES	01730		JORGEN STAUNSTRUP	90007			JOHN A. WEAVER	18042
RHODA P. NOVAK	91307		GENE ROLLINGS	11794		ROBERT L. STEELE II	22101			NEIL W. WEBER	93407
ROBERT E. NOVAK	60148		STAFFAN ROMBERGER	S-100 44	SWEDEN	EDWARD STEEN	01852			KEVIN WELER	15213
TOM NUTE	44106		M. E. RONCHI	I-40033	ITALY	GERALD STEINBACK	94086			LEONARD H. WEINER	79409
ROBERT J. OBERG	01701		J. B. ROSEN	55455		JAMES STEINBERG	02142			DAVID H. WELCH	92324
FLEMING M. OLIVER	94086		SAUL ROSEN	47907		HORST STENZEL	D-4400	GERMANY		ROBERT E. WELLS	02138
GARY OLIVER	97330		CARL S. ROSENBERG	94035		TURNEY C. STEWARD	94112			JIM WELSH	BT7 1NN UNITED KINGDOM
OLLE OLSSON	S-752 23	SWEDEN	RAYNER K. ROSICH	80004		JIM STEWART	08854			JOHN WERTH	89154
DICK OSGOOD	06520		EDWARD D. ROTHE	20784		JOHNNY STOVALL	01581			GREGORY F. WETZEL	66045
MAURICE O'FLAHERTY		UNITED KINGDOM	DAVID ROWLAND	97229		JEFFREY D. STROOMER	19341			TERRY E. WEYMOUTH	60532
STEVE O'KEEFE	20229		STUART W. ROWLAND	14226		CONRAD SUECHTING	74145			NORMAN D. WHALAND	10009
GARYO O'SCHENECTADY	12210		BRIAN G. ROWSWELL	2006	AUSTRALIA	JERRY S. SULLIVAN	10510			WILLIAM A. WHITAKER	22209
H. PAAS		THE NETHERLANDS	NANCY RUIZ	87115		MARKKU SUNI	SF-20500	FINLAND		JAMES D. WHITE	73019
HAL PACE	07470		CHRIS A. RUSBRIDGE	5098	AUSTRALIA	DENNIS SUTHERLAND	52302			JOHNNY WIDEN	S-951 87 SWEDEN
DAVID FLOOD PAGE	CM17 9NA	UNITED KINGDOM	PAUL RUSSELL	90733		DAVID TAFFS	02840			AKE WIKSTROM	S-402 20 SWEDEN
WILLIAM S. PAGE	19711		MARK RUSTAD	55112		MASATO TAKEICHI	182	JAPAN		E. HAROLD WILLIAMS	95050
DONALD L. PARCE	27607		WALTER R. RYPER	92634		RAMON TAN	10016			DENIS M. WILSON	AB9 2UB UNITED KINGDOM
CRAIG PAYNE	18015		LESTER SACHS	21235		JANET TAYLOR	75275			RICHARD M. WILSON	85001
DAVE PEERCY	87106		A. H. J. SALE	7001	AUSTRALIA	MICHAEL TEENER	90403			STERLING WILSON	91342
STEVEN PEMBERTON	BN1 2GJ	UNITED KINGDOM	TIMOTHY J SALO	55455		ROBERT R. TEISBERG	64108			GREG WINTERHALTER	48104
WALT PERKO	55414		E. J. SAMMONS	75080		DANIEL THALMANN	QUEBEC	CANADA		F. S. WINTERSPRING	K2A 1T2 CANADA
DAVID PERLMAN	55427		HELMUT SANDMAYR	CH-9470	SWITZERLAND	DAVID A. THOMAS	K1S 5B6	CANADA		NIELS WINTER	DK-2650 DENMARK
ROBERT D. PERRY JR.	97201		A. E. SAPEGA	06106		RICHARD T. THOMAS	43403			HANS-WILM WIPPERMANN	D-6750 GERMANY
W. J. PERVIN	75235		LYNN SAUNDERS	97077		TERRY THOMAS	95030			NIKLAUS WIRTH	CH-8092 SWITZERLAND
DAVID PESEC	44119		ANTHONY J. SCHAEFFER	47401		KIRK D. THOMPSON	85282			JAN WITT	D-8000 GERMANY
MIKE D. PESSONEY	35805		ROSS D. SCHMIDT	55343		LARS-ERIK THORELLI	S-100 44	SWEDEN		JAY WOODS	83639
BOB PETERSON	55418		MARK M. SCHNEGG	92705		CLIFTON CHANG-CHAO TING	19151			PAUL J. WOZNIAK	56569
DAVID L. PETERSON	55429		G. MICHAEL SCHNEIDER	55455		ALAIN TISSERANT	F-54042	FRANCE		URS R WYSS	CH-8008 SWITZERLAND
TRUMAN C. PEWITT	22180		WALLY SCHNITGER	92335		NOBUKI TOKURA	560	JAPAN		CHI-KEUNG YIP	LAI 1BA UNITED KINGDOM
FRANCOIS PINARD	H3S 1J6	CANADA	AXEL T. SCHREINER	D-7900	GERMANY	HOWARD E. TOMPKINS	15701			KENNETH YOUNG	90020
ALAIN PIROTTE	B-1170	BELGIUM	ROBERT SCHUTZ	11756		JOE TORZEWSKI	46530			GIDEON YUVAL	ISRAEL
RICHARD PITKIN	02114		CARL W. SCHWARCZ	01752		LARRY E. TRAVIS	53706			H. J. ZELL	NW3 UNITED KINGDOM
TOM PITTMAN	95153		STEPHEN C. SCHWARM	19898		MARTIN VERGES TRIAS	14	SPAIN		PAUL ZILBER	11797
S. POKROVSKY	630 090	USSR	ALLAN M. SCHWARTZ	47907		ROBERT TROCCHI	01754			MARK ZIMMER	94704
BARY W. POLLACK	V6T 1W5	CANADA	LARRY SELLER	91126		JOHN TUCKER	79601			ANDREW HARRIS ZIMMERMAN	94086
UDO POOCH	77843		GULSEPPE SELVA	I-40122	ITALY	JOHN TUCKER	88047			KARL L. ZINN	48104
JERRY POURNELLE	91604		FOREST VAN SISE SHAFER	08077		JYRKI TUOMI	SF-33101	FINLAND			

Introduction

It is imperative we clearly set down the semantics of type compatibility for structured variables in the programming language Pascal. The matter is urgent since the lack of an explicit set of rules in that sense has already given rise to some incompatibilities resulting from the use of different Pascal compilers.

On the basis of how a compiler implements type compatibility checking, we can currently distinguish two major classes of Pascal compilers, representatives of which will react differently to particular cases involving operations on structured variables. It is of course clear that such a conflict must not be allowed to continue, and in that sense I will try to explain how the two classes of compilers came into being and also present the reader with a few examples to display the consequences.

Two sets of rules

When you declare a variable in Pascal, its associated type can be specified by use of a type identifier (named type) as in

var *i*: *atypeid*

or by use of an explicit (anonymous) type as in

var *s*: (*one, two, three*)

Distinguishing the use of a named type from that of an anonymous type is necessary here in order to clearly express the nature and implications of a given "set of rules governing type compatibility checking" (SRTCC) implemented in a given compiler.

When the first Pascal compiler¹ made its appearance, it enforced a given SRTCC (lets call it SRTCC0). All succeeding compilers modeled after it consequently inherited the same set of rules. The reason being that the part of the compiler responsible for type analysis and checking can be transported to other computers usually without any change. In fact, SRTCC0 was a set of rules enforced by the CDC 6000 compilers which included the implementation of a restriction (particular to Pascal 6000) on type compatibility conditions of variables. With the advent of a totally new compiler², there resulted a (drastic) change in the handling of type compatibility checking, mostly affecting structured types. As in the first case all compilers modeled around this one automatically inherited a new set of rules (lets call this one SRTCC1).

The whole happening went by almost unnoticed because no one had mentioned the change in policy. Even in the recent Pascal User's Manual³, which more or less coincides with the advent of the new compiler, one cannot find a clear statement of the conditions that must be satisfied in order for two structured variables to be type compatible.

The problem now, is that in the area of type compatibility checking, representatives of both classes of compilers are not fully compatible. That means that in particular situations, a compiler enforcing SRTCC0 will reject a statement on a type-check error while another compiler enforcing SRTCC1 will accept it.

The conditions that satisfy SRTCC0 have already been stated⁴: two variables are considered as belonging to (being of) the same type if and only if they are declared using the same named type,

var *a*: *atypeid*; *b*: *atypeid*

or their associated identifiers both appear in the same list.

var *a*, *b*: array [1..10] of *sometype*

On the other hand, what we know of SRTCC1 was picked up in the source text of the compiler itself since we could not find such information elsewhere. In general, the conditions that satisfy SRTCC1 are based on the principle that: two variables are considered as belonging to the same type if and only if the data structure(s) implementing their respective type are "identical". To know what "identical" really means, one has to refer to the source text of the compiler (as we did) and understand how the Boolean function COMPTYPES works.

We purposely omitted to display the source text of that function here for two reasons: we wanted the reader to be in the same frame of mind as any Pascal programmer (who usually has no such information) when we confront him with a few revealing examples; we also could not guarantee the invariability of the algorithm from compiler to compiler since the semantics it implements have not been clearly stated as yet.

Some examples

The first example demonstrates that enforcing SRTCC1 takes away from the programmer a valuable tool, by reducing the power of named types to that of a short hand notation for anonymous types. Consider the following type definitions:

```
polar = record {polar coordinate system}
         radius, angle: real
       end;

real2 = record {real coordinate system}
        x, y: real
      end
```

Now letting the variables *vpolar* and *vreal2* be of type *polar* and *real2* respectively would you consider the following assignment statement as meaningful ?

vpolar := *vreal2*

Most Pascal programmers, I think, would expect the compiler to signal the statement as erroneous. Others of course reish at the possibility of being able to override the basic usefulness of named types. In a case such as this one however, Pascal compilers will react differently if they enforce different sets of rules for type compatibility checking. A compiler using SRTCC0 will reject the statement on a type-check error, while another enforcing SRTCC1 will let it go by without uttering a letter, since both named types are implemented using the same data structure: a record structure having two fields of type real.

Our second example shows how enforcing SRTCC1 can sometimes lead to side effects which can be disastrous. Even if the example below is based on the particular behavior of a given compiler, it should be easy for the reader to imagine the various related pitfalls he can accidentally stumble into. Consider the following declarations:

Articles

```
type r1 = record
  x, y: real
end;
r2 = record
  x: real; y: real
end;

var a: r1; b: r2
```

In the latest release of the Pascal compiler for CDC machines, memory cells corresponding to record fields declared in the same list, are allocated in the reverse order of appearance of the field identifiers. Whether it be an involuntary omission or a voluntary simplification of compiler code, the underlying assumption that a user does not care about how the memory cells are allocated to his record components, is certainly debatable. All of this however should have no ill-effects on program execution; but because of the fact that the considered compiler enforces SRTCC1, things do not come out that way. To start with, the assignment statement "a:= b" is accepted as legal. Furthermore, because of the peculiar allocation scheme described above, the execution of the assignment statement turns out to correspond to the following assignments "a.x:= b.y; a.y:= b.x"!

As a starting point for our last example, we quote from Wirth & Jensen's Pascal user manual: "Semantically, a subrange type is an appropriate substitution for the associated scalar type in all definitions. Furthermore, it is the associated scalar type which determines the validity of all operations involving values of subrange types." It is our intention to examine some of the consequences of the first part of the quote, in relation to the type compatibility of structured variables.

```
var {group 1}
  a: array [1..10] of integer;
  b: array [1..10] of 0..511;
  c: array [1..10] of 0..255;
  {group 2}
  d: packed array [1..5] of 0..511;
  e: packed array [1..5] of 0..255;
  f: packed array [1..5] of -128..127
```

According to SRTCC0, all the variables declared above are pairwise type incompatible. Under SRTCC1 however, variables of group 1 are all pairwise compatible while in group 2, only the pair e-f is compatible (as obtained when compiled by a CDC Pascal 2 compiler). The incompatibilities in group 2 stem only from differing sizes in storage occupancy (which in turn depends on the packing strategy employed). It should not be necessary to compile a program in order to find out if a given pair of variables are type compatible; language semantics should take care of that.

For the sake of discussion, let us suppose that the Pascal language is to be implemented on a computer providing instructions for efficient byte access. The implementor might choose to compile some arrays (those with the appropriate type of element) as implicit byte (packed) arrays. In such a case, the pairs a-c and b-c would no longer be compatible. Type compatibility should not be implementation dependent.

Conclusion

It was our intention to make every "pascal" aware of the importance that lies in precisely stating the semantics of type compatibility for structured variables. It is our hope that other opinions make themselves be heard.

In any event, bear the following thought in mind: making a programming language a better tool to work with, can sometimes be achieved by lowering its level of permissiveness.

REFERENCES

- [1] Pascal 6000, distribution version: 15 Feb. 1972.
- [2] Pascal 2 (i.e. Pascal 6000-3.4), distribution version: May 1974.
- [3] Jensen K., Wirth N., "Pascal User Manual and Report", Lecture Notes in Computer Science, No. 18, Springer-Verlag, 1974.
- [4] "Restrictions of Pascal 6000", distribution document, 15 Feb. 1972.



Pierre Desjardins
Département d'Informatique et de
Recherche Opérationnelle
Université de Montréal
Québec

(* Received 77/10/18 *)

A Novel Approach to Compiler Design
By James Q. Arnold
Computer Science Department
Kansas University

Disclaimer

The ideas presented in this paper reflect those of the author, and no support for them was either requested or received from Honeywell, the University of Kansas Computer Center, the Department of Computer Science at KU, or the University of Waterloo.

Introduction

During the last several months, we have had the extreme pleasure of using a Pascal compiler produced at the University of Waterloo. The compiler's five versions have given us new insights into the area of compiler design, and we would like to highlight a few of them in this

paper. In the interest of brevity, we shall not delve into all areas possible, but we hope the mention of some important items will stimulate the interested reader to reflect upon the further application of these ideas.

Our discussion will center upon the following topics:

1. Program Portability
2. Program Correctness
3. User Interface.

Although each of the areas interacts to some degree with the others, we feel these are the natural categories exemplified by the Waterloo compiler. Thus we shall strive to present them in a manner commensurate with the clarity and elegance in which they present themselves to the user.

Program Portability

A. "Use machine instructions in the compiler support package which differ from processor to processor."

Both KU and Waterloo have Honeywell 66/60 computers; KU has a processor A, and Waterloo has a processor B. When implementing a program designed for portability, it is of the utmost importance to utilize instructions which behave differently from machine to machine. The Waterloo compiler does just that; but they have taken this principle to its logical conclusion. Not only should programs behave differently, but it is even more desirable if one can arrange for an operation fault as well.

One very important consequence is providing the implementation team with the chance to practice patching core-image load modules (the compiler is only provided in load module form). This is becoming a lost art, and the necessary steps are being taken by Waterloo to keep it alive. We certainly applaud them for this.

B. "Change the language definition."

Many Pascal implementations provide extensions; Waterloo has transcended the lowly extension and introduced the new concept of outright modification. This extraordinary achievement was surprisingly easy to make. Convert the program heading into "procedure main ; ." Notice how mnemonic it is now, and how the first line immediately tells the reader what the program was written for. Notice also that the messy parameters have been eliminated. This will obviously prevent any confusion about the meaning of the undeclared variables in the program heading. Additionally, the "curious" period terminator has been dropped.

It amazes us even now how such simple modifications could add so much to the clarity and portability of programs. We wonder why Niklaus Wirth didn't think of these things himself. Naturally, the compiler rejects Standard Pascal, but this is a blessing in disguise. As soon as we have reached the same level of insight as the designers at Waterloo, we shall certainly let the reader know what the blessing is.

Program Correctness

A. "Distribute compilers which are not debugged."

Once again, the compiler is used as an educational tool. Since most programmers can not be assumed to know Pascal, any compiler for the language should encourage the user to study the user manual. This can be done in several ways, but some of the techniques used by Waterloo struck us as being particularly noteworthy.

(i) "The compiler should abort on some simple syntax errors."

One example follows which aborts the compiler:

```
type sex : (female, male) ; { : should be = }.
```

We relayed our initial reaction of concern to Waterloo, but they reassured us with an explanation [Pascal Release Bulletin, September, 1977], "The compiler is based on an LALR parser, and LALR parsers are famous for this behaviour."

Naturally, we were grateful to receive this lesson in parsing theory. It also illuminated a new attitude which should be instilled in all compiler writers of the future. If there are syntax errors in the program, the programmer must not know the language in the first place. Encourage him to read the manual; furthermore, don't waste computer time by looking at the rest of the program if the error was one which could only have been made by a complete dolt. A compiler abort is the quickest and cheapest way to quit scanning.

(ii) "The compiler should abort on some syntactically correct constructs."

We believe this is a truly ingenious device to educate the experienced programmer. While aborts on syntactic errors are directed at people who still make mistakes, this kind of abort is aimed right at the knowledgeable one. Furthermore, this will help the programmer expand his Pascal vocabulary by forcing him to use different language features than the ones he really wants to use. Surely no further explanation of the power of this device is needed.

(iii) "The compiler should generate incorrect code, which still executes."

This must be considered the successor of both (i) and (ii). Once the program has sifted through the compiler, and a load module is obtained, it will surely help the programmer understand the program better if it runs incorrectly. Hand traces are illuminating, and they are essential in the development process of a program. Waterloo has extended their application even further to include post-runtime. Once again, we are amazed at the insight and courage needed to make this intellectual leap.

At this point it should be noted that these principles combine to help unify the user community. At KU a "bug list" has been compiled (by hand), and all users are invited to contribute. It is a marvelous tool for bringing people together. Furthermore, we have actually discovered that some of those people prefer entomology to computer science. They are indebted to Waterloo for providing the initial motivation to explore the field. The compiler seems to serve as a limited, occupational counselor.

B. "Give brief error messages without referring to program text."

This will obviously make the user study the whole line (or procedure, or program). Our favorite message of this kind is "Syntax error near 'identifier'." The runner-up is "Syntax error near 'program'." It should be apparent how exemplary the latter is, especially for beginning programmers who are trying to write their first Pascal program using Wirth's definition. (The second message actually refers to a use of the defunct keyword, program, which is not allowed to be used at all.)

User Interface

A. "Make the compiler options a dynamic set."

All programmers like to feel they are in control of their machine. One way to foster this feeling is to provide an interface to the

language processor which keeps changing. This is superbly executed by the time sharing command scanner for the compiler. To allow the use of upper case keywords, we have used "-dualcase," "-uppercase," and "-singlecase" with different versions of the compiler. Inexplicably, they neglected to recognize those options only when typed in lower case, to completely rule out the use of an upper case terminal. Perhaps that will be provided in a future version.

In addition, it is also advisable to maintain a dynamic set of requirements for the placement of the options on the command-line in relation to the filenames. Particularly useful here is requiring one

option to follow the name of the source file, while all others must precede it. As new versions evolve, however, this restriction may be loosened. Once again, Waterloo seems to have missed the chance to change which option must follow the filenames.

B. "Do not implement all of the options documented at one time."

The reason for this is to prevent confusion in the user by providing too many things at one time which all work.

C. "Create files for the user, without checking the names of the files already in existence."

We must admit that this is the best feature of all, and the compiler performs with characteristic aplomb. We were also delighted to discover that the filenames used depend upon the options given in the command-line. This is necessary to prevent standardization. One example will open the door for future implementations to follow.

If the user simply wants to compile the program and get a relocatable object deck, the compiler creates a file for that by appending ".o" to the name of the source file (it can not be specified by the user, a nice extra). The Honeywell time sharing file system limits filenames to eight characters; what should the compiler do if the name of the source file is eight characters long? Perhaps the reader has already guessed. The only logical thing to do is write the object deck right over the source. Nothing could be more clever. Source files are useless when one has the object deck. Why waste file space maintaining both?! This will also prevent the user from making wasteful and costly listings of the program. We are continually impressed by the resourcefulness displayed with this unique feature. An important thing in its implementation is the requisite lack of system documentation. The best system features are always the ones which are left undocumented.

Conclusion

We have given new standards for compiler construction, and we have shown how the Waterloo compiler exemplifies them. There are still many things which we have not explained- features we have not discussed. Readers are invited to write the author for more information; perhaps those with access to a Honeywell installation can arrange to get their own version of the compiler. We are sure it will be a rewarding experience.

(* Received 77/10/21 *)

UNIVERSITY OF CALIFORNIA, SAN DIEGO

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO



SANTA BARBARA • SANTA CRUZ

INSTITUTE FOR INFORMATION SYSTEMS

MAIL CODE C-021
UCSD
LA JOLLA, CALIFORNIA 92093

Status of UCSD PASCAL Project

Kenneth L. Bowles
Director, IIS
27 November, 1977
(714)452-4526

This is a brief report on the current status of the UCSD PASCAL project intended to answer the questions of the hundreds of people who have been writing to us or calling by telephone. It is our intention eventually to reach a steady state in which we can afford to have full time help capable of responding to such inquiries. For the present, we have to apologize once again to those who may have been kept too long waiting for replies.

1. Nature of the Project

The project is one of the principal activities of the Institute for Information Systems (IIS). Like other "Organized Research Units", IIS is operated primarily to provide resources and activities within which students and faculty can conduct research and development projects. Within the range of such activities, projects may support instruction and other public services, though the more usual activities of an ORU involve only basic research.

Under IIS we have developed a major software system for stand-alone microcomputers based on the PASCAL language. The initial reason for developing the system was to support instruction activities at UCSD. However, the system is designed for general purpose use, particularly for the development of interactive software, and for software development in general. The system has matured sufficiently that we are distributing copies to outside users at a \$200 fee which pays for some of the student part-time assistants who provide support to users and maintain the software. Under prevailing University policies, we are not attempting to recover capital costs from the fees paid by users of our software package. However, a number of interested industrial firms have provided assistance to further the project through unrestricted grants to the Regents of the University of California marked for use by our project. These grants are our principal source of operating funds at the present time.

Since the PASCAL based software system was developed with the intent to support long term instructional projects, we have placed very high emphasis on machine independence. We expect the repertoire of instructional software developed to use the underlying system to grow very large. The development costs for the instructional software will eventually dwarf the costs of the hardware on which it operates. Since the industry is introducing new microcomputer designs at a rapid rate, we wanted to be able to move the entire software repertoire to new machines with a minimum of effort. As will be detailed in later sections of this note, our system is now running on 5 dis-similar processors, with more planned in the relatively near future. We are using the Digital Equipment LSI-11 for teaching. Versions for the 8080 and Z80 microprocessors are operational and will be ready to distribute on or about 1 January, 1978.

We intend to continue promoting the use of our PASCAL-based system on as many popular microprocessors as practical for two reasons. First, this should provide IIS with a source of continuing income to pay for student projects. Second, PASCAL with extensions is a superior language for system programming, and we believe that it is in the public interest to assist in the current effort of many people and institutions to promote wider use of PASCAL in place of some of the earlier high level languages. Though PASCAL may have some shortcomings for specific applications, when compared to specific proprietary languages, we regard it as by far the best general purpose language now in the public domain.

Our current Research and Development interests include:

- a. Methods of making large software systems like ours more readily transportable to new processor architectures.
- b. The use of microcomputers as intelligent communications devices to assist humans to work together even when located thousands of miles apart. This interest will eventually involve us in a variety of complex software issues. In the near term it will provide us with an efficient method of supporting users of our software system who are remote from UCSD.
- c. Joint use of microcomputers and Keller's Personalized System of Instruction (PSI) as a means of offering high quality college level mass education at lower costs per enrolled student than associated with conventional methods. A published introductory textbook on problem solving using PASCAL, and a library of automated quizzes and record keeping software to go with the textbook, are available to others as a first step in this direction.
- d. Exploration of possibilities and software problems associated with new hardware devices or architectures adaptable to the purposes already described. Examples include video disks, low cost X-Y input devices, and low cost strategies for interconnecting many semi-independent microcomputers.

Partly as a matter of self preservation, we have become interested in the problem of standards for the PASCAL language. The United States Defense Department and many large industrial corporations have recently decided to use PASCAL as a base language which they would extend, and possibly alter, to create system implementation languages. Although almost every organization has chosen to extend or alter in slightly different ways, we have found that the intent portrayed in most instances is very similar. In our own implementation, we too found it necessary to extend PASCAL, and in very minor ways to alter the base language as described in Niklaus Wirth's widely read "Report" on the language (see Jensen, K. and N. Wirth, "PASCAL User Manual and Report, Springer Verlag, 1975). We, and many others in the PASCAL User Group, are very much concerned that all this extension and alteration activity will result in PASCAL going the way of BASIC for which hundreds of dialects are in common use. We believe that a chance still exists to gain consensus on a substantial family of PASCAL extensions for system programming, provided that this can be brought about within the next 6 to 12 months. Unless someone does so before us, we intend to convene a summer workshop for representatives of some of the major using organizations in the hope that such a consensus can be reached.

Another ancillary activity of the project has been continuing search for low cost microcomputer hardware of high quality for use in educational institutions - particularly ours. We have been advising and collaborating with EDUCOM regarding establishment of quantity purchase discounts for stand-alone microcomputers. The first microcomputer to be included in the EDUCOM discount program is the

Terak Corporation 8510A, which is based on the Digital Equipment LSI-11. For the current market, the Terak unit's price of \$5500 to EDUCOM member institutions is highly competitive. Nevertheless, the rate of new announcements from the industry continues very high, and we believe that it is all but impossible to predict what hardware will provide the best cost/performance tradeoff for as long as even one year in advance. Of necessity, our search has concentrated on stand-alone microcomputers with graphic display capabilities, and with enough main memory and secondary storage to handle the extensive software and course materials with which we are working. We welcome inputs on this subject from other institutions, or from any vendor, and endeavor to keep EDUCOM informed of opportunities that seem advantageous. In addition to educational and communications applications, we are interested in word processing and business applications of the same machines.

The following sections of this status report contain brief detailed summaries covering most of the topics just enumerated. If we haven't answered your questions yet, please try again with a phone call or letter. For those who already have our software system in use, we will soon be providing an automatic Tele-Mail facility on a dial-in basis. This should improve dramatically our ability to keep you informed and to respond when software difficulties arise.

2. The PASCAL based Software System

Thus far, users who have received our first released system have copies of version I.3, which was completed in mid August this year. We have ourselves been using version I.3c since early October. By the end of the December academic break, we hope to have a version I.4 available for distribution. The most significant generally useful addition since the I.3 release has been the screen-oriented editor. A major package for preparation of CAI programs, following the general philosophy of the University of California Irvine Physics Computer Development Project (PCDP), has been placed in operation on the Terak 8510A microcomputer. Except for some graphics materials within this package, it can be used on a wide variety of CRT screen display devices. Software more specifically oriented to the Terak machine is also available, and includes a character set editor (for the soft character generator), and a bookkeeping package for keeping track of student progress in a large Keller Plan (PSI) class.

The software system is currently executed in a pseudo machine interpreter, which emulates a hypothetical real machine designed to handle PASCAL constructs efficiently. Our pseudo machine is similar in concept to the P-machine distributed by Wirth's group at Zurich, but we have made extensive changes to compress the PASCAL object code into a much smaller space than possible with the Zurich interpreter. The interpreter, and run-time support routines, currently occupy about 8K bytes of main memory. The interpreter is in the native machine language of the host machine, and thus far has been coded by hand using the host's assembly language. All other code in the system is written in extended PASCAL.

While the interpreted object code runs roughly five times slower than native code for the host machine, several factors allow our large system programs to run substantially faster than this would indicate. The strategy of code compression makes it possible to run relatively large programs without time consuming overlays. For example, the complete compiler occupies 20K bytes in a single overlay. Since the system is designed for frequent compile/go cycles associated with instruction, we have added several built-in procedures and functions to handle low level operations needed frequently by the compiler. As a result, the compiler translates PASCAL source code at about 650 lines

per minute on an LSI-11 with its clock set to 2.2 MHz. On a 4 MHz Z80, the compile speed will be slightly faster than this.

Interpreter based versions of our system are now running on 5 distinct processors, and two others are close to completion. Those operating include DEC PDP-11's ranging from the LSI-11 to the 11/45, using either floppy disks or RK05 disks for secondary storage. Versions for the 8080 and Z80 are operating in our laboratory, but more of that later. Sperry Univac Minicomputer Operations at Irvine is using the system on the V-75 and related machines. Another group at UCSD has the system running on the Nanodata QM-1. With support from General Automation, a version is close to completed on the GA-440 family of machines. National Semiconductor has an implementation nearly completed on the PACE microcomputer.

The principal modules of the system as it will be distributed in the 1.4 release include the following:

- PASCAL compiler
- File manager (capabilities similar to DEC's PIP)
- Screen oriented editor (cursor positioning, immediate updates)
- Line oriented editor (similar to DEC's RT-11 Editor)
- Debugger (single line execution, reference to variable contents)
- SETUP program (adapt system to most ASCII terminals)
- BASIC compiler (ANSI standard plus strings)
- Operating System and user command interpreter
- PASCAL pseudo machine interpreter
- Linker program (for linking independently compiled program segments)
- Desk Calculator utility program

Users of the Terak 8510A may, on request, also receive copies of the CAI package, and automated quizzes for the introductory textbook, as well as the bookkeeping package.

Documents describing all of the above are available, and part of the release, but not all documents can be considered complete at this time. We distribute source and object code files on separate floppy disks formatted to be compatible with the IBM 3740 standard, with 512 byte blocks laid out in alternate 128 byte sectors according to DEC's standard. We have occasionally sent copies recorded directly on disk packs for the RK05 drives. All other media are painful or impossible for us to handle, and no promises are made to use them. Those who order the full \$200 release package will be sent both the documents, and printed listings of the source programs. Copies of the descriptive documents, amounting to approximately 150 pages, may be ordered at \$10 each (checks payable to the "Regents of the University of California") to cover printing and handling costs.

3. Minimum Configuration

In order to use the compiler, you need a total of at least 48K bytes of main memory, including the 8K bytes assigned to the interpreter. We use 56K bytes. Ideally, the interpreter should be completed re-entrant and thus it should be possible to operate the interpreter from Read Only Memory. To date, the ideal has not quite been achieved, as none of our sponsors has yet insisted on that feature.

At present, the system we use with students contains several built-in functions not needed for system development. The aggregate size of these functions is large enough to prevent compiling the compiler itself, or the operating system, even on a 56K byte system. Accordingly, we currently have two versions of the system, one for students, and one for system development. Within the next few months, we plan to add a means of configuring general purpose libraries for the

system, and by that means expect to be able to return to a single version for all purposes. That single version should be practical to use in less than 48K bytes for some purposes.

If you intend to compile on one microcomputer, and to executed object routines on others, the others can get by with as little as 16K bytes of main memory if the operating system is not used. The resident portion of the operating system occupies about 8K bytes itself. This will undoubtedly be reduced as part of the libraries project.

The system is designed to be used with standard IBM compatible floppy disks. Clearly it can be used with other varieties of floppy disks, or with other secondary storage media, with appropriate I/O drivers. The I/O drivers have proven to be one of our principal bottlenecks, and we make no promises in advance about supporting other devices. For DEC PDP-11 machines, the floppy disk drivers are assumed to be compatible with the RX-11, or with the Terak 8510A drives. Hard disks are assumed to be compatible with the RK05.

The system is normally supplied with the assumption that the user has a simple line-oriented ASCII terminal. The SETUP program can be used to configure control codes for more appropriate use of most CRT terminals. Copies of the system supplied to users of the Terak 8510A make fairly extensive use of the special graphics and character generator facilities of that machine.

4. 8080 and Z80 versions

The Z80 version is now running on the Tektronix 8002 Microprocessor Development Aid system, for which Tektronix has supplied substantial support to the project. The 8080 version uses virtually the same source code as is used on the Z80, with conditional assembly altering certain passages in the source to substitute for a few of the extended Z80 instructions that proved useful.

Release of the 8080/Z80 version of the system for other machines has been held up primarily because of the awkwardness of handling I/O. We currently have a Zilog Development System, a Processor Technology SOL system, and a Computer Power and Light COMPAL-80 system. The floppy disk provisions for each of these machines is non-standard. As a result, we have been forced to down-load programs via serial interfaces to get from the LSI-11 host machines used for development over to the new 8080 or Z80 based host. This has proven to be a very time consuming process, and a serious bottleneck in our work. Moreover, we are somewhat amazed to find that the Assembly of large programs on these machines runs almost a factor of ten slower than compilation of PASCAL programs that carry out similar tasks! Clearly, something has to give if we are to reach the objective of distributing PASCAL systems for more than a few 8080 and Z80 based machines.

The solution to this problem that we now plan to use is based on the extensive market penetration of an operating system called CP/M, which is a product of Digital Research Inc. We have talked with many OEM and hobbyist users of the 8080 and Z80 who wanted to know when we would have the PASCAL system operating under CP/M. We then learned that CP/M is distributed in a package which assumes that most users will write their own I/O drivers. In effect, CP/M establishes a quasi standard for the interface between an 8080/Z80 operating system and its I/O drivers. With thousands of copies working in the field, CP/M seems to be far ahead of the field in this area. Accordingly, we have decided to release the UCSD PASCAL System for 8080 and Z80 users in a form that will work with I/O drivers and bootstrap loaders developed for use with CP/M. This does not mean that our package will run under CP/M. However, if CP/M runs on your machine it should be relatively easy to

install the PASCAL system on that machine. We have been in contact with Digital Research on this concept, and they have offered to cooperate. If you do not have CP/M for your machine, the implementation package may be obtained from Digital Research Inc., Box 579, Pacific Grove, CA 93950 for \$70. Since CP/M has been implemented on a very wide variety of 8080 and Z80 based machines, there is a high probability that CP/M I/O drivers are already available from Digital Research or someone else for your machine.

Alteration of our present interpreter to match the CP/M I/O calling conventions has proven to be very simple, at least on paper. We expect that some implementors of CP/M will have installed standard console input routines which automatically echo to the standard console printer or display device. This will necessitate a change, since our system uses both echoing and non-echoing console input. At this writing, the exact method to be used is under discussion. Barring some unforeseen calamity, copies of our system designed to run with CP/M I/O drivers should be ready for distribution by early January, 1978. The distribution medium will be IBM compatible floppy disks formatted in a manner yet to be finally specified. We will undertake to transform the system for other media and other formats, in general, only if a copy of the necessary hardware is available in our laboratory, and only if funds are available to pay for the extra conversion work.

For many of the 8080 based machines we have seen, the most practical way to install our system will be to use 48K bytes of RAM augmented with 8K bytes of ROM for the interpreter. Any additional RAM or ROM required by the host processor system will also be needed.

5. PASCAL Extensions and Alterations

We have attempted to implement faithfully as much as possible of PASCAL as defined in Jensen & Wirth's User Manual and Report. The principal extensions to PASCAL embodied in our system are related to STRING variables, Turtle Graphics, handling of disk files, Segment (overlay) Procedures, and several functions for support of the system itself. Alterations include a prohibition against passing procedure or function identifiers as parameters, restriction against GOTO out of a procedure, the addition of EXIT(<procedurename>) to effect a normal exit from the procedure named in the parameter, and a change in READ applying to the interactive INPUT and KEYBOARD files. Further details than given in this section are given in our system release documents.

Type STRING is a pre-declared record containing a character count followed by a packed array of characters. Built-in procedures and functions include LENGTH, POS(ition), INSERT, DELETE, COPY (i.e. extract), CONCATenate, SCAN, FILLCHAR, MOVERIGHT, and MOVELEFT. The last four of these also operate on conventional packed arrays of characters.

Turtle Graphics describes a technique originated by Seymour Papert of MIT in which one can either MOVE a cursor (called the "turtle") an arbitrary number of screen units in the current pointing direction, or TURN an arbitrary number of degrees at the current position. A PENCOLOR procedure allows the line drawn by a MOVE to be either WHITE, BLACK, or NONE.

The disk file extensions allow working with fixed length logical records corresponding to any legal <type>, which might typically be a RECORD data structure. GET and PUT operate normally through a window variable of the same <type>. OPENNEW creates a new file, OPENOLD opens a pre-existing file, and CLOSE allows saving or purging a file. SEEK (which will be distributed with the I.4 system for the first time) allows random access to logical records within a file.

SEGMENT Procedures are separately compiled and then linked into the host program using the LINKER. A Segment procedure is only loaded into main memory when it is entered for the first time, and its memory space is deallocated upon exit from the first invocation.

READ(INPUT,X) is defined by Wirth as X:=INPUT↑; GET(INPUT); which we find to be extremely awkward for interactive use. Our solution is to place the implied GET before the implied assignment in the case of interactive files of type TEXT. READ operates as defined in Wirth's Report for other TEXT files.

PACKED records on our system which fit within 16 bit fields are automatically packed and unpacked without explicit action by the programmer.

6. Introductory PASCAL Course and Textbook

Many of those inquiring about our system have heard about it through having seen the textbook "Microcomputer Problem Solving Using PASCAL" by the author of this note, published this fall by Springer Verlag. If you haven't seen a copy, they may be obtained from Springer at 175 Fifth Ave., New York City, NY 10010.

The book is the basis for teaching the large attendance introductory computer science course at UCSD. This course comes close to matching the specifications for course CS1 in the recently published curriculum recommendations from ACM's SIGCSE. The approach is non-numerical as far as practical, as a tactic to reach the many students who come to us with inadequate preparation in high school mathematics. The problem solving and programming approach taught is the same as we would teach even if all the problem sets were mathematics oriented. Because many problem examples and illustrations use our string and graphics extensions to PASCAL, the textbook currently assumes that the student will have access to a computer which runs under the UCSD PASCAL system. We will be glad to discuss the possibility of conversion to other software systems, but have very limited resources to apply to such conversions. There are several stand-alone microcomputers now being sold in large quantities on which our system would run, given a small conversion effort, and we would welcome support funds to pay for such conversions.

Software in the form of automated quizzes is available with our system release for those who may wish to teach using the textbook. Each chapter in the book has a list of study goals for the students to achieve. Wherever appropriate, the quizzes test for mastery of the topics enumerated in the goals lists. The quiz programs have been implemented using a set of CAI primitive routines patterned after the well known DIALOG CAI system developed at U.C.Irvine by Alfred Bork and his colleagues.

The introductory course is taught using Keller's Personalized System of Instruction (PSI). PSI has been found to be a more successful method of instruction than any other method commonly used in universities and colleges. This success is achieved, almost completely without conventional lectures, by using experienced students as Learning Assistants called "proctors". The characteristics of this method make us believe that it is possible to offer this course, or others constructed along the same lines, on a packaged basis for use at other institutions. A separate paper describing this possibility in detail called "Microcomputer Based Mass Education" is available from the writer of this status report.

7. Tele-Mail User Support Facility

We have reached the point where it will be possible for us to begin operating a dial-in computer "mailbox" by early in the winter quarter. We have been using the Terak 8510A machines occasionally as intelligent terminals for exchanging messages via the large B6700 computer operated by the campus computer center. Our own Tele-Mail facility will use its own single telephone number reachable directly from the national dialed telephone network, or internally via the California state government telephone network. Paid subscribers to our software release will be notified when this mailbox facility is ready to be used.

The mailbox will be operated primarily to serve users of our software system. It will provide notices of recent bug corrections, down-loading of program files (either source or object) where appropriate, notices on new additions to the software and new machines on which implementations have been completed, and other useful information from us to the users. It will also serve as a means for us to collect messages from specific users, and to answer them expeditiously, without the hassle of both parties having to be at their telephones at the same time.

Through the use of block transfer software, the mailbox will make relatively efficient use of the dialed telephone network. We would like to begin immediately by offering a dial-in port at 1200 bits per second. However, the present state of confusion in the industry at that speed (which is the fastest one can use with acoustic couplers) leads us to move cautiously. We can and will install a port at 300 bits per second using the standard Bell 103A equivalent conventions. The system will answer an incoming call from an ordinary terminal by providing a brief summary of recent developments. It will otherwise expect a "handshake" from a special file transfer program that we will provide to users of our software package. This program will be the means of interchange based on efficient transfer of messages in the form of complete files. If you wish to send an ordinary text message to us, you will prepare the message using either of the editors built into the system. Only after the message is complete will you need to make the telephone connection.

8. Forthcoming Improvements

As mentioned earlier, our next significant improvement in the software will be a more flexible system allowing libraries of programs. One of the main reasons for doing this will be to allow the software to be configured to make efficient use of main memory in cases where the user does not need all of the built-in facilities. For example, we have no need for turtle graphics when compiling large system programs.

One of the long awaited features of the new library system will be an arrangement allowing mixture of PASCAL procedures with Assembly language routines and/or procedures compiled directly to the native code of the host machine. The necessary assemblers and code generation will come somewhat after the library system is operational. If all goes well, the library system should be ready to distribute during the winter quarter of 1978. The assemblers and native code versions of the compiler will come somewhat later as time for the necessary work permits.

Many people have asked whether we have in mind extensions to support Concurrent PASCAL, or similar facilities to allow independent processes running concurrently. This is something we would like to do eventually, but our current resources do not allow making definite plans in this area.

(* Received 78/01/03 *)

UNIVERSITY OF CALIFORNIA, BERKELEY

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO



SANTA BARBARA • SANTA CRUZ

PROGRAM IN QUANTITATIVE ANTHROPOLOGY
DEPARTMENT OF ANTHROPOLOGY

2220 PIEDMONT AVENUE
BERKELEY, CALIFORNIA 94720

21 December 1977

SUGGESTIONS FOR PASCAL IMPLEMENTATIONS

These suggestions are not based on implementation experience (I have none), but on using, teaching, and arguing about the language and on maintaining and transporting Pascal programs across several machines. Items 1 through 7a below concern implementation and have no bearing on the Pascal language. Items 7b through 10 would affect portability if implemented on only some machines, and thus seem worth considering as "conventional extensions."

1) A very nice feature offered by some Fortran compilers is a special comment construction which is treated as a comment under one compiler option (the default), while such comments are compiled if the appropriate option is specified. This feature allows debugging code to be inserted in with the rest of the program at no cost to the size or execution time of the final production object code (since the debugging code is treated as comments in the final compilation). In addition to debugging, this feature can be used to produce two slightly different object versions from a single source program, or for insertion of machine-dependent extra features in an otherwise transportable program. This could be implemented in a Pascal compiler in two ways:

a) The most analogous implementation would use a \$ comment option which would cause the preceeding open and the following close comment symbols to be ignored under "debug" mode, while otherwise these \$ comments would be treated as comments. This means of implementation has the advantage that it can be used for declarations or for actions. It has the disadvantage that no comments can appear within the "debug" code since few (if any) compilers allow nesting of comments (some allow pseudo-nesting through different comment symbols).

b) A different implementation would use Pascal language structure rather than special comments: "debug" code would appear within an if statement which has as its Boolean expression only a single constant identifier. The following (compound) statement is compiled if and only if the constant has the value TRUE. The code for the if statement itself is not generated in either case. Means of implementation b) has two advantages over a). First, the programmer can provide as many different "constant switches" as desired: one actually for debugging, another if certain features are available on the host machine, etc. The second advantage is that any Pascal compiler will generate correct code from programs using these constant switches, in fact, I used them frequently to move programs back and forth between machines (Texas-Austin CDC and DEC 10). Of course, such programs will run less efficiently if the special use of the Boolean constant is not recognized at compile time. For the specific purpose of debugging, it would be convenient to have a predefined Boolean constant identifier DEBUG global to the main program. This would be FALSE unless set by the job control command invoking the compiler, and could as usual be overridden by a local same-named constant.

2) Large programs would be easier to read and debug if the compiler source listing included, at the end of each procedure or function, the names of global variables referenced in that routine, and the routine in which they were defined.

Suggested Extensions to Pascal
Part I

R. A. Fraley
University of British Columbia
August 25, 1977

- 3) A compiler option to flag non-standard constructs is quite useful. At some future time it may be desirable to expand this to flag only constructs which are both non-standard Pascal and non-"conventional extensions".
- 4) If the reserved word PACKED is ignored by the compiler, the procedures PACK and UNPACK should be implemented as the appropriate FOR assignment loops.
- 5) If LINELIMIT (not a part of Standard Pascal) is not called for a file, the default should be no line limit at all, rather than a small positive limit. A small default limit is undesirable for two reasons: a) programs writing to several files may run correctly during testing and then fail on a large production run (Pascal 6000 enforces a limit on every output file, not just printer files), and b) portability of programs written on machines not implementing this extension is impaired.
- 6) The syntax which makes one-pass compilation possible also makes possible a compiler for interactive program entry, with syntax checking after each line of code is entered by the user. Actually, program entry could be done by a syntax scanner which copies good input to a file but does not generate code (thus would be transportable). If the user was given some simple editing facilities, such a system would compare favorably with many of the selling points of a good Basic interpreter.
- 7) The following two comments on numeric input apply to either free-field or formatted input:
 - a) The standard language definition of <unsigned real> (Jensen and Wirth, p 111, p 138) requires digits to precede and follow the decimal point. This restriction on the source language should not carry over to input performed by user programs, but unfortunately the "read real" routine in many run-time libraries does not accept input such as 5. and .5. Since the user/programmer has no control over the form of input in many applications, this restriction can be a considerable inconvenience ("We would want to write a Fortran pre-processor anyway, so why not just do the whole thing in Fortran...").
 - b) The suggestion by Peter Grogono allowing recovery from input errors (PN #9-10, p 51) is excellent. The solution to the "123J5" problem he mentions is simply that such matters are the programmer's responsibility: she/he would have to check ITEM.FOUND = TRUE AND INPUT↑ = " " if this is an appropriate expectation of the input data.
- 8) File buffer emptying is an important operation in interactive environments. Agreement on a "conventional extension" for this would reduce the current name proliferation: SENDTTY(<filename>) on Texas CDC; BREAK on Nagel DEC 10 (no argument); FLUSH(<filename>) on UCB UNIX PDP 11. Help!
- 9) Compilers should pad short strings with blanks. This may be considered a departure from the strict typing of Pascal, but counting of characters is very undesirable in a high-level language. (It is interesting to compare Pascal in Wirth's A+DS=P examples p 319 with 241 and 17).
- 10) In addition to the predefined constant MAXINT, the following constants would be useful: MININT, MAXCHAR, MINCHAR, MAXSETINT, MINSETINT, MAXSETCHAR, MINSETCHAR. These allow the user to define types such as INTSETTYPE = SET OF [MINSETINT..MAXSETINT]; and even write statements such as IF CH IN [MINSETCHAR..MAXSETCHAR] THEN independently of the machine environment. The SETINT and SETCHAR constants could probably be derived from each other with CHR and ORD, except that if only one were provided the other could not be used in declarations.

That's all

Willett Kempton

(* Received 77/12/23 *)

A number of extensions and modifications to Pascal are suggested below. It is the author's belief that Pascal, as it stands, cannot compete successfully with more complete languages in production environments and over wide ranges of applications. Some of these suggestions would significantly change the language, but they would hopefully preserve its clarity and simplicity. Some of these extensions are optionally available in the Pascal/UCB compiler for the IBM 370 [PUGN #8].

Dynamic Arrays and Parameterized Types

The inability to write procedures which can be applied to arrays of different sizes is probably the greatest weakness of Pascal. The ability to compute array sizes at run time is also desirable. Parameterized types provide a mechanism for supplying these capabilities [2,3,4].

Example:

```
TYPE STRING (LEN: INTEGER) = ARRAY [1..LEN] OF CHAR;  
VAR S: STRING (10);  
T: STRING (N+3);
```

S is an array of 10 characters, while T has N+3 characters. Type parameters may only appear as array bounds or parameters to component types. Assuming the example above, the following notations can be used:

```
S refers to the entire string.  
S[2] refers to the second element.  
S.LEN refers to the length parameter.
```

Whenever the type name STRING is used, the parameter must be supplied. There are two exceptions to this rule:

```
TYPE SP = ↑ STRING;  
PROCEDURE P (VAR STR: STRING);
```

When a parameterized type is the subject of a pointer, or used as a VAR parameter, type parameters cannot be provided.

Wirth has suggested that dynamic arrays should not be allowed within structures [PUGN #8]. With parameterized types, the weaker condition below will still permit a simple and efficient implementation.

When a parameterized type is a component of a structured type, its parameters may only be constants or parameters of the containing type.

The following implementation may be used:

1. The first field is the size of the entire data item.

This allows simple implementation of moves and compares.

2. The parameters are stored next. These fields may be interrogated or used for type compatibility checks.
3. The data area for variable sized fields is placed at the end of the record structure. This allows direct access to all fields without the need for computing displacements at run time.
4. Each field of variable size is replaced in the structure by a pointer to the appropriate data area. Unlike other pointers in Pascal, the field pointer is relative: it is the displacement of the data area from the pointer field. This allows record items to be moved without changing the pointers.
5. When an array has a component of variable size, the length and index origin are stored in a dope vector.

One additional restriction is that parameterized types are not compatible with simple types. For example, if we have the declarations

```
VAR V: STRING (12);
    W: ARRAY [1..12] OF CHAR;
PROCEDURE Q (VAR STR: STRING);
```

the call Q(V) is legal, but the call Q(W) is not.

Modules

One of the strengths of Fortran is the ability to have library packages. Some Pascal compilers allow separate compilation, so that libraries can contain individual procedures. But for a package of procedures, there must be a shared data area for retaining values and communicating from one procedure to another. The Fortran mechanism for this process is "named common". Pascal has no such mechanism. Its global data area serves as a structured form of "blank common", but there should be a structured form of named common.

Modules in the language Modula [5] provide such a structure. Selected constants, types, variables, and procedures of a module are known to the outside world; all others are the private property of the module.

Pascal should allow separate compilation of modules. Later compilations could reference the "object module", which would contain definitions of all external names. The compiler could include the module's object code with the current compilation, or could insert the appropriate loader control lines for fetching the module. Some provision should be made for alerting users when the external specifications of library modules are modified.

Files

A number of complaints have been made about Pascal's I/O

facilities [PUGN #5 (Eisenberg), #6 (Haqerty, Cichelli), #8 (Sale)]. The suggestions in this section and the two that follow address these complaints and related topics.

What is a file? A file is a record, one of whose components is a system object. The system object contains a sequence of typed components. Pascal over-simplifies this situation by ignoring the system object. When a user first learns about files, he must learn the system conventions quite early. Hiding them in Pascal does not shorten the learning time significantly.

Files in Pascal contain the one important property which makes them variables: they vary. The assignment operator is not defined for this data type. Euclid even has a method for assignment to be illegal for selected user types. But assignment of files in Pascal could be given a meaning. Just as "F↑" suggests a dereference which does not necessarily occur, the assignment "F1 := F2" could assign the system object from one file variable to another. After this assignment, F1 and F2 would access the same system object. (The coordination between these files, the contents of their buffers and flags, and other issues would need to be decided if this interpretation is made. These issue must be pursued anyway since two external files may access the same system object.)

Here are some suggested operations on files. Some of these would be meaningless for the current Pascal definition of files.

- DEFINED (file)
 - Returns TRUE if there is a system object associated with this file variable.
- OPENED (file)
 - Returns TRUE if the file has been used. If FALSE, a RESET or REWRITE should be performed rather than a GET or PUT.
- RESET (file, name); REWRITE (file, name)
 - Changes the system object associated with the file to the named object. This allows the program to reference catalogged files without the user supplying the name each time the program is run. Conventions for names would need to be established for portability.
- EXTEND (file); EXTEND (file, name)
 - These work like REWRITE, but can be used to extend an existing file without reading it first. (Example: A compiler might wish to keep a statistics file of all error messages issued to users. The second form of EXTEND could be used, since the user would not know about this file.)
- ENDFILE (file)
 - Assuming EOF(file) is TRUE, this procedure ends the current file. If EOF(file) is FALSE, skip past the end of the file. This is useful if the actual file is a tape.

Random access facilities may be added to the existing file facilities with little cost. Unlike the "slow array" proposals, the technique below does not permit integer indexes into the file. While this may be inconvenient, it can be easily

implemented on most machines. In addition, if a file element is deleted in an integer indexing scheme, there would be a question as to whether the subsequent elements should be renumbered.

POSITION (file)
 This function returns a value of an internal data type "filepos", indicating the current position of the window in the file.

DELETE (file, pos)
 Deletes an element of a file. "pos" must have type "filepos".

REPOSITION (file, pos)
 Repositions the file window to the parameter position. Following this position, a GET or a PUT is permitted, regardless of the current value of EOF(file). If the component at the indicated position has been deleted, a GET will set EOF(file) TRUE.

NEXTPOS (file, pos)
 This procedure differs from REPOSITION in two ways:

1. If the component has been deleted, a GET will access the next component which has not been deleted.
2. A PUT will place the current buffer into the file position immediately following the indicated position.

FIRSTPOS (file); LASTPOS (file)
 These functions return the first and last positions of a file, respectively.

These functions should be easy to implement for files having a fixed component size. A clarification of the effects of these procedures would be needed for text files. (Our system does deletions and rewriting of entire lines, rather than individual characters.)

As to the usefulness of this scheme, most random accesses require the use of a directory, links in fields, or remembering a location. The special pointers can be stored as easily as integers. Hashing to a file isn't possible, but a procedure for mapping integers into file locations might be specified.

Formatting

The Pascal Report [1] does not specify the effects of the formatting procedures READ and WRITE in sufficient detail. The suggestions below clarify a few points, and present some extensions and changes which seem more consistent and flexible than the current conventions.

1. What is the effect of WRITE (1234:3)? To avoid erroneous printing of numbers, the full value "1234" should be printed. This has the desirable side effect that WRITE (I:0) produces free-form output. No extra spaces are placed around the number, so it will appear naturally in the text.
2. What is the effect of WRITE ('ABCD':3)? Consistency says that the output should be "ABCD". We avoided

losing significant information in the integer case, so we should avoid it in the character case. The statement WRITE ('ABCD ':3) could be defined to produce "ABCD" instead of "ABCD ". This would allow the form WRITE (A:0) to be used inside of messages without leaving irrelevant spaces.

3. There are certain applications where the programmer may wish to truncate integers or strings on output. The statement

```
WRITE (1234:3:TRUNC,'ABCD':3:TRUNC)
```

could produce the output "234ABC".

4. Real number formatting in Pascal is currently inadequate. When the decimal digits aren't specified, the format ("E" or "F") should be chosen by the output routine to maximize the number of significant digits which can be contained in the field. The call WRITE (r:w:d) causes "F" format output with "d" decimal digits, unless significant digits would be lost on the left. The field could be expanded, but a switch to "E" format would be appropriate. (The field would be widened to allow for a value having the specified number of decimal places and the exponent as well.) The statement WRITE (r:w:d:E) could force "E" format with the specified number of digits.

For consistency with the previous cases, the call WRITE (r:0) could output in free-format. The conventions could be those found in APL, in which trailing zeros are eliminated when possible.

For consistency with the source language, at least one digit must be printed to the left of the decimal point. If no digits are requested to the right of the decimal, "." is not printed. An additional convention may be that exact zeros are printed "0" with no decimal part and no exponent.

5. Formatted output of scalars and characters should be provided, and should be identical to string output.
6. Input of character strings should be allowed. READ (s) would start at the current input position and fill s with characters. If the end of line is reached before s is full, the remainder of s is filled with blanks, and the window remains at the end of line. If EOLN is TRUE when the read is started, the read begins on the next line. (Note that if zero length lines are permitted, a null line could pass unnoticed. Our system returns one space with EOLN false even if the original line was empty.)
7. It should be possible to associate read and write formatting routines with each user-defined type. This would allow all variables to be used in READ and WRITE statements.

Input formatting, while not discussed above, should be provided. People won't use Pascal if they can't read their existing data, whether or not it is archaic. Sometimes data is generated by laboratory equipment as strings of digits; modifications to the hardware should not be forced on the user. In addition, most users don't have the time to write their own

number conversion routines for reading the data a character at a time.

The use of Fortran carriage control is archaic, and doesn't interface well with some operating systems. (i.e.: Pascal would need to know whether the first character is carriage control in order to generate the proper file format.) The following format generators could be included as arguments to READ and WRITE:

```
EOL          Start a new line.
SKIP (n)     Perform EOL n times.
SPACE (n)    Leave n spaces (skip n columns).
TAB (n)      Skip to column n.
PAGE         Start a new page (WRITE only).
CONTROL (x)  Implementation defined control.
```

The function LINELENGTH (file) could return the length of the current line. For input on text (and array of char) files, this can be useful if the number of trailing blanks is significant. For output on text files, the result is the current column.

The current handling of end of file conditions leads to errors. A user who is reading pairs of numbers might write READ (I, J), and then check for end of file. Since this is equivalent to READ(I); READ(J), and EOF condition in the first read would cause a run time error in the second, before the user could test for the condition. An EOF within a single READ statement should cause an exit from the entire statement.

Interactive I/O

Using Pascal in an interactive setting can be quite irritating. Since INPUT \uparrow is the first character of the input, a read must have been performed before a prompt message could be printed. The user should be required to write RESET (INPUT) just as he does for other files. (But correct performance on the first GET or PUT could be implemented if the RESET or REWRITE is omitted.) The initial status of INPUT should be: INPUT \uparrow = ' ', EOFN = TRUE, and EOF = FALSE.

For interactive users, READLN is useless. It encourages programmers to read the next line before issuing the prompt message. If "EOL" is allowed as an argument to READ, it should be explicitly coded and READLN should be discarded. It then becomes obvious whether the line is read before or after the data. With the EOF convention mentioned above, READ (EOL, I) would allow the user to test for EOF without causing an end of file error.

A user should be able to detect whether the program is communicating with a terminal or a data file. The function TERMINAL (file) could return a Boolean value indicating whether a specific file is a terminal. This would be more useful than a function indicating whether the run was started from a terminal, as a terminal user may wish to run the program with stored data. (The other function has its uses as well.)

Conclusions

Part I has presented three major areas of Pascal which would benefit from change. While the nature and basic structure of the language would be preserved, it would no longer be compatible with the current language. If these suggestions were incorporated into a language, perhaps we would need to call it "Pascal II": a new language based on Pascal. Part II of this paper presents additional changes, dealing more with syntactic and semantic details.

Suggested Extensions to Pascal Part II

R. A. Fraley
University of British Columbia
August 25, 1977

This part suggests smaller changes than those suggested in Part I. In many cases, minimal changes would be needed to existing compilers; a number of extensions would also be needed.

Typed Values

Values of arbitrary types should be constructable. CHR currently constructs values of type CHAR from integers; such a function is needed for other types. Scalar types could be produced from integers by using the type name as a generator.

Example

```
TYPE COLOR = (RED, ORANGE, YELLOW);
COLOR (1) = ORANGE
CHAR (ORD ('A')) = 'A'
```

A similar construction could be used for arrays.

```
TYPE VEC = ARRAY [1..10] OF INTEGER;
VAR V : VEC;
V := VEC (1, 12, N, 7 REP 0);
```

The operator REP replicates its right operand the number of times indicated by its constant left operand.

A similar notation should replace the current set constructor. Consider the type declaration and set below:

```
TYPE SS = SET OF 1..10;
[3, N, TRUNC (EXP (R))]
```

Does the set belong to the type SS? There is no way to determine whether this was the programmer's intent. If an implementation uses different representations for different sizes of sets, it would need to use its maximal representation

for the set shown. Changing the notation to `SS[3, N, TRUNC(EXP(R))]` would enable the compiler to select the appropriate set size.

A similar constructor could be offered for records. Because such a constructor depends on the field ordering, it could be "dangerous". Listing the field names could be tedious. Other suggestions are welcomed.

Initialization of Variables

A variable initialization facility for Pascal would be useful. A number of implementations already have a "VALUE" section. The utility of this facility is reduced by the presence of typed value constructors, but some sort of initialization should be made available.

A module facility was suggested in Part I. Each module effectively has a static area of shared data. While a module specification could require a call to an initialization routine, better reliability could be obtained by either initial value declarations, or by generating an initialization call when the module is included.

The mechanism below could serve as an alternative to the value section.

```
TYPE INT = INTEGER INIT 0;
     PTR = ↑NODE INIT NIL;
```

Automatic variable initialization, when declared as part of the type, could be extremely useful in building special purpose packages. Where failure to initialize a variable could lead to arbitrary results, predictable results will occur with initialized values. The overhead for initialization is only incurred when specifically requested.

Value References

Pascal's data referencing facilities are tied too closely to the syntax of the value description. One could distinguish atom (constant, variable, or expression) from type. A data referencing operator should be independent of form.

Here are some examples. Given `CONST C='ABC'`, Pascal does not allow reference to `C[1]`, even though C is an array of characters. Let `PF` be a function which returns a pointer value. Pascal does not allow the notation `PF↑`. It may be useful to refer to this function on the left of an assignment: `PF↑.DATA := 17`.

A function should be able to return any (assignable) type of data. There should be no difficulty in making such an implementation: a pointer to the result could be passed between the routines.

The technique for specifying function results is quite clumsy. As the function name is used on the left of the

assignment to receive the result value, a new user is tempted to refer to it on the right as well. This is interpreted as a recursive call. An alternative might be to use a reserved word, like `RESULT`, to specify the function result. This could also be used on the right to reference that value. Another approach would allow the programmer to specify the result name:

```
FUNCTION F (X: INTEGER) Y: REAL;
```

Record Variants

Record variants are unsafe. It is difficult to determine whether a field reference is valid [6]. Without such an ability, garbage collection cannot be implemented safely, and range checking cannot assume that the field contains the declared range.

Simula [7] provides a safe variant method using class prefixes. Translating into Pascal terminology, a prefixed record defines a new variant for an existing record. Effectively, the new record is a variant to the prefix record. The advantage is that a type defined within a library package could be extended by the user's program. A collection of library routines could provide functions over a structured type without knowing the format of data being stored within that type. For example, a library procedure might define a routine `TRAVERSE` over a data type `TNODE`. The user program could use `TNODE` as a record prefix, adding its own data to the nodes of the tree. The disadvantage of Simula prefixes is that new names must be provided for each variant; variants cannot be indexed.

Another safe method is type unions, as defined in Algol 68 [8]. These have additional uses as well.

No matter what method is used to define variants, a statement resembling Simula's `INSPECT` should be provided. This statement examines the tag, and only allows a reference to a variant field when the tag contains the proper value.

A type escape mechanism is appropriate to replace this obscure use (or abuse) of record variants. For example,

```
I := 'AECD' TYPE INTEGER
```

could assign the representation of 'AECD' to the integer I. Such a statement is useful for computing hash values. The statement is clearly implementation dependent, and should be used with restraint.

Packing

The restrictions on the use of packed data are a blemish on Pascal. They are not obvious to the user who is ignorant of implementation problems. Consider the segment below:

```
TYPE T = 1..10;
VAR V1 : T;
     V2 : PACKED RECORD
         F1, F2, F3: T
     END;
```

V1 and F1 are declared to have the same type, yet they cannot be used interchangeably. In particular, they may not both be used as VAR parameters. Here is a modified notation for specifying packing.

```

TYPE  T1 = 1..10;
      T2 = PACKED 1..10;
VAR   V1 : T1;
      V2 : RECORD
          F1, F2: T2;  F3: T1
      END;

```

Internal representation differences require that the programmer code a different type. It is now obvious that his data items are not interchangeable. Because packing is not associated with the environment, not all fields need be packed. Unfortunately, this notation would be more verbose than the current notation.

Control Structures

To hasten the learning process, there should be a uniform structure for compound statements. The "closed" format used by Algol 68, Modula, and Euclid is most desirable. This format uses a closing word for each statement, rather than requiring BEGIN...END brackets to build compound statements.

Example:

```

IF B THEN X := Y; Y := Z END_IF

```

(Other formulations use "FI", "END IF", or "END" in place of "END_IF".) The advantage to this form is that the programmer need not decide whether a BEGIN is needed or not when writing his code. A line may be inserted without needing to add BEGIN END brackets. The "dangling ELSE" problem disappears: it is always obvious which IF an ELSE is associated with.

Most of Pascal's compound statements may easily be given a closed form: a suitable closing word is selected for the end. For convenience, "ELIF" (a contraction of ELSE and IF) is added to the IF statement, allowing multiple tests with only one closing END_IF. The REPEAT statement already has a closed format. The CASE statement requires the most modification to meet this suggestion. The format below might be used:

```

CASE I
WHEN 1,2 DO X := Y; Y := Z
WHEN 3  DO X := Z; Y := X
END

```

The word "CASE" might be replaced by "TEST" for readability.

The CASE Statement

The CASE statement, as specified by the report, has a number of weaknesses. The legality of the statement below is not specified by the report.

```

CASE I OF
  1: X := Y;
  10000: Y := X
END

```

Some compilers might try to generate a branch table having 10000 entries! (Pascal/UBC generates a table search.)

A default case is sorely needed for writing fail-soft programs. Such programs need to intercept errors due to unexpected data values. Pascal/UBC uses the following syntax:

```

CASE C OF
  '<', ':': PROCESS (C);
  '<>', ':': ERROR (C)
END

```

This has an advantage over the "OTHEWISE" notation [PUGN #8, pg 26 (Steensgaard-Madsen)] because specific error values can be listed along with the default symbol "<>".

Wirth has rejected the use of a CASE default [PUGN #8, pg. 23]. This author feels that it is no worse than the unqualified ELSE in an IF statement. It is an absolute requirement if the CASE is to be useful in real programs. Consider Wirth's example of a CASE being used with a character field. A program which will receive wide distribution may use a restricted character set. All characters cannot be listed, because implementation character sets may differ. On certain machines, some characters have no external representation, and could not be entered as case labels. Sometimes the internal, printer, and terminal character sets differ, adding to the confusion.

To restrict the temptation to misuse the case default, a range of values (such as 1..10) should be allowed as a case label. For characters, the ASCII ordering could be assumed to permit portability. ('..'@' means "all characters from ' to '@' in the ASCII sequence". It does not imply that the internal representation is ASCII. Compilers for non-ASCII implementations would need a table to select the appropriate character values for the case label.)

Loop Statements

The existing loop statements are not applicable in all situations. The author suggests an infinite loop, together with an exit statement.

```

LOOP
  EXIT IF F(I) < 10;
  I := I + 1;
END

```

A semantics check could guarantee that each loop contains an EXIT. The EXIT can be embedded in an IF or CASE, but not another loop. The condition following EXIT is optional. (Compilers should generate a table of all exits from a loop, or flag exit statements in the margin.) A FOR clause could introduce a loop:

```

FOR I := 1 TO 10 LOOP ... END

```

For loops would be more useful if a set value could control the iteration.

```

FOR I IN [1, 7, 9..N] ...

```

No specific order of value selection should be assumed. Another improvement to FOR loops (as modified above) would be a code region which is executed only when the iteration terminates normally, but not when an EXIT is used to leave the loop.

```
FOR I := 1 TO LIMIT LOOP
  IF KEY = TABLE [I] THEN
    BEGIN RESULT := I; EXIT END
AFTERWARDS
  LIMIT := LIMIT + 1; TABLE[LIMIT] := KEY;
  RESULT := LIMIT
END
```

The value of the controlled variable should be defined on exit. Many machines cannot trap references to undefined values, so implementation dependent programs could be developed which rely on the specific values left by that implementation. The variable should have the value of the last time through the loop, or be unchanged if the loop was not executed.

Miscellaneous Points

- Allow "_" in identifiers.
- Allow ";" before ELSE (optionally). Programmers could then use ";" as a statement terminator if they wish.
- Allow strings to be extended with blanks on the right. Do not automatically truncate characters from strings. Allow single characters to be extended to strings.
- Provide a substring function. Alternatively, allow a number range as a subscript.
- Explicitly prohibit character constants from being continued on a new line.
- Allow subranges instead of type identifiers in formal parameter lists.
- Change ":" to "," in procedure parameter declarations.
- Require parameter types for procedures passed as parameters.

```
PROCEDURE Q (PROCEDURE P (VAR REAL; 3 REP INTEGER));
```

- Procedure argument lists are a specialized form of record. The variant mechanism should be available for parameter lists.
- Allow arbitrary type specifications in pointer declarations.

```
TYPE PTR = ↑ RECORD ... END
```

(Also allow PTR to be referenced within the record.)

- Allow arbitrary scalar types in case tag field

declarations. Require a ":" if the tag name is omitted.

- Allow "PROCEDURE" in place of "FUNCTION".
- Specify a maximum source line length. 100 is a reasonable value. (This allows some extra formatting space on a 132 character print line, and allows for line expansion when 80 character lines are edited.)
- Provide procedure variables. A representation of procedure pointers is available for procedure parameters. Only top-level procedures (within a module) may be assigned to variables, thus avoiding the "vanishing environment" problem.
- Use parentheses for array references (A[I]) instead of A[I]). An array is a method for implementing a function of integer arguments. Why should implementation decisions affect the technique for writing the function reference? A library package might change its implementation, without changing its external specifications; users would then need to change their programs. (Note: there is currently no distinction between a variable reference and the invocation of a parameterless function.)
- Provide a compiler option format, instead of examining comments for compiler options. "\$" brackets could surround option specifications. Each option name may optionally be followed by "+", "-", or "=<constant>". Options are separated by "," or ";".
- Compiler options should be standardized. An implementation need not provide an option, but if it does, the name is chosen from the standard list. Options not on the list may be provided as well. Compilers not recognizing an option name should ignore it.
- Compiler options which affect the generated code (such as options for range and index checks, non-standard features, optimization, or debug facilities) should follow standard nesting rules. Option changes made within a procedure should not affect outer procedures.

Semantic Ambiguities

The validity and interpretation of each construct below should be made explicit in any forthcoming standard.

1. Consider the program segment below.

```
PROCEDURE P;
  TYPE T = ↑XX;
  XX = RECORD ... END;
  ...
END;
```

T obviously points to a record object of type XX. But what if the debugged procedure P is placed in a surrounding program which also contains a type XX. Using many compilers, T is now a pointer to the outer XX. Is this interpretation proper?

2. Our compiler will process the declaration below without any error messages.

```

TYPE COMPLEX = RECORD
    REAL: REAL;
    IMAG: REAL
END;

```

Should it produce an error message? Maybe two?

3. Exact rules for type compatibility should be stated. We recently received a program which passed a 0..100 variable as a VAR INTEGER parameter. Should this be legal? If so, how can valid range checks be performed? How about two record declarations having identical field structures (and names)?
4. Our compiler used to loop when it encountered the program below.

```

TYPE A = ↑A; B = ↑B;
VAR X: A; Y: B;
BEGIN X := Y END.

```

Are these types legal? Does your compiler generate the error message properly?

Conclusions

Extensive changes to Pascal have been suggested in this paper. The author feels that changes of this magnitude are required if Pascal is to be useable for program development in a professional setting. Quite a number of additional features could be recommended; the changes above have the highest priority.

If Pascal was modified as described, we may need to call it something else, for it would no longer be Pascal. Perhaps it is Pascal II? Or Pascal 77? A standard language resembling Pascal is surely needed, but why should hundreds of programmers be involved in writing a standard, new processors, and textbooks if the standardized language has a limited life or cannot be used for large systems?

Do not condemn Fortran without understanding it. It has changed over the years to provide improved facilities for program development, even if its basic structure has remained static. Each feature, as used in program writing, should have an equivalent in Pascal, or good reason should be given as to why it is not. Fortran owes its long life partly to its standard, even though it has not been strictly implemented. But it also owes its life to its usefulness in solving many of the every-day tasks of programming, such as communication with the operating system and referencing library packages. Pascal would benefit from expansion in some of these practical directions.

Bibliography

- [1] Jensen, K. and N. Wirth, Pascal User Manual and Report, Springer-Verlag New York (1976).
- [2] Lampson, B. W. et al., "Report on the Programming

Language Euclid", SIGPLAN Notices, 12:2, Feb 1977.

- [3] Wirth, N., "An assessment of the Programming Language Pascal", SIGPLAN Notices, July 1975.
- [4] Fraley, R., "Parameterized Types", Unpublished Paper, April 1975.
- [5] Wirth, n., "MODULA: A Language for Modular Multiprogramm
- [6] Fischer, C. N. and R. J. LeBlanc, "Efficient Implementation and Optimization of Run-Time Checking in Pascal", Proc. ACM Conf. on Lang Design for Reliable Software, Raleigh, N.C., March 1977.
- [7] Dahl, O. et al., Common Base Language, Norwegian Comp. Ctr., Oslo, 1970.
- [8] vanWijnngaarden, A. et al., "Revised Report on the Algorithmic Language Algol 68", SIGPLAN Notices, 12:5, 1977.

(* Received 77/09/13 *)

WHAT TO DO AFTER A WHILE

D. W. Barron

& J. M. Mullins

Computer Studies Group,
University of Southampton.

The Problem

The problem is simple. If one writes while p and q do ..., will q be evaluated if p is false? The "boolean operator" approach says yes, the "sequential conjunction" approach says no. The *Revised Report* leaves the question open (though the *User Manual* says "yes!"), and A.H.J. Sale (1) has argued strongly that the boolean operator approach should be uniformly enforced. We take the opposite view.

A Proposal

We propose that in the forthcoming revision of the Revised Report the operators and and or should be defined as

p and q \equiv (if p then q else false)

p or q \equiv (if p then true else q) *

The reasons in favour of this proposition are summarised here for the convenience of readers who may not wish to study the detailed argument in the remainder of this paper.

- i) Sequential conjunction permits a programming style that is more in the spirit of PASCAL.
- ii) If sequential conjunction is adopted as the standard, existing programs that assume the "boolean operator" interpretation will continue to work. The converse is not true. (This argument does not hold for programs that rely on side-effects of functions, but we have no sympathy whatever for those who perpetrate such monstrosities.)
- iii) If sequential conjunction is adopted as the standard the effect of the boolean operator interpretation can be fabricated by user-defined functions: the converse is not true.

Programming style and the "spirit of PASCAL"

The argument in favour of sequential conjunction hinges on programming style. Efficiency is sometimes raised as a consideration but it is a red herring. For some architectures (e.g. pipelines) sequential conjunction is inefficient: for other architectures it is more efficient. But in the Pascal community we should have gotten beyond judging language features solely in terms of implementation efficiency. What matters is being able to write *correct* programs that are easily comprehensible.

One of the classic illustrations of the two approaches is the problem of searching a table:

```

var table : array [1..maxsize] of whatever;
:
:
index := 1;

while (index <= maxsize) and (table [index] <> item)
do index := index + 1 ;

(*condition for item not found is "index > maxsize"*)

```

* The first of these relations defines sequential conjunction. The second defines sequential disjunction, but we shall use the term "sequential conjunction" to include both.

This is a natural way of expressing the operation to be carried out; what follows the while is a sequence of guards* which determine whether or not the body of the loop is to be executed. If we specify that the guards are inspected in strict left-to-right order until one of them "triggers" there are no problems: this corresponds to evaluating the while condition by sequential conjunction. However, if the boolean operator approach is used we are in trouble (array bounds) if the item sought is not in the table. OK, there are lots of ways round this, (e.g. put the item sought at the end of the table, use a boolean variable, or even use a goto), but as we illustrate in the appendix, these are distortions: the program no longer bears a simple and obvious relationship to the problem, and this is bad style. Our contention is that the need to distort the program to satisfy the boolean operator interpretation is contrary to the spirit of Pascal. Study the examples in the appendix to see the truth of this assertion.

Uniformity and Regularity

Our argument has concentrated on while statements (and similar arguments could be applied to repeat..until constructions). This is because these are the only constructions where the sequential conjunction interpretation is required and cannot be fabricated by other means. The question arises whether all occurrences of and and or should be treated the same way. Sequential conjunction should certainly be used for conditional statements. If the boolean operator approach is used, we can fabricate the effect of sequential conjunction by writing

```

if (p and q) then S1 else S2
as
  if p then begin
    if q then S1 else S2
  end
else S2 ;

```

but this is not perspicuous, and is not in the spirit of PASCAL.

What should happen to boolean expressions in other contexts e.g. expressions? If we were starting the language design again, there would be quite strong arguments for allowing the boolean operator interpretation in these other contexts. It is a good principle of language design to separate clearly the data objects and the permitted operations on them from the mechanisms for achieving flow of control. Sale's paper distinguishes between boolean expressions in a jump context and in a value context. He is recognising (unconsciously perhaps), that whilst boolean variables are data objects, and can be combined into boolean expressions, the condition following an if or a while is something different. To emphasise the difference, we might give it a different name; taking a term from English grammar⁽³⁾ (an obscure subject, rarely studied these days), we could call it a protasis. Because it is a different kind of object it is possible for it to obey different evaluation rules, and sequential conjunction is the right approach. Equally, because it is a different kind of animal we would need a new notation e.g. and, or for sequential conjunction,

* Compare Dijkstra's if and do constructions expounded in "A Discipline of Programming".⁽²⁾

booland, boolor for boolean operator. This is the solution adopted by some other languages e.g. POP-2⁽⁴⁾, RTL/2⁽⁵⁾. But given that we have a fairly well defined language already that we *don't* want to change, it seems better to make sequential conjunction uniformly the method to be used for evaluation of and and or in all cases. This has the supreme advantage that existing programs (and particularly compilers) continue to work, and don't have to be changed to conform to a new standard. It also happens to be the rule adopted by MODULA and EUCLID.

Before leaving this topic it is worth remarking that the dangers of confusing a protasis with a boolean expression are well illustrated in Algol 68. The pursuit of regularity, orthogonality, etc. allows us to write

```
while S1; B do S2;
```

meaning

```
L: S1; if B then S2; goto L;
```

which is thoroughly confusing.

Getting the best of both worlds

Adopting the sequential conjunction interpretation allows a more natural style of programming. (Compare Welsh's compiler with Amman's to see this illustrated vividly.). If anyone really wants to enforce the boolean operator interpretation of and he has only to define

```
function andop (p,q : boolean) : boolean;
```

```
var p1,q1 : boolean;
```

```
begin
```

```
p1 := p; q1 := q; (*ensures both arguments evaluated*)
```

```
andop := p1 and q1
```

```
end;
```

then he can replace while(p and q) by while(andop(p,q)) and be assured that p,q will both be evaluated.

References

1. Sale, A.H.J. Compiling Boolean Expressions - the case for a "boolean operator" interpretation. PUG Newsletter No. 11, 1977.
2. Dijkstra, E.J. A Discipline of Programming. Prentice-Hall, Englewood Cliffs, 1976.
3. Onions, C.T. Modern English Syntax. Routledge and Kegan Paul London 1971.
4. Burstall, R.M., Collins, J.S. and Popplestone, R.J. Programming in POP-2. Edinburgh University Press 1971.
5. Barnes, J.P. RTL/2 Design and Implementation. Hayden and Son, London 1977.

Appendix

Searching a list without sequential conjunction

1. INSERT A DUMMY ITEM

```
var table : array [1..maxsizeplusone] of whatever;
:
:
index := 1;

table [maxsizeplusone] := item;

while (table[index]<>item)
    do index := index + 1;

(*condition for item not found is "index=maxsizeplusone"*)
```

2. USE A BOOLEAN VARIABLE

```
var table : array [1..maxsize] of whatever;
    notfound : boolean;
:
:
index := 1;
notfound := true;

while notfound and (index <= maxsize)
    do if table[index] = item
        then notfound = false
        else index := index + 1;

(*condition for item not found is "notfound"*)
```

3. USE A GOTO

```
label 9;

var table : array [1..maxsize] of whatever;
:
:
index := 1;

while(index <= maxsize)
    do if table[index] = item then goto 9
        else index := index + 1;

9: (*condition for item not found is "index > maxsize"*)
```

(* Received 77/09/21 *)

ADAPTING PASCAL FOR THE PDP 11/45

D. D. Miller
GTE Sylvania, Inc
Mountain View, California

ABSTRACT

This paper describes our adaptation of the University of Illinois' PASCAL student compiler for a PDP 11/20, to a production compiler on an 11/45. We will discuss, a) the extensions to the language which were necessary to communicate between PASCAL programs, data and MACRO-11 code, b) support routines such as a routine debug and source update and reformatting, and c) how we introduced PASCAL into an existing software system and to MACRO programmers.

INTRODUCTION

Historically GTE Sylvania has delivered turnkey systems with software written in machine language. For all the usual reasons management decided to investigate the possibility of developing software based on a higher level language. The target project was a system which was already in the field, but under contract for a major revision. The system controls a real time experiment. It has several special purpose I/O devices and doesn't use any DEC software. It is about 25,000 lines of macro-11 code and runs on a PDP 11/45 with 64K of memory.

In March 1975 we attempted to locate compilers other than FORTRAN which were running on a 11-series machine. Candidate compilers had to be able to handle complex data structures with ease and would have to generate LINKable code so we could build our own system. The only compiler we could find to meet these criteria was PASCAL at the University of Illinois. It was running on an 11/20 under a version of DOS-4.

We began in-house training for PASCAL using the Jensen and Wirth User's Manual¹ in April. The legal aspects of acquisition were completed in the meantime and the Compiler was ready for use in May.

We then wrote some toy problems to get a better feeling for PASCAL and to test its non-standard features. In June we identified the changes we would have to make and those we would like to make, and then laid out a schedule.

The production compiler was released to the programming staff in November so that they could begin compilation. It was completed in December according to our requirements. But we continued to modify it as we saw the need.

We will discuss the specific changes we had to make to the compiler, a description of the supporting software and the non-technical problems we encountered introducing the compiler to the staff.

CHANGES TO PASCAL

The changes outlined in this section were made because the several programmers are all working on a single system. The system as originally designed, had a large common data base which was accessed by all programs. And the system contained several utility programs. Furthermore, from a practical level, the routines in the system are better developed and maintained in smaller units. A fifty page PROGRAM is more difficult to develop and maintain than five ten page PROGRAMS. A fifty page PROGRAM may also exceed the compilers table limits.

Program Cross Linkage

PASCAL does not allow linkage to data outside the PROGRAM and only allows linkage to predefine external utility programs. This was unacceptable as we expected to make use of existing MACRO code and data. Therefore the major change to the compiler was to implement the following syntax:

```
<program>      ::= <prog>. | <module>.  
<prog>         ::= <program heading>  
                <program block>  
<program heading> ::= program <identifier>;  
<module>       ::= <module heading>  
                <module block>  
<module heading> ::= module <identifier>;  
<module block>  ::= <declaration part>  
<program block> ::= <declaration part>  
                <statement part>
```

A program is an executable entry, i. e., it contains outer block code and has a start point. A module

is a non-executable entity, i. e., it does not contain outer block code.

A module is composed of data and procedure declarations only. Modules allow for portions of the system to be separately programmed and compiled. Selected data elements and/or procedures and functions can be accessed from other modules or from the program via the external and entry declarations. Data spaces consist of the following types:

normal
external
own

Normal data space is the same as Jensen described. Normal data space is allocated on the "stack" upon entry to a procedure and is deallocated (erased, lost) upon exit from the procedure. Each entry to a procedure which declares normal data structures causes space to be reallocated to those data structures. Consequently, no memory of previous values is retained, and the same stack space may be shared among procedures.

external data space consists of those data structures that are used locally but are allocated in another module or MACRO program and have been declared as either entry points or global. The usual declarations are used to describe these data structures, but no space is allocated for them.

own data space consists of those variables that are allocated in a program or a module but are not put on the "stack." own data structures retain their values from procedure entry to procedure entry. own data structures may be initialized and/or may be designated as entry points. Entry points are used by LINK to resolve references created by use of EXTERNAL data spaces in other modules.

Example:

```

program LISTEN;
  external
    var TALK: CHAR;
    procedure SPIT (ANSWER: CHAR);
    procedure ASK;
  end external;
begin
  while TALK = ' ' do ASK;
  SPIT (TALK);
end.
module SAYSO;
  own
    TALK: CHAR;
  initial TALK: = ' ';
  procedure ASK;
  begin READ (TALK) end;
  procedure SPIT (X: CHAR);
  begin WRITE (X) end;
  entry TALK, ASK, SPIT.

```

In this example the main, or controlling program is program LISTEN. Within LISTEN the procedures SPIT and ASK are declared external, indicating that these two procedures may be invoked but are not defined within this program. The assumption is that some other body of source code defines these two procedures appropriately. The variable TALK is also defined as external to the program.

The module SAYSO satisfies all of the external references made by LISTEN. Note that the variable TALK is declared in own space (a necessity for entry points) and is initialized to ' '. TALK, ASK, and SPIT are declared as entry points, i. e., they may be referenced from other modules or programs as externals.

Packed Records

The Illinois PASCAL didn't allow packed record or array declarations. But our existing system contained a lot of packed data, which naturally fell into the packed record syntax. Since Wirth leaves the semantics of packed records to be defined at implementation time, we chose the following scheme for storage allocation:

Storage is allocated in one of two modes, normal and packed. For normal storage allocation (i. e., the variable has not been declared as packed, it is not a packed type, and is not part of a packed variable) each unstructured variable is allocated either a byte or word. A byte is allocated for scalars with less than 128 elements, for subranges within -128, +127, and for char. A full word is allocated for all other unstructured types. If the next element to be allocated is a full word and the next available byte is an old address, then that byte will be skipped and the full word will be allocated on the next higher even address. Arrays and records are allocated as a set of contiguous elements.

packed records are allocated in a slightly different manner. Each element is allocated only the number of bits necessary to contain the element. If the next such element to be allocated will fit within the current word then it will be allocated there. If the element does not fit, then it will be allocated starting at the next even byte. Allocation of elements starts at bit 0 of a word and proceeds to bit 15 (i. e., from right to left).

Examples of PACKED record:

Channel	Status	Unit
15	13 12	4 3 0

```

STATUSWORD = packed record
  UNIT: 0..15;
  STATUS: array [0..8] of boolean;
  CHANNEL: 0..7
end;

```

Variant Record

We changed the syntax of variant record descriptions slightly to ease the job for the compiler and to make the presence or absence of a tag field in the record more consistent. The semantics of this syntax means that the record may or may not contain space for the tag field. The variant record is similar to FORTRAN's equivalence statement. The syntax is

```

<variant>:= case <tagfield>: <type> of
  { <variant subfield> } end
<tagfield>:= <identifier> | empty
etc.

```

Examples:

1. With tagfield in record


```

case X:(A, B, C) of
  A:(      );
  B:(      );
  C:(      )
end

```
2. Without tagfield in record


```

case :boolean of
  true: (    );
  false: (   )
end

```

Additional Functions

We found that we needed several functions added to the compiler

Procedure MACRO (CODE:INT;X:INT); This procedure reference signals to the compiler to generate an inline word of instruction. This is the mechanism to be used if special instructions must be executed in the middle of PASCAL code. CODE must be -1, and X may be either a constant or the name of a variable. IF X is a constant, then that value is the value used. IF X is a variable, the P-relative address of the variable is used.

Example:

```

MACRO (-1, 016746B);
MACRO (-1, X);

```

Procedure TRAP (TRAPNO:INT;var SPECBLK;tbdb;var LINKBLK;tbdb) This procedure first places the addresses of SPECBLK and LINKBLK on the stack then executes a trap instruction. The trap number is indicated by the parameter TRAPNO. The code generated by the compiler for this procedure call is:

```

mov #SPECBLK, -(SP)
mov #LINKBLK, -(SP)
trap #TRAPNO

```

Function FIRST (X:INT):INT; This procedure reference signals the compiler to use the lowest value defined for the parameter X. X may be of type integer, scalar or subrange. See below for example of usage.

Function LAST (X:INT):INT; This function reference signals the compiler to use the last, or highest, value defined for parameter X. X may be of type integer, scalar, or subrange.

Example:

```
var      X : 13..21;
        Y : (A, B, C, D);
begin    X := FIRST (X);
        while (X:=X+1) <= LAST(X) do;
        for Y:=FIRST(Y) to LAST(Y) do;
```

Function ADDR (X:TYPE): TYPE; This function generates a pointer (i. e., the address) to the variable given as a parameter.

```
var      Z:integer;
        Y:↑ integer;
        .
        .
begin    Y:=ADDR(Z) end;
```

Function POINTER (X:int):↑; This function causes the integer parameter to become a universal pointer. This allows the programmers to do pointer arithmetic:

Given the following declarations:

```
type DIGITS = array [0..9] of integer;
var    X:DIGITS;
        W:DIGITS;
        Z:integer;
```

then we may write the following statements:

```
W:=addr(X);    "W points to X"
Z:=addr(X[ 2 ]);
Z:=addr(W[ 3 ]);
Z:=pointer(ord(W)+6); "Z points to X [ 3 ] or
                    W [ 3 ]"
X [ 1 ]:=ord(Z)-ord(W); "# of bytes"
If pointer (ord(addr(X))=addr(X) "true"
                    then . . . ;
```

SUPPORT ROUTINES

The University of Illinois supplied us with several debug and support routines, which we have enhanced. We have written others to support the user. The compiler writer must have access to intermediate results in order to find errors. This version

of PASCAL is seven core loads, linked together with disk files. There are two routines the compiler writer has to debug PASCAL. One prints all the intercommunication files and the other prints the resultant code in MACRO format. This second routine was originally the last pass of the compiler, and generated input to the MACRO assembler. However that final pass had to be modified to print out its table of PASCAL labels and statement numbers versus relocatable address. This was required to relate console debugging to compiler listings.

The compiler included a cross reference pass. This pass sorts all the routine variables and prints all references to that variable along with its definition. This feature is found on very few compilers but is a valuable aid to debugging and maintaining any block structured language routine.

The University of Illinois included a source deck reformatting routine. This routine is used to enforce a consistent statement indentation scheme throughout a program and across all programs. This permits all programs to be more portable between programmers. To assure that it is used regularly, we added a line editor to the front end, and embedded the whole thing as the first pass to the compiler.

NON-TECHNICAL PROBLEMS

At the outset, the project staff was composed of six programmers; five had just completed delivery of the predecessor system. The author had been hired to complete the staff. The five had little recent contact with high level languages. Most had spent their career at the machine language level. And so when it became known that management was considering a high level language for the project update their reaction ranged between disbelief and rebellion.

Of course management was quick to point out that a high level language was only being considered and that the language decision would be made in the future, based on availability and economic factors. And of course, the basic design of the project wasn't affected by the language. So design could continue independently from the language.

When the Illinois version of PASCAL was chosen as the candidate compiler, in-house classes were held, and the basics of the language explored. When the compiler actually became available, a few five line programs were attempted by the staff.

The first relevant PASCAL introduction for the five was a compilation of the system's data areas in June. This sparked some discussions (out of

curiosity) concerning some practical applications of the language to the problem that they were familiar with. Occasionally, some suggestions were made to improve the data definitions. And thus a gradual acceptance of PASCAL was beginning to be seen.

The modification of the target system included first, documenting the existing system and then upgrading it according to customer requirements. Necessarily the documentation was carried out at a high level, i. e., higher than machine language flow charts. And as the system was modified, the same high level documentation technique was used. It would be ideal to conclude at this point by saying that the documentation style closely resembled PASCAL, but that simply isn't true. Each designer developed his own style. What did take place is that they learned how to express their ideas in a language above that of the machine while still precise enough to be short of arm waving.

As the language decision date approached, the lines-of-code estimates for the two language alternatives were made. Based on those estimates alone, PASCAL was clearly the choice. However, some of the designers, looking forward to coding, were appalled by the suggestion the machine language was still being considered.

When PASCAL was finally declared to be the language, real programs began to appear. And consequently the designers' imagination crashed the compiler on a regular schedule. Syntax was added and in one case (the variant CASE) changed at the suggestions of the designer turned programmer. So finally we realized not only acceptance of PASCAL, but an effort to improve it to better describe the problem at hand.

Acknowledgments

We wish to acknowledge Professor Don B. Gillies (deceased), Ian Stocks, and Jaynt Krishnasamy of the University of Illinois for their assistance in transferring the PASCAL compiler from the PDP 11/20 to GTE Sylvania's PDP 11/45. In addition we wish to acknowledge the National Science Foundation for making the Illinois PASCAL compiler available to GTE Sylvania. Especially we must thank Larry Drews and David Shaw, whose foresight and diligent work modified this version of PASCAL.

(* Received 77/09/13 *)

- (1) Jensen, Kathleen; and Wirth, Niklaus "PASCAL User Manual and Report," Lecture Notes in Computer Science, Volume 18, Springer-Verlag, 1974

PASCAL: Standards and Extensions

A. General

In this article I would like to make some comments on the current standards/extensions argument, and to suggest some specific modifications to the standard and some useful extensions.

In my opinion the following points are relevant to the standards argument:-

(1) Any implementation of PASCAL should conform to some minimum "standard". This is the basic requirement for portability, which is so necessary if PASCAL is ever to supplant FORTRAN. If any implementation does not meet this standard it should not be called PASCAL.

(2) The standard should be based on the revised Report. However, there are areas in which the Report needs tidying up before the standard can be specified (see the contributions by Niklaus Wirth [pp. 22-24, PUGN #8], Andy Mickel [pp 28-30, op.cit.] and below).

(3) Extensions should be allowed, but they should be "conventionalised" as suggested by Andy Mickel [loc.cit.]. By conventionalised I understand that any extension to provide a specific function should be consistently supplied in only one standardised form. Possibly, only extensions that can be expressed in standard PASCAL (albeit in a possibly lengthy or inefficient manner) should be allowed (?).

(4) Any compiler that implements extensions should also be capable of processing only the standard language. This would enable any program to be checked against the standard. This could be controlled by a compiler option indicating whether the standard or the extended language was to be processed.

(5) The standard should be specified as soon as possible to restrict the current proliferation of incompatible implementations. The standard should hopefully be agreed upon by some form of consensus of the PASCAL user/implementor community. There should also be some recognised means of conventionalising any extensions.

B. On the Standard Language

I would like to briefly discuss some of the points that Andy Mickel [loc.cit] has placed in categories of standards and extensions.

(1) I think that variable extent arrays as formal parameters should be allowed as part of the standard. For many general purpose applications programs these are virtually indispensable unless the source code is to be modified each time for each specific application. (See also the discussion on varying arrays below)

(2) Some decision must be made on a standard character set upon which the type char is based.

(3) While the suggested otherwise part in a case statement would be useful I don't think it is essential. However, I do think that the standard should specify the action to be taken when a missing value is encountered, i.e. error or null operation.

(4) I would like to keep well away from formatted read statements (see below as well).

Taking into account the above consideration, I agree basically with the classification of the various points discussed by Andy Mickel. However, I do have some additional points which I would like to see discussed. These are dealt with below.

(C) Modification to the Standard for Repeat and Case Statements

Currently, multiple statements under control of a while statement are bracketed by a begin ... end pair, those under control of a repeat statement are bracketed by the repeat and the corresponding until and those controlled by a case statement by the case and end.

While admitting that these forms are reasonably straightforward, I think they are inconsistent. I consider that programming style can be much improved with little or any extra compilation cost by treating all statements controlled by a control statement in a consistent manner, i.e. by grouping with a begin ... end pair in the form of a compound statement.

To my mind the greatest advantage of this is that related statements are always grouped in the same manner and that when reading a program each end matches with a corresponding begin.

I therefore suggest that the standard syntax for the repeat statement be modified to:

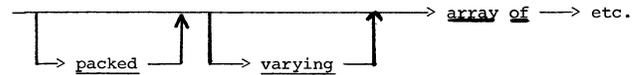
repeat → <statement> → until → <expression> → and that the syntax for the case statement be: case → <expression> → of → begin — etc.

Admittedly the repeat statement can already be used in the suggested manner, however I don't think it would be a bad thing to force programmers to use it in this way!

(D) Varying Size Arrays as an Extension

This extension to the language is prompted by a foreseen need for varying length character strings (they make the manipulation of non-numeric data so much simpler!). However in as much as a character string can be considered to be an array of char or a packed array of char, the idea can obviously be extended to any array, with the added advantage that the treatment of many problems involving the use of varying quantities of data might be simplified.

I therefore suggest the following extension to the declaration of array types:-



The declared size of the array would be treated as a maximum only, the initial size being zero.

As an example, a variable declared so:-

var S: packed varying array [10..19] of char

would be a string of maximum length 10, the first element being S[10] and the initial length being zero.

I think that such an extension would minimally require the following extensions in addition:-

(1) A concatenation operator to allow appending of one string or array on the end of another.

(2) An additional form of selector to allow the equivalent of the PL/I substring function.

For example give S above, the reference:- S[12..15] would be equivalent to a reference to a variable of type

packed array [12..15] of char.

This form of selector need not apply only to varying arrays but could apply to any arrays.

(A relevant aside: This idea would be even more useful if variables were considered to be of the same type if they had the same structure, not just if they had the same type identifier.)

(3) A size function. This would be used to return the size of any array, varying or otherwise.

(E) A General Inverse to the Ord Function

Very briefly, I think that some function comparable to the char function but acting on any scalar type (except real) as an inverse to ord would be very useful. At the moment I can't think what to call it!

(F) A Different Approach to the Treatment of File Variables

There has been much discussion on the meaning of statements such as:-

```
f1:=f2
```

where f1 and f2 are both file variables.

I think much of the discussion has arisen through some conceptual confusion between a file and the file variable used in PASCAL to reference a file.

I would like to suggest an approach to the whole subject which I think is different to those that have been put forward before, but which is both natural to PASCAL and consistent with the normal usage of file variables in PASCAL.

The notation f↑ (where f is a file variable) for an item in a file is identical to the notation p↑ for the variable referenced by the pointer variable p. Thus file variable f could be considered to be a pointer variable tied to the set of items in the file which it references. (It should be noted that this argument does not assume that an item in a file need be transferred to a file buffer before it may be used even though it may be transparently implemented in such a manner.)

Then, just as p1:=p2 (where p1 and p2 are pointer variables based on the same type) assigns to p1 a reference to the same variable as it referenced by p2, so f1:=f2 (where f1 and f2 are file variables) assigns to f1 a reference to the same item in the same file as is referenced by f2.

Consequently, the expression f1=f2 would have the value true if and only if f1 and f2 referenced the same item in the same file.

The use of file variables as formal var or value parameters is also obvious by analogy with the similar use of pointer variables.

The use of file variables in any other expressions (except as parameters to I/O procedures or when qualified by ↑ as in f↑) would be meaningless and should be treated as syntax errors.

Naturally enough, the usage of file variables in the above statements and expressions would only be legal if the base types of the relevant files were the same.

As has been mentioned by others, files can have four basically different kinds of temporal existence:-

- (a) Existence before and during program execution, but not after (e.g. file removal in a archiving system).
- (b) Existence during program execution only (i.e. a scratch work file).
- (c) Existence during and after program execution but not before (i.e. file creation).

- (d) Existence before, during and after program execution (e.g. permanent data-base file).

Clearly, in case (a), (c) and (d) the files have some existence separate from the program and the file variables used to reference them. There must be some way of associating a file variable with such a "permanent" file in such cases. This is currently done by effectively passing the files as parameters to the program and associating the file with the file variable by the file variable identifier. This seems to me to be O.K. as a start particularly if the operating system provides some method for associating external and internal file names (e.g. B6700 label equation). However, I think this method has two disadvantages. Firstly, it seems to imply that the files must exist for program execution to be successful, there is no reason why this should be so. Secondly, it lacks flexibility in that a file variable must always be associated with one particular file during program execution.

I would like to suggest that a file variable only become associated with a file when the file variable is referenced by a file reference function or procedure, e.g.:

```
eof(f),
get(f),
put(f),
reset(f),
or
rewrite(f).
```

Until a file variable is associated with a file, it should have the value nil or an undefined value (is there/should there be a difference?). A file variable could be disassociated from a file either by setting it to nil, or by using a procedure such as

```
close(f).
```

It may be considered useful to have a corresponding procedure, the only action of which is to associate a file variable with a file, e.g.

```
open(f)
```

Of the four types of file discussed above, for (a) and (b) we need to be able to indicate that the file is to be expunged from the system and that the file variable is to be disassociated from the file. For cases (c) and (d) we need to be able to disassociate the file variable but leave the file known to the system. May be a procedure

```
delete(f)
```

or something similar, could be used to perform the first of these functions and let the suggested procedure close perform the second. (What should the default action be?)

When performing output to a file we need to be able to distinguish between creating a new file (case (c) above) or outputting to an existing file (case (d)). When creating a new file it is possible that we may be creating a new version of an already existing file. I would suggest that by default, output to a file should append to a file if the file already exists, or to a new one if not. If it is wished to replace an existing file, this can be done by preceding the first output to it by a rewrite (or possibly a delete).

So far it has been assumed that a file variable is associated with an actual file by correspondence of names, i.e. the file variable diskfile would reference a file known to the system as diskfile, unless over-ridden by some operating system control. If a file variable is to be able to be associated with files at will then there ought to be some method of associating a file name with a file variable. To make things simpler this could be restricted to being legal only when the file variable is not associated with a file. A possible way of allowing this would be to introduce (yet another!) standard procedure, e.g. name (f, s) where f is a file variable s is packed array of char and contains the name of the file.

Finally, I would like to put in a plea for random access of files. While accepting that a file should be considered as a sequential data structure, I think that the term sequential should only be considered to imply that there is some positional ordering of the items in the file and not that the items may only be accessed sequentially. Thus, I think it should be possible to access any item of a file at random using a procedure such as

```
seek(f,n)
```

where n (of type integer) is the ordinal number of the item to be referenced. If the n'th item does not exist the eof (f) would become true.

If the action required by the seek was not possible on the particular file, then a run-time error condition should be indicated.

Notes:

1. This scheme assumes that get and put may be mixed on the same file.
2. Seek(f,1) is equivalent to reset(f).
3. Failing anything more efficient, seek(f,n) could be implemented by:

```
reset(f);
i:=0;
while not eof(f) and i<>n do
begin get(f); i:=i+1 end;
```

4. An alternative to using seek would be to extend get so that

```
get(f,n)
```

would assign to f, a reference to the n'th item in the file. get(f) would still assign to f a reference to the next item in the file. Similarly for

```
put(f,n)
```

5. While the file variable can be conceptualised as a pointer to an item in a file, it would probably best be implemented still as a pointer to a buffer area and let the various file manipulation procedures be responsible for the data transfer between the buffer and the file.

G. I/O on Text Files

I think there is a great need to rationalise the methods of I/O on text files, particularly with reference to the read statement. Present suggestions are for formatted read statements, this I do not like.

In my opinion, execution of a read statement should be considered to be equivalent to the assignment to the relevant variables of some (unknown) constant values. The values should be represented in the text file in the same way that the same constants would be represented in the program. For example, given:

```
type colour = (red, green, blue);
var i: integer;
    b: boolean;
    r: real;
    c: colour;
    s: packed array [0..4] of char;
    v: array [1..3] of integer;
```

the effect of

```
read(input,i,b,r,c,s,v)
```

should be to assign

an integer value to i,
one of true or false to b,
a real value to r,
one of red, green or blue to c,
a character string of length 5 to s,
and an integer value to each of V[1],V[2] and V[3].

The actual values assigned would be the constants represented by the characters in the textfile, each constant being separated by one or more blanks, or possibly a comma. i.e. with the textfile input containing the following characters:-

```
12 TRUE -1.342 GREEN 'ABCDE' -5 2 15
```

the effect of

```
read (input, i,b,r,c,s,v) would be identical to
```

```
i:=12;
b:=TRUE;
r:= -1.342;
c:=GREEN
s:='ABCDE';
```

The occurrence in the textfile of a constant of the wrong type for the variable should cause an error.

Output to a textfile using the write statement should produce the characters representing the relevant constant value, preceded by a blank so that any output produced by a write statement could be input by a read statement. The current simple formatting in the write statement, using a field width specification, could be retained provided there is some predefined action (non-fatal error, or default output) if the value will not fit in the given field width.

H. Addition of Exponentiation Operator

If PASCAL is to ever supplant FORTRAN then I think an exponentiation operator (e.g. **) should be included at least as a conventionalised extension, if not in the standard itself.

I. In Conclusion

I suspect that some of my suggestions (if not all) will not meet with the general approval of PASCAL implementors and users. However, they do cover my main dissatisfactions with the language (or at least the ones that I don't think are being adequately dealt with by others) and I hope they will be food for thought. In writing all the above, I have tried to bear in mind that there are basically three areas of application for PASCAL with possibly conflicting requirements:-

- i. A simple teaching language
- ii. A satisfactory replacement for FORTRAN
- iii. A useful and powerful systems programming language.

I think that PASCAL can (but doesn't at present) fulfill all of these, without straying from its basic principles, but it is necessary to take care that in suggesting modifications one doesn't push one use at the expense of the others.

Above all, I would like to see some consistent standard (and soon!) that implementors would feel morally obliged to adhere to if they wish to call their language PASCAL.

Chris Bishop,
Computing Centre,
University of Otago,
P.O. Box 56,
Dunedin,
NEW ZEALAND.

(* Received 77/12/27 *)

LEIBNIZ-RECHENZENTRUM
DER BAYERISCHEN AKADEMIE DER WISSENSCHAFTEN
BARER STRASSE 21 D-8000 MÜNCHEN 2

Pascal User's Group
c/o Andy Mickel
UCC: 227 Exp. Enr.
University of Minnesota
Minneapolis, MN 55455
(612) 376-7290

München, den 9.11.1977
Telefon (089) 21 05/84 84 HW/br
21 05/
Telex: 05/24 634

Dear Mr. Mickel,
thank you very much for sending me PASCAL Newsletters
so promptly. Sorry that I am not equally prompt with my
thanks.

On my PUG Membership certificate you ask whether I am the
"EULER" Weber; sorry, I am not. My interest in PASCAL has
begun only half a year ago when we received the Zürich-
Compiler for our CYBER 175 (with Operating System NOS 1.1).

At the PASCAL-meeting of the German Chapter of the ACM I met
Urs Ammann who told me that you could probably help me with
information about interactive PASCAL-Systems for CYBER 175/
NOS 1.1.

From PASCAL Newsletters I learned that the maintenance of the
Zürich System is now in your hands. Did you get from Zürich
a list of all installations of their system? Anyway we are
very interested to be on your mailing list for future
corrections, extensions (?) and releases. From Urs Ammann
I heard that in early 1978 there will be Release 3 including
dynamic arrays. Since I am interested in procedures for
mathematical algorithms, I will be very glad about this
extension.

Also in PN, it was announced that a PASCAL-Software pool will
be created and will be handled by the respective compiler
distributors. Can you, please, inform me what libraries *and are*
available for CYBER 175 on what conditions.

I enclose an American Express Cheque to the amount of US \$ 7.00.
Please send me another set of backissues and two issues of
each PN for my membership year. Beside the set for our
official library I should like to have one for my own.

I am very expecting PN 9. The other ones I like very much.

Yours sincerely,

Helmut Weber
Helmut Weber

Direktorium: o. Prof. Dr. G. Seegmüller (Vorsitzender), o. Prof. Dr. F. L. Bauer, o. Prof. Dr. G. Hämmerlin, o. Prof. Dr. K. Samelson

Department of Computing Science,
University of Adelaide,
North Terrace,
Adelaide,
South Australia 5039

28th October, 1977.

Dear Andy,

I am writing to correct an impression which may have been given
by Arthur Sale's letter in Pascal News, September 1977.

At the University of Adelaide, we were fortunate to have access to
Pascal several years ago because the University computer was a
CDC6400 - I personally first taught Pascal to students in 1974.
After two years experience with earlier versions of the language,
in 1976 the department adopted Pascal as its main teaching language.

We have now completed two years of teaching with Pascal as the main
language in first year, second year and third year! In detail, some
500 students in our courses are programming in Pascal in any one
year - writing programs from prime number generators to compilers.
The number of enthusiasts is growing!

Yours sincerely,

Barbara Kidman.

Copy to: Professor A.H.J. Sale,
University of Tasmania.

Thomas J. Kelly Jr
58-B Meadowlake Drive
Downingtown, Pa. 19335

Dear Andy

3 Nov 1977

Just a short note to tell you:

I've moved, new address is above.

I've changed jobs: I now work for Burroughs Corp. I exerted some
effort and we are now using the UCSD implementation of PASCAL on our
B7700. For any other B7700 users who need a compiler, you can get one
from UCSD, I suppose. It's okay, but we have discovered several bugs.
There is also a fix that needs to be installed to allow the generated
code to run properly on a B7700 (as opposed to B6700). I will send that
fix on to UCSD, I don't know if they'll put it in. If anyone wants to,
they can get one directly from me (at the above address).

We are currently beginning work on adapting Brinch Hansen's Sequential
Pascal for one of our internal projects. Interest in PASCAL and its
derivatives (especially MODULA) is increasing here at Burroughs, which
I consider to be a very good sign. I can't say that it is all my
doing, but I like to believe I've helped.

Take care,

Tom

Open Forum for Members

Open Forum for Members

THE UNIVERSITY OF BRITISH COLUMBIA
2075 WESBROOK MALL
VANCOUVER, B.C. CANADA
V6T 1W5

DEPARTMENT OF COMPUTER SCIENCE

25 August 1977

WCC Memorial Hall W.
Indiana University
Bloomington, Indiana 47401
October 12, 1977

Mr. Andy Mickel
Editor, Pascal Newsletter
Computer Center
University of Minnesota
Minneapolis, Min 55455

Dear Andy:

I would like to add my two cents worth to the animated discussions concerning Pascal.

I believe that the difficulties that arise while using interactive input files would be resolved if a GET was required before the file variable was defined. This would allow a programmer to state explicitly when (s)he wanted the first read from the file. This modification could be made to Pascal, and currently running programs would compile and run correctly by simply adding a GET at the beginning of the program.

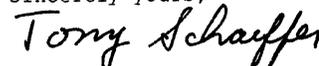
I think that the discussion of standards is missing historical observations of natural languages. Well standardized languages are dead languages! One only has to look at Latin or ANS Fortran. I believe that the following proposal would allow Pascal to evolve, but at the same time allow portability of programs. The language processed by a translator can be called a dialect of Pascal if:

1. It includes "Standard" Pascal as a subset.
2. The implementor has provided a mechanical translator (written in "Standard" Pascal) of the dialect into "Standard" Pascal.

I would be willing to accept a dialect that satisfied both of these conditions as "being consistent with the design of Pascal." Of course, when a program in the dialect was shipped to another installation, the dialect version and either the "Standard" version or the translator would also be sent. I would suspect that it might be reasonable for the "Standard" version that is appropriate in this context to be "sub-Standard" in comparison to the present idea of "Standard" Pascal.

I would like to obtain copies of the back issues of PUGN, but I have two sheets that conflict in the listed prices. How much would it be for the issues that you have.

Sincerely yours,



Anthony J. Schaeffer

Andy Mickel
University of Computing Center
227 Exp. Engr.
Univ. of Minnesota
Minneapolis, Minn.
U.S.A. 55455

Dear Andy,

I have finally found the time to rewrite the paper which I sent last spring. I'm afraid it was sent without proofreading. It is much longer now so I've divided it in two parts in case you want to print it in installments.

I disagree with your statement about not changing Pascal [PUGN #8, pg. 29]. You may have guessed this by the nature of the accompanying paper. Pascal can, and should, be changed now, before a standard is created.

You state that documentation, implementations, and software exist, so the language should remain static. But minor changes will be needed to all of them when a standard is created, so it is a good time to make other needed changes.

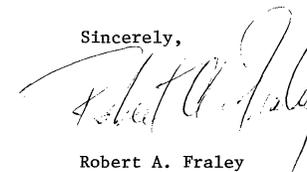
A change from Pascal to a revised Pascal ("Pascal II") can be compared to the change from Fortran II to Fortran IV which occurred in 1964. A mechanical translator aided in the translation, but many long hours were spent re-writing programs. A Pascal to Pascal II conversion should allow a more effective mechanical translation to occur, but machine-dependent features and under-the-counter type conversions would hinder this effort.

The point is this: a change now will affect a relatively small number of people. If you wait until a standard is made and then try to change the language, the amount of pain and agony will be increased by an order of magnitude or more.

I don't pretend that every change suggested in my paper would produce a superior product. Some have not been tried; none have been subjected to a large user community. I am describing problems which I have found with proposed solutions so that reactions can be gathered. Above all, I do not suggest that change be made for its own sake, but that change remain an open possibility.

While committee action is not good for a language design effort, it can be effective for the task at hand. Designers in the committee can independently create language versions which combine the present language with the suggestions presented in the newsletter. Committee critics can force a justification of each change, and can compare different solutions to problems. A group from varied backgrounds can provide a degree of universality which a single designer cannot hope for.

Sincerely,



Robert A. Fraley

November 7, 1977

Professor Arthur Sale
 Department of Information Science
 University of Tasmania
 Box 252C
 Hobart, Tasmania 7001
 Australia

Dear Professor Sale:

I read with interest your contributions to PUGN #9. Here are a number of reactions.

While I am strongly in favor of an "ELSE" clause in the CASE statement, I do not like your choice of keyword. In my recent work at the University of British Columbia I found that a common error, committed both by students and faculty members, is to place a ";" before the word ELSE in an IF statement. It was so common that (in our extended compiler) we allow an optional semicolon in the IF statement. This has several effects:

- a. Programming errors and resulting frustrations were reduced.
- b. 2-token look-ahead is required. (In a recursive descent parser, a Boolean flag can be added to indicate whether a ";" was swallowed by the IF statement procedure before discovering that no ELSE exists.)
- c. Combined with your syntax, 3-token look-ahead is needed.

Without this addition, however, an IF statement in the preceding case clause which contains an erroneous semicolon will probably get an obscure error message.

Our choice for the default clause designator was the token "<>". We thought that it looks nice, and it suggests that values not equal to the other labels should use this clause.

Sets of characters are a problem, but I think you missed the heart of the issue, at least in this discussion. The Pascal standard does not define the meaning of sets, so implementations differ. Sets behave in many ways like

PACKED ARRAY [type] OF BOOLEAN

except that "type" must start at 0 and has a fixed upper bound (in some implementations). If such a restriction were placed on arbitrary arrays, with the upper bound being implementation defined, the users would scream loudly. So why make such restrictions on sets?

Sets of characters aren't the only problem. I am implementing an interpreter which has 150 instructions, defined as a scalar type. Will I be able to transport my program if I use sets of these instructions?

The one language "feature" which prevents a general definition of sets is the set constructor.

[3, I, TRUNC(EXP(R))]

Is the set above of type SET OF 0..10? Type SET OF 0..59? I can't tell. This is one situation where the strong typing of the "pure" language fails, and the resulting type is therefore left to the implementor.

I would make three proposals:

1. When S is a set type,

S [A, B, C]

 denotes a set of type S. This allows explicit control of the subrange.
2. When the elements of a set constructor are a scalar type T other than INTEGER, the current set constructor produces type

SET OF T

 rather than a subset of this type.
3. The current set constructor is illegal with non-constant INTEGER operands. The notation in point 1 above must be used.

One other point which you raised was the character set question. I am sick and tired of this problem, due to the variety of character codes which I've encountered. (1401 BCD, 7090 BCD, 1620 code, Univac 1108 code, CDC code, EBCDIC, ASCII, and occasional others.) The ASCII code was agreed upon ten years ago by representatives from across the industry, and closely resembles the international standard code. While its purpose is information interchange, we could contort its original purpose and say that programs are indeed information, and their algorithms (to be transportable) must be expressed in terms of ASCII.

If Pascal were to impose an ASCII rule today, it may alienate a number of supporters. But if users don't push for some sort of standard, we will never free ourselves of this problem. Pascal currently requires that '0'..'9' be contiguous. Perhaps it could also require that sets such as

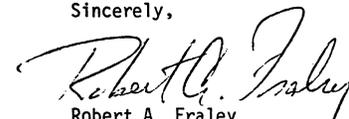
['!'..'@']

be interpreted as containing those ASCII characters between '!' and '@'. This requires a conversion table at compile time to compute the set. In those relatively rare occurrences where the limits are variable, a runtime conversion table is required.

An eventual goal of ASCII in all aspects of the language within (say) five years might be nice. It might encourage Pascal implementers to prod the hardware companies into supporting ASCII files and allowing an override of the ASCII to internal conversion for terminal input. (Univac provided such facilities several years ago for the 1108 series.)

I am enclosing some of my own suggestions for Pascal changes, which will appear in PUGN #11. I am looking forward to hearing your reactions to these suggestions.

Sincerely,


 Robert A. Fraley
 Hewlett-Packard Laboratories
 Electronics Research Laboratory

RAF/hma
 cc: Andy Mickel

OMSI COMPUTING

"OMSI Computing" is a product and service name of Oregon Minicomputer Software, Inc.

December 26, 1977

Dear Andy,

It seems that we owe you and the Pascal User's Group a good deal of information about our work with Pascal, so perhaps this letter can serve as an informal introduction.

'We' are a small group of computer freaks who have been working and playing together for about 7 years. The group is an outgrowth of the Student Research Center of the Oregon Museum of Science and Industry (OMSI), a not-for-profit private science education center in Portland, Oregon. To make a long history short, we began in 1970 with a very small DEC PDP-8 system and over the years grew to a large PDP-8 and a large PDP-11/45 computer facility. This expansion was mostly financed by swapping software written at OMSI with the manufacturer (DEC) for additional hardware. Several past and present DEC software products were first developed at OMSI (the PDP-11 APL is the latest example).

The group members (John Ankcorn, Don Baccus, Wayne Davison, Steve Poulsen, Barry Smith, Rusty Whitney, Dave Vann) grew up as assembly language programmers. Our interest in Pascal began when Wayne noticed the Acta Informatica report - our continued interest is probably because Pascal has the structuring tools we found necessary (and available!) in assembler. Pascal is also "small enough" for practical implementation, efficient for real-world programs, and (if used carefully) really machine independent. (I should note that we're not Pascal fanatics - we use several other languages, and follow Wirth's Modula with much interest.)

Our Pascal history shifted in 1974 to Electro-Scientific Industries (ESI) in Beaverton, Oregon. ESI manufactures precision electronic equipment, including computer controlled systems for on-line laser trimming of hybrid integrated circuits. The first laser systems were controlled by DEC PDP-8's with a special-purpose language. When ESI switched to PDP-11's and was faced with the task of rewriting the control software, we (somewhat hesitantly) suggested Pascal for the implementation. ESI's enlightened management (Don Cutler, Bob Conway) agreed, and ESI sponsored the development of a PDP-11 Pascal compiler (ESI Pascal). The compiler (initially patterned after an early version from the University of Illinois) was written by John Ankcorn and Don Baccus with assistance from David Rowland, and became largely stable in early 1975.

Significant features of ESI Pascal include:

- single pass compiler coded in PDP-11 assembler
- translates to assembler code (not interpretive)
- full standard Pascal
- extensions for process control
- stable, reliable production compiler

Experience with ESI Pascal at ESI and OMSI has certainly convinced us of the practical utility of the language - if anyone in the PUG needs a showcase example of "Pascal in the non-academic world", contact ESI. However, ESI's design constraints resulted in a compiler which is not as general-purpose as one might wish. Specifically, the requirement of a 16K word operating environment led to a very tightly coded bare-bones compiler (both Wirth and Tony Hoare expressed surprise at the 11K word (22KB) compiler, and I believe we have a reasonable claim to the smallest standard compiler in existence). The ESI compiler is not well suited to the "naive user", and of course an assembler coded compiler is hardly portable.

Agreeing with Arthur Sale's comment that Pascal must have very robust compilers and support software if it is to be taken seriously, we began in late 1975 the development of a totally new Pascal programming system. The compiler for this system was written in standard Pascal and was designed with several coequal goals: efficiency of compiled code; portability of the compiler and support system; robustness, and usability of the entire system. We attempted to see each design decision from the user's viewpoint rather than the compiler writer's. Our view of the 'typical user' is the professional Pascal programmer - one who expects system software without glitches or rough edges, one who works in a Pascal environment with high-level support tools (excellent diagnostics at compile and runtime, Pascal debug and library facilities, measurement facilities for practical engineering).

The compiler for this system (written almost entirely by Don Baccus) is currently producing code for some members of the PDP-11 family. It is certainly larger than the ESI Pascal compiler (runs in 2-4 phases, depending on optimization and other options), but the time from compilation to program execution is roughly the same (the compiler produces directly executable object files, whereas ESI Pascal requires use of the assembler and object linker - both much slower than the compiler!).

Recently (October 1977) we formalized our relationship and created a corporation independent from the science museum - Oregon Minicomputer Software, Inc. (known simply as Oregon Software to our friends). "Out of the frying pan, ..."

We've arranged with ESI for Oregon Software to distribute and support ESI Pascal (we call it OMSI Pascal-1). The updated implementation notes should read: Implementors, John Ankcorn, Don Baccus, David Rowland; Distributor and Maintainer, Oregon Software; Machine, any model PDP-11 (including LSI-11); Configuration, 16K memory, RT11, RSTS/E, or RSX operating systems (all the DEC systems at last!); Distribution, 9 track 800 bpi magtape, DEC cartridge disk, \$1500 (\$995 for educational use); Reliability, excellent - currently about 60 installations and growing steadily.

We've also been talking with Ken Bowles at the University of California, San Diego, and are pleased to announce that the UCSD Pascal system is available now for RSTS/E timesharing systems - an excellent system for introducing programmers to Pascal. Price is not yet fixed, but should be established by the time this is published.

Our new Pascal programming system (known as OMSI Pascal-2) is not yet available - we'll keep you informed as to our progress with the PDP-11 and other machines.

One last note - we have a limited supply of Pascal T-shirts with a portrait of Blaise Pascal from a woodcut frontispiece for a biography published in 1891 - see enclosed copy. These shirts are 100% cotton, hand silk-screened, available in sizes S, M, L, XL - price is five dollars postpaid from Oregon Software.

Merry Christmas, and a happy New Year!

Barry Smith

Barry Smith
Oregon Software
4015 SW Canyon Road
Portland, Oregon 97221
(503) 226-7760



Oregon
Software inc. minicomputer

4015 SW Canyon Road Portland, Oregon 97221 (503) 226-7760

Dear Andy,

Just a short note to describe an application of Pascal here at I.C. over the last couple of months.

All university computer installations must be plagued by their share of Startrek games, mostly written in the most hideous basic or horrendous pl/i (sorry Ploughskeepie). Naturally most of these installations suffer because of it, both in terms of machine time wasted and the constant staff overhead of having to spend hours looking for and destroying incarnations of the game only to have a Physics undergraduate restore it the same evening.

Our own case was slightly different. The Department runs VM on a 370/135 with 384k of store. Mercifully, our basic compiler wouldn't handle the Startrek program run on our College Computer Centre's CDCs: but even had it compiled, it could have been debatable whether the kind of response that would have been achieved could have justified the tag 'interactive'. Thus we set about writing our own Startrek, in Pascal, which we hoped would be more elegant and efficient than the versions we had access to.

The user interface was designed and implemented by Greg Pugh (the guy who implemented our P4 compiler). It maintains a full screen display on Lynwood VDUs, showing the status of the Enterprise, short and long range scans, damage control etc.. The Lynwood features blinking, cursor positioning, protected mode etc. and Startrek uses them to the full. Using the cursor positioning, only the fields that change between moves are updated. This leads to a faster (and more exciting) game. Because everything is continuously displayed on the screen, we have been able to remove all the commands controlling the display (like damage control report), so the user only has to remember about 10 commands. These are entered as English words, like 'WARP' :no more lost games 'cos you typed 3 hoping to fire photon torpedoes and actually warped into a Nova!

We produced our first version in June '77, which ran quite successfully (and efficiently) despite the abuses received during our department's open day.

However (and this is the interesting bit, I hope!) we have since extended the Startrek system to run up to 16 users simultaneously, without modifying one line of the source code. This was possible because of the natural re-entrancy of Pascal code. All that was needed was a small interrupt handler (1000 lines of Assembler) to interface Pascal and the terminals. As each user 'dials' into Startrek, the interface routines find some store for his (her) stack/heap, and enter the Pascal bit of Startrek at the beginning. The program runs until it initiates I/O, at which point it is suspended and other users run until it completes. Because Startrek is thoroughly I/O bound, we have experienced no problems with its being totally interrupt driven (indeed on VM the level of I/O activity is a boon; it stops you from getting dropped from the "fast response" queue!).

We have run 8 simultaneous users on the same Startrek program, with no noticeable degradation in the response of the system generally or of Startrek itself.

Since Startrek has paved the way (boldly going where no man etc...), we have really "gotten into" multi-user Pascal programs here at IC. Iain Stinson has written a filing/archiving system which services 16 users simultaneously. A multi-user context editor is on the way.

I think the thing that surprised us was the ease with which all these were implemented. In all cases, the actual low-level interface needed was surprisingly small and took very little time to write. The Pascal programs, because they do not need to know that they are multi-threaded, are equally easy (and of course are easily tested with a more conventional, single user interface).

Anyway, if anyone is interested, send us a tape (preferably containing some goodies of your own) and we'll return a copy of the system (including our Pascal compiler). Please specify 800 or 1600 BPI. The game itself has a "user manual" in the form of a Starfleet technical directive, but unfortunately only sketchy implementation notes exist.

Keep Blaising the way.

Live long and Prosper.

Dave Thomas

Dave Thomas.

Department of Computing and Control,
Imperial College
180 Queensgate
London SW7
London 589-5111



STATE OF MINNESOTA
 Crime Control Planning Board
 6th Floor, 444 Lafayette Road
 ST. PAUL 55101

November 7, 1977

Andy Mickel
 227 Experimental Engineering
 University Computer Center
 University of Minnesota
 Minneapolis, Minnesota 55455

Dear Andy:

In the September 1977 issue of Pascal News I was quoted regarding LEAA (Law Enforcement Assistance Administration) regulations vis-a-vis programming languages for use in criminal justice information systems. While the thrust of that quotation was essentially correct, it may be useful to publish the actual guideline referred to:

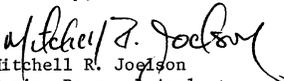
From: U.S. Department of Justice
 Law Enforcement Assistance Administration
 National Criminal Justice Information
 and Statistics Service
Comprehensive Data Systems Program Guideline
Manual (M6640.1, April 27, 1976)

Chapter 3/Paragraph 37d:

Whenever possible, all application programs will be written in ANS COBOL in order that they may be transferred readily to another authorized user. Where the nature of the task requires a scientific programming language, ANS FORTRAN should be used. Request for waivers shall be justified in writing first to the appropriate Regional Office and then to NCJISS/SDD for review.

The most recent Direction of Automated Criminal Justice Information Systems shows virtually all existing Criminal Justice software (whether developed under the federal guideline above, or by a local agency without benefit of such paternalistic guidance) to be written in COBOL or FORTRAN.

Sincerely,


 Mitchell R. Joelson
 Senior Research Analyst
 Community Crime Prevention

MRJ/amc

THE UNIVERSITY OF NEW SOUTH WALES

P.O. BOX 1 • KENSINGTON • N.S.W. • AUSTRALIA • 2033
 TELEPHONE 663 0351
 EXTN.



PLEASE QUOTE
 Department of Computer Science,
 School of Electrical Engineering.
 December 30, 1977.

Dear Andy,

First let me congratulate you and your colleagues for the excellent job you have been doing on the Pascal News (letter).

The new Australasian distribution arrangements will certainly be welcome to people down in this part of the world and I am grateful to Arthur Sale for offering to print and distribute Pascal News but I too am confused over the Australasian subscription rate of \$10! This makes Pascal News the most (relatively) expensive journal/newsletter to which I now subscribe. You can take it as considerable proof of the quality of your publication that/if members continue to subscribe. The price disparity between the Australasian on one hand and the USA and UK subscription rates on the other is such that Australasian subscribers are surely owed a public explanation (through Pascal News).

I feel I must correct Arthur Sale's misleading claim to "a first for reactionary Australia" for his proposed switch to Pascal in his first year course in the next academic year. That Australia is reactionary is beyond doubt but surely Arthur is letting his newfound enthusiasm for Pascal blur his perspective of what is happening in the rest of Australia. Simply to set the record straight and without any pretension to claiming a "first", I would like to point out that the Computer Science Department at this University has been teaching Pascal as a first and principal programming language for the past three years.

I enclose a collection of sundry comments on various aspects of Pascal.

Best wishes.

Yours sincerely,



Ken Robinson

Sundry comments on Pascal

Standardization: I fully support the current moves for standardization of Pascal, however I believe a number of minor changes should be made to Pascal before such standardization takes effect. I disagree with Prof. Wirth's sentiment (PUGN #8) that changes should not be made to "Standard Pascal" as "it would seem unfair to suddenly declare that what once was a Pascal compiler now suddenly isn't any longer." Standard revision must be prepared to add or even change features to a language if there are compelling reasons for doing so, and as a result it has to be accepted that compilers will be obsoleted and require change. Pascal should be kept alive as a language; standardization could sound the death knell of Pascal somewhat earlier than necessary. FORTRAN standardization provides an example which should not be followed - FORTRAN is of course dead; many people do not realize it.

Exponentiation: The exponentiation operator should be part of Standard Pascal. The look of disbelief when a Pascal programmer first discovers that it doesn't exist!

Complex: 'Complex' should be added as a predefined type. Have you ever tried to convince a group of Electrical Engineers to use Pascal? They work most of the time in the complex domain and when you explain how you do complex arithmetic in Pascal ... !

Pascal is surely not only for compiler writers. We cannot ignore programmers who work with real and complex numbers.

Semantics, Axioms and Undefinedness: The comment is frequently made that Pascal "leaves undefined the action of a CASE where the expression evaluates to a value not matched by a case label" (PUGN #6 p60), or some such similar comment, usually implying that the action is left to the whim of the implementor.

Nonsense! The observation above is quite precise: the action is undefined not arbitrary. Just as the action of multiplying two boolean expressions is undefined. We seem to have become accustomed, due to our experiences with badly implemented software or even hardware, to confuse "undefined" with "arbitrary". Surely if an action is undefined we expect good software to detect such an infringement of the semantics and to signal an error? Incidentally there is no analogy between a case statement with no label for a particular value of the case-expression and a if-then statement in which the boolean expression; the semantics of the latter are quite well defined.

On the above question and many others the axiomatic definition of Pascal by Hoare and Wirth [1] is very clear and I am surprised that this paper is not referenced more frequently within discussions of the semantics of Pascal.

The case-statement: I fundamentally lean towards Wirth's position that the case statement should not have an exception case (else label) but one particular exercise has fairly well convinced me that such a facility is necessary. The exercise: write a reasonably portable lexical analyzer in Pascal.

Solution 1: Use a case-statement to classify the next character. Then all characters will have to appear as case labels, but many/most of the characters in ASCII/EBCDIC are unprintable (at the least) which certainly does not aid documentation. Some characters may even be all but impossible to include in the program, for example chr(0) in the CDC character set does not always have a character representation; the ETH compiler gets around this one by predefining a constant (undocumented) COL = chr(0)!

Solution 2: Guard the case statement using a set as follows:

```

if nextcharacter in set of characters of interest
then case nextcharacter of
    ...
    end
else ...

```

Useless! due to the severe constraints on set of char in most implementations (see below).

Solution 3: Guard the case statement by selecting a subrange as follows:

```

if (nextcharacter >= first character of interest)
and(nextcharacter <= last character of interest)
then case nextcharacter of
    ...
    end
else ...

```

OK for ASCII but not of much help for EBCDIC. In general the problems encountered in solution 1 remain.

Solution 4: In desperation you might abandon the case statement and use a mapping vector to map from char to an enumerated scalar type. This solution eases the portability problems but at some cost: the mapping vector is neither easily nor efficiently initialized in Pascal!

Conclusion: the case statement remains attractive, but an exception case label appears to be necessary when discriminating on an expression of type char. I believe the exception case is not necessary for expressions of another type.

Note: the lexical analyzer in the ETH compiler is not a good solution to the problem, being solution 1 above. Unfortunately (or fortunately) the CDC character set is small enough to mask most of the problems which arise with larger character sets.

Set of char: I am inclined to go further than Arthur Sale (PUGN #9,10 p66) in relation to minimum set size. Rather than picking an arbitrary number such as 32 I feel it would be better to insist that every Pascal compiler should support set of char for its particular character set(s). There is nothing to stop the compiler optimizing in the case of smaller sets of course.

Without the above requirement sets containing characters must be banned from programs which are required to be portable, and surely many if not most large programs should be portable.

For statement: While agreeing with Arthur Sale's comments on the semantics of the for-statement (PUGN #9,10 p66) I disagree with his suggestion of describing the semantics in the form of an equivalent Pascal sequence. A similar scheme was used in an earlier Pascal Report and was rejected in the axiomatic definition of Pascal [1] for reasons of over specification. The definition of the for-statement given in the axiomatic definition is not open to the same ambiguity as that discussed by Arthur Sale since the concept of "assignment to the control variable" does not enter the axiom. Nor does the axiom say anything about the final value of the control variable, i.e. it is "undefined" (see above).

All of the problems associated with the control variable are removed if the for-statement is implemented in such a way that the control variable is local to the statement (as does AlgolW and dare I say Algol68); the value of the control variable after executing the statement is beyond discussion since the variable no longer exists! I strongly urge that the Pascal for-statement be implemented in this way and note that it is only an implementation change - the semantics as defined in the axiomatic definition are not changed at all.

Files: I must disagree strongly with many of Arthur Sale's numerous criticisms of Pascal's file type. A file is definitely a valid data structure and it is one of Pascal's strengths that file is but another predefined structure and not some strange special object. True! Pascal, at the moment, only supports sequential files and there are other file structures which are ignored, amongst which direct access could be singled out as a file organization which should be supported. I would disagree with Wirth's second thoughts (PUGN #8 p23) of substituting "sequence" for "file"; perhaps the qualified "sequential file" instead of simply "file" is preferable but I feel that the word "file" is required as a reminder to the programmer that this structure will in general reside on secondary storage.

Why is an array of files an absurdity? Done any merge sorting lately? As for file lifetime and scope being bound; doesn't this give you the perfect realization of a temporary file associated with a particular process? The anomaly of the lifetime of external files and their apparently different scope is removed by my suggested change to the program statement below.

The program statement: Pascal's program statement has been frequently picked on as "an importation from CDC FORTRAN" and as some kludge to allow external files to be specified. There seems to me that there is a lot of truth in these observations but the real problem is that the program statement should have the same form and function as the procedure statement. After all a program is really only a distinguished procedure which is called from an outer block (level 0) which represents the external environment.

I would like to see the syntax of Pascal changed (as shown in the syntax diagram below) to allow declarations to be placed at level 0 and for the program statement to have the same syntax as the procedure statement except for the replacement of procedure by program. I assume a number of restrictions will be required, for example a program may be declared at level 0 only and there must be only one program - but maybe you could have more than one program?

The effects of this change are manyfold:

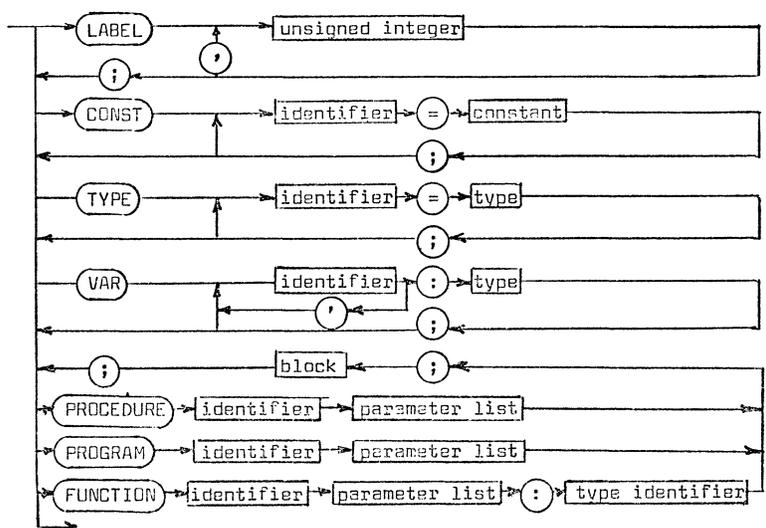
external file parameters would now be declared as var parameters;

the files input and output would be declared implicitly at level 0 and thus simple programs could reference input and output without specifying any file parameters in the program statement;

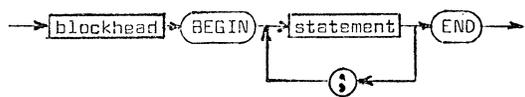
external procedures and functions could be declared in their correct environment and, assuming the compiler compiles an object of class "module", external procedures and functions could be compiled separately without awkwardly (and usually erroneously) imbedding them in a dummy program environment;

a program may now possess formal parameters other files.

blockhead



block



module



Revised syntax of Pascal <block>.

Example:

Old syntax:

```
program foo(input,output,myfile);
    ...
var
    myfile: text;
```

New syntax:

```
program foo(var input,output,myfile: text);
```

Reference:

1. C.A.R. Hoare and N. Wirth: *An Axiomatic Definition of the Programming Language Pascal* Acta Informatica 2, 335-355 (1973)



SPECIAL TOPIC: PASCAL STANDARDS

(* Andy Mickel and Jim Miner *)

We (Jim and Andy) feel that it is a good time to review what is happening with Pascal standardization and bring new PUG members up to date.

We believe it is essential to have a tight, officially standardized base language especially on which to develop conventionalized extensions.

ISO STANDARD Pascal

In general, events are going very well in the effort to obtain an official standard. In a very short time (less than a year) the effort is more than half done. For those of you who haven't read PUGN #8, Tony Addyman is obtaining an official ISO (International Standards Organization - the people entrusted with, for example, the metric system of measurements) Pascal standard. Tony began to organize a British Standards Institute (BSI) working group after the University of Southampton Pascal Symposium. It is passing on a standards document consisting of the Revised Report with the semantics "tightened up." In no case were new language features to be considered.

Tony is a member of the programming language committee at BSI, and his working group has met several times (in June, September, and this January). Jim Welsh and R. Tennent have supposedly written papers for the group which we at PUG central have not seen. So far Tony has attended an ISO language subcommittee meeting in the Netherlands in November at which he requested consideration of an ISO Pascal standard for the first time. A Swedish technical committee also had representatives there to request the same thing. The French and German representatives were also keen on the idea (the Americans were COBOL people who "were not interested at this time."). A superior committee of ISO had to decide the question, and after the necessary applications are made it should be on the ISO agenda by late March (Ken Bowles' letter quoting 3-5 years notwithstanding).

One problem has been that up until now, all programming language standards have been American (ANSI) standards and that ANSI has cooperated with the rest of the world. For example the ISO COBOL standard is simply a couple of sentences which says to refer to the ANSI standard. In other words, precedents do not exist for an ISO standard starting from scratch. Apparently a group recently pushing for ISO ALGOL-60 standardization was helping do the work to set precedents on procedure. The ALGOL-60 effort has been stymied, and ISO is now considering mechanisms for advancing a language standard. As Tony puts it, the UK has the conscience of Pascal "in its pocket because the UK proposed first."

We at PUG central have stated before our reluctance to go through a lengthy ANSI process for a standard. COBOL, BASIC, and FORTRAN ANSI standards were long in coming. No one that we know of has proposed going through the American National Bureau of Standards (NBS). This may perhaps be an easier route. But we endorse the current European effort to standardize Pascal - after all it is a language with European origins!

The Swedish technical committee announced their existence to PUG with a letter from Bengt Nordstrom (in this section), and sent "Yet Another Attention List." We really appreciate that, and we're glad to see contact with the British. Further, Olivier Lecarme has in the past told us of the Pascal sub-group he coordinates in AFCET (the French counterpart of ACM). We understand that the Germans are also organizing as a result of the successful Pascal Conference in October held by the German ACM (see Here and There).

CONVENTIONALIZED EXTENSIONS

Pierre Desjardins wrote us on 77/10/13: "Have you given any thought as to how to proceed once you (we, "the pascalers") decide to conventionalize an extension of Standard Pascal?"

Good question. Going back to last year in PUGN and at the University of Southampton Pascal Symposium, we developed a consensus which directed the standards effort to do precisely what Tony is doing. Further, areas exist (such as those outlined in Niklaus Wirth's letter in PUGN #8) in which there is no point in having different implementations add similar constructs in different ways. So, if an implementor chooses to extend Pascal in a certain direction, he or she should stick to an established convention if one exists.

Ken Bowles' letter (in this section) proposes a workshop for the purpose of getting together a set of conventionalized extensions. We feel this is a good first start. Because there is a lack of practice and experience in having implemented a number of extensions (such as variable extent array parameters), the results of the workshop should not be interpreted as a final act.

LAUNDRY LISTS OF ADDITIONAL FEATURES

Pascal is often mistaken as a SIL (systems implementation language). Although it may be good for writing compilers, Pascal without extensions cannot be used to write operating systems, for example. We think more support should be given to making Pascal available for general user use where there will be widespread benefit.

It is sad to see more people calling for more and more and more redundant and even whimsical features in Pascal. Pascal's virtue is its small size (limit on the total number of features) which has enabled its quick spread to other machines; implementation effort is small. That's a significant fact. If you ever want to push your own language or large software system someday, perhaps you can write it in Pascal for portability purposes instead of straitjacket FORTRAN. So Pascal is paving the way.

With regard to these laundry lists of changes, people still seem to be forgetting things we learned last year from discussions in PUGN: Niklaus Wirth wrote in a letter in PUGN #5, "we must clearly distinguish between the language and the implementation...the language is defined by the Report alone, and intentionally leaves many details unspecified that an implementation inherently must define in one way or another."

We think that dissatisfaction (leading to suggestions of new features/changes) may be the result of problems in the implementation and could be solved in the implementation and not by changing the language. Niklaus Wirth also wrote in a letter in PUGN #8: "...most... other extensions...belong to a different category which, I believe, has nothing to do with the goal of obtaining a common language. Rather their primary objective is to introduce some favourite facility suggested by either a particular application or, more frequently, an existing operating system. Whereas I have no objection to such extensions in principle, they do not belong in the core language, whose facilities must be understood without reference to any particular implementation. If at all possible they should be incorporated in the form of predefined procedures, functions, types, and variables and in the documentation they must be clearly marked as facilities pertaining to a given system..."

And what about the design goals of Pascal (which appear on the inside back cover of PUGN)? Remember that the combination of these goals has to be considered. Some of the articles we print in PUGN may unfortunately give the impression that there is a lot wrong with Pascal because they are full of suggested changes. We are growing weary of such articles which too often don't explain their suggested changes in the light of the design goals in proper proportion.

To quote Richard Cichelli, one of the world's foremost practitioners of Pascal in industry, "the problem in the United States is not the lack of language facilities, but rather making programs understandable and to communicate them to average and below-average programmers!"

We say that changes to Pascal are basically irrelevant. It is so far ahead of the competition! It's strange that the phenomenon of real people using Pascal as it is for so many real things can be overlooked. People who need to get real work done now can't wait for academics to come up with Utopia 84. So-called "improved" languages such as EUCLID were announced before being implemented (shame, shame!) and are not tempered by good ole practice and experience.

Let's examine as an example, the case for adding an exponentiation operator to Pascal. One of the design principles of Pascal is not to hide time and space efficiency costs from the user. An exponentiation operator makes it easy to be wasteful because we dare say that in FORTRAN, its most frequent use is to square a number. (Pascal provides a square function (SQR) which can be compiled in-line for efficiency.) The addition of an exponentiation operator confuses the evaluation of arithmetic expressions. Is $-2**3**2 = -512$ or -64 or 64 ? Of course parentheses can be used, but exponentiation like subtraction and division (but not multiplication and addition) is not associative (or commutative for that matter). Also exponentiation is hard to axiomatize (like real arithmetic) so that is probably another reason why it was left out of the base language.

But by using the principle of extending the language through predefined identifiers, simply predefine a function called power (as suggested by section 11 of the Revised Report:

```
function power ( x, y : real ) : real;
```

We know of implementations which have easily done this.

As a last thought, we don't think people should look at Modula, a language developed by Wirth for time-dependent ("real-time") programming, to find features which can necessarily be viewed as improvements. Modula's design goals are different. For one thing it is a much smaller language than Pascal, and its syntax is stripped down, too. Modula is another experiment, and we think it is an interesting one, but Modula will not replace Pascal in the foreseeable future.

Pascal COMPATIBILITY REPORT

To end on a bright note, on 77/12/19 we received an excellent report by Arthur Sale:

Sale, A.H.J. (1977): "Pascal Compatibility Report", Department of Information Science Report R77-5, University of Tasmania.

which should be of interest to the standards working groups and we hope they obtain a copy.

The abstract reads: "This report collects implementation variations between significant Pascal compilers for the edification of users, the education of compatible software writers, and the influencing of implementors. In most cases, the behaviour cited is a result of (a) an explicit loophole allowed by the User Manual or the Revised Report, or (b) an undefined loophole left by these documents. An attempt has been made to document important differences only, not local extensions, nor areas where great deviations are to be expected."

A special note reads: "The information contained in this report is dated at November, 1977. Since compilers change with time, and the ones on which the tests were made cannot always be certified in mint condition by the author, caution is to be used in relying on any information found here. A future edition of this compatibility report may be issued as more information is available."

The main report consists of a series of two-page sections containing a test program and the responses made by compilers to it and the questions posed. The responses are sometimes abbreviated to cut down verbiage at the expense of accuracy.

Test programs include: boolean expressions, const declarations, identifiers, for loops 1, 2, 3, and 4, pointer types, set types, with statements, variables, variant storage.

Compilers used in the preparation of the report were the Burroughs B6700 (Tasmania), CDC 6000/Cyber 70 (Zurich), ICL 1900 (Belfast), ICL 2900 (Southampton), Univac 1100 (Copenhagen), and the DECsystem 10 (Hamburg).

December 9, 1977

Postal address: Technical Committee on Pascal
CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Sciences
Fack
S-402 20 GÖTEBORG 5 SWEDEN

1977-11-30

Dear Andy,

There is a growing interest of Pascal in Sweden, both in industry, universities and other governmental institutions. There are half-a-dozen groups working with implementations of Pascal of some variety. This interest has lead to a couple of meetings with implementors, users and possible users. These meetings have resulted in the creation of a technical committee for Pascal in Sweden.

The goals for this committee are to:

- critically analyze current definitions and implementations of Pascal in order to discover problematic spots.
- suggest a standard for implementation of Pascal which solves most of these problems.
- work for getting an international standard for Pascal.
- in a second phase develop
 - a programming standard for Pascal
 - a standard for extensions of Pascal
 - a "very" portable subset of Pascal.

We have been in touch with the Brittish standardization group. We find it very important that an international standard group will be created. Otherwise there will be several national variants of Pascal, which probably will be hard to follow since the computer market is very international.

We would like to get in touch with any group working with some sort of standard and also with implementors and their lists of problems in the definition of Pascal.

Yours sincerely,



The members of the committee are:

Per-Olof Lundberg, L M Eriksson, Kft, Fack
S-431 20 Mölndal
Hans Lunell, Informatics lab, Linköping University,
S-581 83 Linköping
Lars Mossberg, Volvo Flygmotor, Box 136,
S-461 01 Trollhättan
Bengt Nordström, Dep:t of Computer Sciences, CTH, Fack,
S-402 20 Göteborg
Staffan Romberger, Dep:t of Computer Sciences, KTH,
S-100 44 Stockholm
Åke Wikström, Dep:t of Computer Sciences, CTH, Fack,
S-402 20 Göteborg

Yet another attention list

Swedish Technical Committee on Pascal
Department of Computer Science
Fack, CTH
S-402 20 Göteborg, Sweden

3. Terminology: This section should cover all relevant terms and expressions used in the definition of the language (associate, identical, correspond, denote, undefined, scope...). These should be defined, explained and/or listed for easy reference.

Special symbols:

nil is here listed as a reserved word.

It is however conceptually a predefined constant comparable to "true" and "false". The ETH-compilers view in fact nil as a predefined constant and you may redefine it.

Comments may not always be removed! They should be considered as equivalent to space instead.

4. Identifiers: Add "programs", "fields and tagfields in record", "values of scalar types" to first sentence (cf. Burnett-Hall).

The phrase: "Their association..." is neither clear nor sufficient. What type of association? What about record definitions and with-statements?

6. Terminology: The terminology for description and classification of types and their properties is obscure to say the least (cf. below).

... associates an identifier with the type. array A of B is a type. What is the associated identifier?

- 6.1.1. Scalar types: This term is in the subsequent used to denote at least three different things:
- all simple types;
 - all simple types except real;
 - the types that are described in this section.
- UM-5A (p 34) states that they are given the values 0,1 etc.

Real: Is real a scalar type? Note that scalar types are "ordered set(s) of values by enumeration"!

- 6.1.2. Standard types: Are they predefined scalar types or predefined simple types?
- Char: UM-2D (p 14 f) also describes some properties that are shared by all implementation (cf. also Axiomatic Definition).
- 6.2.2. case <tag field><type identifier>: Certainly not every typeidentifier.
- 6.2.4. file of <type>: Restrictions on type should be clearly stated.
Can a file be a part of other structured types?
Can new be applied to a file?
8. Sequences of operators: The expression a+b+c may be evaluated (b+a)+c, but not as a+(c+b). What about a*b+c*d?
- 8.1.2. Is type-checking of a subrange done only in assignments or in arithmetic too? Is pred(succ(i)) always legal, where var i: 1..10? (See also the axiomatic definition: §1.3-1.4)
- 8.1.3. The syntax for <simple expression> is erroneous. The possibility <adding operator><term> allows you to write (for a Boolean b): or b.
- 8.1.4. Pointers: = and <> on pointer variables should be listed here.

Sets: = and <> defined in UM-App B (p 108).
- 8.2. Function application: There are several things left undefined here. For example, in what order is the actual parameters (evaluated and) bound?. Especially in connection with functions one has to consider side-effects, global variables etc.
- 9.1.3. What is the significance of leading zeroes in a label?

- 9.2.2.2. Why not change the syntax to:
`<case list element>::=<case label list>:<unlabeled statement>`

It is not good programming practice to jump between case list elements.
- 9.2.3.3. The manual and the report is not in agreement about the semantics.
- 9.2.4. It can be argued that the with-statement evaluates its <record variable list> only once. This is part of a more general problem: The philosophy of Pascal is to leave certain things undefined so that a program which relies on these undefined constructs is not portable. It is assumed that such a program is poorly written. But is this the right way to prevent people from making bad programs? It is usually not the same person who writes a program and who transports it. The wrong person is punished.
- 10.1, Should passing of predefined procedures and functions
11.1. as parameters be prohibited?
- 10.1.1.(1)What happens to eoln at reset of a text file?

(2)It has been suggested that reset should not assign the value of the first element of the file to the buffer variable.
- 11.1. Whenever possible, define typerules by a corresponding <function heading> or several in the case of generic functions.
- 11.3. What are the types of the result of trunc and round?
- 11.3. How is ordinal number defined? With
`type colour = (blue,red,yellow);`
`warmcolour = red..yellow;`
`var c:colour; w:warmcolour;`
`begin c:= red; w:= red;`
what is ord(c) and ord(w) and what is ord(-1)?
- 11.1.4. Should succ of the last and pred of the first value of a scalar type be explicitly illegal?



INSTITUTE FOR INFORMATION SYSTEMS

MAIL CODE C-021
UCSD
LA JOLLA, CALIFORNIA 92093

30 Dec, 1977

To Andy Mickel
PASCAL User's Group

Subject: Standardized PASCAL Extensions

As discussions in PUGN have made clear, many people in the PASCAL user community feel it mandatory that PASCAL be extended in various ways, either for specific applications, or to make the language easier to work with in general. A large subset of the same people are also very much concerned because all the extension activity seems to be leading PASCAL into the same multi-dialect state that characterizes BASIC.

You and others have ventured the opinion that the evolution of PASCAL has already progressed beyond the stage where there is much hope of obtaining formalized standards, via ANSI or ISO, for more than the language described in Wirth's revised "Report". One U.S. representative to the ISO language standards activity told me recently that it is not unusual for an item to take 3 to 5 years just to be put on the agenda for ISO consideration. Those who cite the short (two years) time it took to standardize MUMPS should recognize that MUMPS is used primarily by a close knit user community concerned with a fairly small range of applications.

My feeling, shared by many others, is that PASCAL is now being accepted so rapidly as the base language for practical system programming that there is no time for formal standardization to be completed before extended versions of the language come into very widespread use. In that environment, defacto standards are likely to prevail, and there may be many defacto standards as is the case with BASIC.

As an alternative to formal standardization, you have already proposed that there might be formed several quasi-standard extensions to PASCAL covering specific application areas. At a small PUG meeting, held in Seattle during the recent ACM Convention, it became apparent that numerous large industrial firms are preparing to use PASCAL as the base language for serious system programming. The point made repeatedly is that these firms find it necessary to extend PASCAL to make this practical. The nature of the extensions generally falls into several familiar areas which have been mentioned already in the pages of PUGN - particularly random access facilities for disk files. In general, each firm is making its own extensions since there has been no evidence that anyone would come forward to coordinate the extensions in this field of applications. The industrial people said repeatedly that their firms would be making sufficiently large investments in system programs using the extended dialects of PASCAL

(typically in units of Millions of Dollars) that they would find it effectively impossible to retrofit to a standardized set of extensions starting more than 6 months to a year from now.

I am aware that there is a subset of the current PUG membership which feels that PASCAL should be retained by the academic community as a basis from which to study better future languages. They feel that extensive practical applications of PASCAL would prevent this from happening, and argue that PUG should not assist in the effort to make PASCAL a practical alternative to BASIC, FORTRAN or COBOL in the world of computing at large. Judging from the rapid growth of PUG membership among those who wish to use PASCAL for practical purposes, I would guess that the great majority of the membership would (very soon if not now) favor pressing on with the promotion of PASCAL as the next major language. I would also guess that, lacking leadership from PUG or someone else, there will be a wide divergence of opinion on what constitutes PASCAL in this sense.

The PUG membership should also be made aware of another large scale activity that is sure to have a big impact on the PASCAL community, like it or not. This is the project at the United States Defense Department currently known as "Ironman". The ambitious goal of this project is to force all development of so-called "embedded" system programs to be done in a common programming language that we can refer to as "DoD-1". The range of programming activities currently excludes the major business applications, which tend to use COBOL, and scientific applications, which tend to use FORTRAN. Embedded system programming is reputed to cost DoD more than \$3 Billion per year, much of it redundant or inefficient. The expectation is that common use of the new language will make this activity more efficient by a significant amount. At present the Ironman project is waiting for delivery, expected in February 1978, of reports on preliminary language designs from four contractors all of whom are working from an overall specification published by DoD last year. All four are reported to be using PASCAL as their base language. The overall specification makes it appear very likely that DoD-1 will differ slightly from PASCAL within the range of the base language, and it will contain many important extensions very similar to those already being discussed for industrial and commercial system programming. If all goes well, DoD expects to start implementing early in 1979, after a round of additional refinements based on the reports due this year.

It is probable that any possible quasi-standard extended PASCAL, as discussed in this note (I'll call it PASCAL-X from here on), will differ in some respects from DoD-1. If the Ironman 1977 specification is held in the final DoD-1 product, there will be no subset or superset languages. If our experience with code compression is a valid guide, this will probably rule out the use of DoD-1 for interactive program development on small microcomputers such as those we use at UCSD. On the other hand PASCAL-X might amount to a large subset of DoD-1 and still be implemented on the micro's. A visit to

DoD in December convinced me that they too are very much interested in using stand alone micro's for interactive program development. Apart from the size issue, the major differences that will probably make DoD-1 incompatible with the PASCAL base language have to do with tightened rules to help prevent side effects committed within functions and FOR loops. A persistent rumor that I have not yet been able to check is that Niklaus Wirth himself has had a hand in recommending that these rules be included in DoD-1, and/or in at least one industrial firm's version of extended PASCAL.

At the Seattle ACM meeting, and on several occasions since then, I have asked representatives of large industrial firms interested in using PASCAL for serious system programming whether their companies might be willing to participate in a 'workshop' convened to seek a consensus on PASCAL extensions among such firms. The implication would be that the influence of such a group of firms would be great enough to make the consensus language, PASCAL-X, a defacto standard for those interested in PASCAL for practical system programming. The verbal responses I have received so far have been enthusiastic and affirmative.

Lacking an alternate invitation, we propose to convene such a workshop here at UCSD during the forthcoming summer, probably for several weeks during July. In order to have a reasonable expectation that the workshop will indeed emerge with consensus on a substantial range of extensions, it will be necessary to limit the attendance to a maximum of about 30 people. Attendance will be by invitation only. Those attending from industrial firms, and from government agencies, will be asked to pay a registration fee of several hundred dollars to help pay the expenses for running the workshop. In addition to paying the fee, these organizations will be asked to give credible assurances that the participant(s) they send will be able to influence their employers to use the resulting consensus language. Approximately three quarters of those attending should be from organizations which will have made investments in practical system programming uses of extended PASCAL exceeding 10 person-years by the end of 1978. We would hope to attract a small number of academic experts who enjoy the widespread confidence of large subsets of the remaining PASCAL user community. The aforementioned fee will cover the expenses of these experts, and of a small UCSD group (mainly students) who will serve as the staff of the workshop.

As a plan of action for the workshop, we propose that those who expect to participate should begin circulating position papers and proposals regarding features they wish to see included in the expected consensus language PASCAL-X. Though we have no formal commitment from them, it appears likely that the Defense Department group will allow us to use the language descriptions resulting from the Ironman project as part of the set of position papers. We assume that the Ironman papers will allow a moderately accurate projection to be made regarding the description of the expected DoD-1 language. Assuming that our conjecture is correct, that DoD-1 will amount to an

extended version of the PASCAL base language with few incompatibilities, we will urge the workshop group to make PASCAL-X as compatible as possible with DoD-1. Naturally, we will insist that PASCAL-X remain faithful to the overall philosophy embodied in Niklaus Wirth's original design. Wherever incompatibilities with the base PASCAL language seem frivolous they will be strongly discouraged.

The workshop will probably consist of relatively short plenary meetings on most days in the weeks when it is in session, plus smaller working group meetings on specific topics. I plan to chair the plenary meetings, as convener, in the hope that I may be able to keep the attending group focussed on the objective to reach consensus on as large a set of extensions as seems important to most of the group. To assure reasonable acceptance of the results of the workshop, most decisions will have to be reached by consensus - i.e. with the acquiescence of virtually all of those attending. Except for our own extensions associated with the pre-declared type STRING, and with the READ statement for interactive implementations of the INPUT file, we at UCSD still have relatively small investments in the specific syntax of most other changes or extensions to PASCAL. We will circulate position papers regarding these matters.

Copies of this note will be going directly, with invitations, to representatives of the firms that we already know are likely to be interested in participating in the workshop. Since we may not know about others, readers of this note in PUGN should feel free to contact us. Please bear in mind that we are a small group, and currently close to saturated for communications with others due to the high interest in our software package. For a reader who thinks that his/her firm should be on our invitation list for the workshop, a brief letter explaining why would probably be the most effective way to get started.

Kenneth L. Bowles
Professor, Director IIS



Implementation Notes

GENERAL INFORMATION

A number of short comments are in order about this issue and our editorial policies in general:

- **INDEX:** The index near the end of this issue covers the Implementation Notes from last issue (#9-10) and this issue. Earlier issues are not referenced because: (1) they are out of print, and (2) the information in them was summarized and updated in #9-10.
- **Corrections:** Unless otherwise stated, the information in this issue supercedes the information in #9-10. We received several corrections (and complaints) about incorrect information in #9-10. As a policy, we print the information that we have available to us. Although this sometimes leads to confusion, we have found that printing incorrect information causes people to send corrected material -- when they might not have done so otherwise.
- **Software Tools, and Applications:** At the suggestion of Rich Cichelli we start with the next issue to print the source code of programs which are known to be useful in the practice of writing software. Rich will be editing this section, and programs may be submitted to him for consideration. We also encourage criticism and comments on these programs. Rich's address is: 901 Whittier Drive, Allentown, PA 18103, USA. Also, Tom Tyson (DECUS SIG) has offered to distribute software at cost. Details are not yet clear to us, but Tom's offer is quite encouraging.
- **Pascal Variants:** We have decided as a policy to print notices of machine-dependent implementations of Concurrent Pascal, Modula, etc., in the "Machine Dependent Implementations" section. Also, the "Index" will not discriminate between the variants of Pascal. Notices of general interest will continue to appear in the "Pascal Variants" section.
- **Checklist:** When submitting implementation notes please use the Checklist (#9-10, page 60), and send dark camera-ready copy (see Policy, page 2!). As we begin to concentrate more on applications, standards, and software tools, we will be less willing to rekey material which is not properly prepared.

- Jim Miner (78/1/5)

APPLICATIONS

HELP WANTED!

If PASCAL is to make any inroads into serious scientific computing (currently the almost exclusive preserve of FORTRAN) it must have a decent library of scientific subroutines - which means, as far as the U.K. is concerned, that there must be a PASCAL version of the NAG (Numerical Algorithms Group) library.

It should be possible to make a PASCAL NAG library largely machine-independent with all machine-dependent features being collected into the "X" routines. Probably the easiest method of production of the library would be a straight transcription of the existing ALGOL 60 versions, together with the writing of the set of "X" routines for each different range of machines.

Please send your views on this matter, and offers of help, to:

Professor D.W. Barron,
Computer Studies Group,
Department of Mathematics,
The University,
Southampton, Hants, SO9 5NH, United Kingdom,

who is coordinating this project and negotiating with NAG.

PORTABLE PASCALS

Pascal P4 -- How (Non-) Standard is it?

Some of us at Pascal News have seen a disturbing trend recently to label P4-based implementations as "Standard Pascal" simply because they are based on P4. In fact, P4 differs from the standard in a number of significant ways. We have compiled the following list of deviations, based on a list sent to us by Ted Park (Director, Systems Development; Medical Data Consultants; Suite 302; 1894 Commercenter West; San Bernardino, CA 92408), and also based on our own experience with P4. In no way do we intend this list as a criticism of the authors of P4; rather we hope to raise the awareness of implementors using P4.

-Jim Miner

1. P4 implements nil as a predeclared constant, and forward as a reserved word. The standard indicates that nil is a reserved word, and forward is not listed as a reserved word.
2. P4 does not support the standard comment delimiters { and }.
3. P4 does not provide the standard predeclared identifiers maxint, text, round, page, or dispose. Further, the following standard predeclared identifiers are recognized, but are flagged as errors: reset, rewrite, pack, and unpack.
4. P4 does not require a program heading. Further, where a program heading is included, P4 does not require it to contain a parameter list.
5. P4 does not allow "non-discriminated variant record types"; i.e., every variant record must have a tagfield. The standard does not require a tagfield.
6. P4 does not allow a ";" before the "end" in a record type. (See the P4 Bug list, item 3.)
7. P4 does not implement any of the following file-related features:
 - Declarations of file types, variables, and parameters.
 - The standard predeclared type text, and the standard predeclared procedures reset, rewrite, and page.
 - The requirement that standard files input and output must appear in the program header if they are used.
 - Access to non-text files via the standard procedures read and write.
 - Output of Boolean expressions, or output of real expressions in the fixed-point form (r:ex1:ex2) via the procedure write.
8. P4 does not support formal procedure or function parameters.
9. P4 does not allow set constructors containing the subrange notation (e.g., ['0'..'9']).
10. P4 does not support goto statements which jump out of the procedure or function in which they occur. (In fact, P4 has a bug wherein it fails to diagnose such goto's, and treats them like "local" jumps -- see the P4 Bug list, item 6.)

Pascal P4 -- Bug Reports.

We have received several reports of bugs in P4, in addition to Updates 1 and 2 which were printed in Pascal News #8. Since Zurich has not promised support on P4 we intend to print such reports here, including fixes when possible. Also, to make sure that they are widely known, we are reprinting Updates 1 and 2 from Chris Jacobi. These should already be included in the P4 distribution tapes.

The following list is based on bug reports from: Juha Heinanen (Computer Center; University of Tampere; P.O.Box 607; SF-33101 Tampere 10; Finland; (931-156111)), O.W. van Wijk (TNO-IBBC; P.O.Box 49; Delft, Holland; (015-138222)), and Jim Miner (Pascal News).

P4 bugs not affecting portability.

1. Disallow zero-length string constants.
Insert after PASC.461:
 IF LGTH = 0 THEN ERROR(205)
 ELSE
2. Assure non-compatibility of different length strings.
Insert after P.145:
 AND (FSP1^.SIZE = FSP2^.SIZE)
3. Allow ";" between field list and "end" in record type.
On line PASC.1079
 change: IN [IDENT,CASESY]
 to: IN FSYS + [IDENT,CASESY]
4. Correct comment.
On line P.307
 change: (*L0D*)
 to: (*LDO*)
5. Correct generation of set constants.
On line PASC.1772
 change: := 0 TO 58 DO
 to: := SETLOW TO SETHIGH DO
6. Correct failure to disallow non-local goto's.
Insert after PASC.2893:
 WHILE DISPLAY[TTOP].OCCUR <> BLCK DO TTOP := TTOP - 1;
 TTOP1 := TTOP;
Replace PASC.2895 and PASC.2896 with:
 LLP := DISPLAY[TTOP].FLABEL;
7. Correct comment.
On line P.500
 change: (*IJP*)
 to: (*UJP*)
8. Allow label definition inside of a with statement.
On PASC.3153
 change: DISPLAY[TOP]
 to: DISPLAY[LEVEL]
9. Avoid spurious "undefined forward type" diagnosis.
Replace PASC.1368 and PASC.1369 with:
 END
 ELSE LCP2 := LCP1;
 LCP1 := LCP1^.NEXT
10. Correctly diagnose "read(f)".
On P.347
 change: = 8 THEN
 to: = 5 THEN

11. Correctly diagnose "write(f)".
On P.380
 change: = 10 THEN
 to: = 6 THEN
12. Correct error numbers.
On PASC.2277 and on PASC.2285
 change: ERROR(125)
 to: ERROR(116)
13. Write end of line on last line of listing, and also flag errors in all cases on the last line.
Replace P.555 with:
 IF LIST THEN WRITELN(OUTPUT);
 IF ERRINX > 0 THEN
 BEGIN LIST := FALSE; ENDOFLINE END
14. P4 does not diagnose forward-declared procedures and functions which are not actually defined. No fix has been submitted for this bug.

P4 portability-related bugs.

The items listed here involve implementation dependencies.

15. Correct declaration of set values.
On PASC.85
 change: SET OF 0..58
 to: SET OF SETLOW..SETHIGH
16. Diagnose set declarations exceeding implementation defined limits.
Replace PASC.1275 to PASC.1278 (inclusive) with:
 IF LSP1 <> REALPTR THEN
 IF LSP1 <> INTPTR THEN
 BEGIN GETBOUNDS(LSP1,LMIN,LMAX);
 IF (LMIN < SETLOW) OR (LMAX > SETHIGH) THEN
 ERROR(169);
 NEW(LSP,POWER);
 WITH LSP^ DO
 BEGIN FORM := POWER; SIZE := SETSIZE;
 ELSET := LSP1
 END
 END
 ELSE ERROR(169)
 ELSE ERROR(114)
17. On some implementations it is not safe to allow tests of (in-) equality on arrays and records. This is because P4 does not guarantee that all storage units within an array or record type are accessible to the programmer, due to alignment considerations. The following changes disallow such comparisons, except for strings.
Replace PASC.2826 and also replace PASC.2831 with:
 ERROR(134);

PASCAL - P4 Installation Parameters

intsize,realsize,charsize,boolsize,setsize,ptrsize:
Number of addressable storage units to be reserved for variables of type integer, real, character, boolean, set, pointer. As to 'setsize', remember that a set must be able to hold at least 48 elements if you intend to use the system to bootstrap the compiler.

intal,realal,charal,boolal,setal,ptral:
Variables of the corresponding types will be given an address which is a multiple of these alignment constants.

Implementation Notes

stacksize:

Minimum size for a value on the expression stack. The expression stack is that portion of the stack which is used for the evaluation of expressions. 'Stacksize' has to be equal to or a multiple of 'stackal'.

stackal:

Alignment constant for a value on the expression stack. 'Stackal' must be a multiple of all other alignment constants and must be less or equal to 'stacksize'.

strlgth:

Maximum length of a string. (In fact all strings will be of length 'strlgth'). A string must be able to hold the character representation of a number (real or integer) with its sign. The minimum length for a bootstrap is 12.

intbits:

Number of bits used for representing an integer without the sign. So the largest integer is: 2^{intbits_1}

sethigh,setlow:

Maximum and minimum ordinal values for the element of a set.

ordmaxchar,ordminchar:

Maximum and minimum ordinal values of the character set.

Depending on the alignment conditions there may be two possibilities for the assignment of store on top of the expression stack:

- Each stack element requires the same amount of store. In this case 'stacksize' has to be greater than or equal to the maximum of the other size constants. (Remember: 'stacksize' is a multiple of 'stackal').
- No waste of store: A new element on the expression stack has to be placed at the next position allowed by the alignment constant 'stackal'. In this case 'stacksize' has to be less than or equal to the maximum of the other size constants.

Thanks to George Richmond for sending us revised Pascal-P ordering information.

Pascal-P may be ordered from:

- In Europe, Asia, and Africa: Chris Jacobi, Institut fuer Informatik, ETH-Zentrum, CH-8092 Zuerich, Switzerland. (*last published cost was SFr 160 for configured compiler - do not prepay*)
- In Australasia: Carroll Morgan, Basser Dept. of Computer Science, Univ. of Sydney, NSW 2006 Australia. (*last published cost was \$A30 *)
- In North and South America: George Richmond, Computing Center: 3645 Marine, Boulder, CO 80309 USA. (*new prices: \$60 for tape, documentation, and overhead, please prepay, and \$30 additional for configured compiler.*)

PASCAL VARIANTS

Pascal-S

(* See the Checklist and letter from Rich Cichelli under CDC 6000 in the Machine Dependent Implementations section. *)

Concurrent Pascal

COMPUTER SCIENCE DEPARTMENT
SALVATORI COMPUTER SCIENCE CENTER

22 September 1977



UNIVERSITY OF SOUTHERN CALIFORNIA, UNIVERSITY PARK, LOS ANGELES, CALIFORNIA 90007

Dear Colleague:

I am pleased to announce that the distribution of Concurrent Pascal tapes has been resumed. The two tapes contain copies of the Solo Operating System and the Sequential and Concurrent Pascal compilers. The system is ready to run on a PDP 11/45 system and can (with some effort) be moved to other minicomputers.

To obtain the system tapes, please contact

Mr. George H. Richmond
Computing Center
University of Colorado
3645 Marine Street
Boulder, Colorado 80309

Concurrent Pascal and three model operating systems written in the language are described in

P. Brinch Hansen, The Architecture of Concurrent Programs.
Prentice-Hall, Englewood Cliffs, New Jersey, July 1977.

The compiler is described in

A.C. Hartmann, A Concurrent Pascal Compiler for Minicomputers.
Lecture Notes in Computer Science 50, Springer-Verlag, New York, N.Y., 1977.

I am interested in hearing about any experience you may have had in using Concurrent Pascal.

Yours sincerely,

Per Brinch Hansen

CONCURRENT PASCAL DISTRIBUTION George H. Richmond September 1977

Concurrent PASCAL is a variant of PASCAL developed by Per Brinch Hansen while he was at the California Institute of Technology. This system is implemented for the PDP 11/45. It includes the Solo Operating System, the Sequential PASCAL Compiler, and the Concurrent PASCAL Compiler. The software supplied is ready to run for a suitably configured PDP 11/45 system. It could be transported to other minicomputers as the bulk of the code is written in PASCAL.

The University of Colorado is distributing Concurrent PASCAL in cooperation with Per Brinch Hansen following the publication of his most recent book "The Architecture of Concurrent Programs" by Prentice-Hall in July 1977. This book describes the Concurrent Pascal system. It is not included in the documentation distributed by the University of Colorado.

The materials available include two magnetic tapes containing a complete copy of the Solo Operating System for loading onto a RK05 disk pack on a PDP 11/45 computer and a source and virtual code copy of the Solo programs for listing on any computer. Documentation includes two reports not included in "The Architecture of Concurrent Programs" and other items or suitable replacements as follows:

TRW

TRW DSSG
One Space Park
90/2178
Redondo Beach, CA 90278
(213) 535-0312
19 October 1977

Dear Andy:

The Multi-Minicomputer Architecture IR&D Group at TRW, headed by Roger A. Vossler, is using Per Brinch Hansen's Concurrent Pascal, to write special-purpose operating systems for distributed computing research. Although Hansen's Concurrent and Sequential Pascal compilers generate virtual code for an ideal machine which is implemented by an interpreter, we find that for some applications this approach can compete successfully with real machine code running under a general purpose operating system. One of the advantages of Concurrent Pascal is that its compile time enforcement of access rights eliminates many potential time-dependent run time errors.

We have developed a number of utility programs to supplement those available on the SOLO distribution tape. Other work has involved writing device drivers for additional IO devices, improvements to the compilers, and an experimental kernel with all IO drivers removed and rewritten in Pascal. Future plans include moving the interpreter and parts of the kernel to microcode on machines with WCS, such as the PDP-11/60.

Although we are unable to serve as a distribution center, we are interested in exchanging ideas and programs with other users of Concurrent Pascal. An Implementation Checklist for our implementation is attached.

Sincerely,

Bill Heidebrecht

JBH:nc

J.B. Heidebrecht

(* The Checklist mentioned is listed under DEC PDP-11 (Redondo Beach) in the Machine Dependent Implementations section. *)

UNIVERSITY OF YORK

HESLINGTON, YORK, YO1 5DD
TELEPHONE 0904 59861

Modula

DEPARTMENT OF COMPUTER SCIENCE
Professor of Computer Science: I. C. Pyle

13 September 1977

Dear Mr Mickel,

A MODULA Compiler for the PDP-11 computer

A recent copy of the Pascal Newsletter asked for details of any compiler for the programming language MODULA. We are writing this letter to inform you and the readers of the Newsletter about the existence of a compiler written at the University of York and completed in the spring of 1977.

The York MODULA compiler is written in BCPL and is structured in four passes using a sequential binary stream for communication between the passes. The present version of the compiler runs under RSX-11D and requires a 16k partition for the compilation of a program of about 1000 lines. The compiler output is MACRO-11.

All of the MODULA language as defined by Wirth has been implemented with the following exceptions.

- (i) All names must be declared before use, thus mutually recursive procedures are not possible,

- (ii) The VALUE statement for presetting global variables has not been implemented,
- (iii) The procedures "off" and "among" have not been implemented. Their effect can be obtained by other means within the language.

The following run-time systems are available.

- (i) A nucleus to allow MODULA programs to be run on a base PDP-11/40. Including a "loader" to overwrite the operating system, the nucleus requires about 150 words of code,
- (ii) A similar nucleus for the PDP-11/05, including the out-of-line routines for division and multiplication, requiring about 200 words, and
- (iii) A simple system for running "sequential" MODULA programs under RSX-11D.

The compiler and its run-time system have been tested using:

- (i) A structured test set of about 100 programs to test the sequential parts of the language, and
- (ii) Many device driving MODULA programs (including those given by Wirth) for testing the full language. We have device drivers written in MODULA for the DL-11, KW-11L, RK-11, AR-11, GT40, CA-11F and many other DEC devices.

So far the compiler has not been distributed to any other institution. At the present time our efforts are directed towards providing full external and internal documentation of the compiler (we hope to produce a document similar in style to Hartman's description of the Concurrent PASCAL compiler) and putting the present RSX-11D system into a presentable package. Our present development work is:

- (i) Providing a cross compiler for the INTEL 8080 that runs on the PDP-11/40. An initial version of the code generator and run-time system is now complete. Our initial conclusion after development of this version is that the 8080 is unsuitable for an efficient version of MODULA, and
- (ii) Making the compiler run under UNIX.

We have just applied to the Science Research Council of Great Britain for funds to support a programmer who would look after the MODULA compiler maintenance and distribution. Until we know the result of this application we will not know what distribution scheme we will adopt. In the meantime, we would like to hear from any PASCAL Newsletter reader who would be interested in receiving the compiler so that we may gauge the interest in this work. We would also be interested in knowing what areas of application potential users have in mind.

Our initial conclusions on MODULA, together with further details on the compiler, are given in

J. HOLDEN & I. C. WAND: 'Experience with the programming language MODULA'.

A paper presented to the 1977 IFAC/IFIP Real Time Programming Workshop held at Eindhoven, Netherlands, on 20-22 June 1977. To be published by Pergamon Press.

We have a very limited number of preprints of this paper available.

Yours sincerely,

J. Holden
J. Holden

I. C. Wand
I. C. Wand

FEATURE IMPLEMENTATION NOTES

UNIMPLEMENTABLE FEATURES - WARNING

This note addresses some features I have noted others adding to their PASCAL compilers which I condemn as particularly nasty because they are not implementable on the Burroughs B6700 system and possibly on other computers. One of PASCAL's great potential features is that of portability. I make a plea to everyone not to introduce features that are not implementable on all computers, and to remove any that presently exist.

(1) Passing pointer values as addresses, even in-stack.

Seen in PN #9/10, page 40. The following program is allowed to execute, storing the (stack) address of x in p:

```
program P;
  var p : ↑ integer;
      x : integer;
begin
  x:=1;
  p:=↑x;      {the horror}
end.
```

Since the concept 'address' does not exist in some computers, and notably not in the Burroughs B6700 in the sense used here, this is virtually unimplementable. Stack-addresses are generally not computable segment addresses have two components; and any contortion made to try to implement this feature will cause either serious security risks, or explode all pointers into requiring two words of 48-bits for storage (with a time penalty too), or destroy the structure of B6700 programs making them CDC-like. Pointers are not necessarily addresses; a point which seems not to be widely realised.

The restriction in the PASCAL report was there for a reason: leave it there.

The feature should not exist, or should only be available when specifically enabled by a computer directive.

(2) Returning function values of all kinds except files

The Report allows one to define functions which return values which may be:

- * scalars of all types,
- * reals, and
- * pointers.

It has been proposed to extend this permission to all types except file-type. (PN 9/10, page 48). The omission of file-types is a blessing, but the others were also omitted for a definite purpose. Apart from the invention of array and record temporaries on the PASCAL run-time stack (which do not otherwise occur), and the greater difficulty of distinguishing recursive calls from com-

ponent accesses if regularity is attempted, this feature, too, is virtually unimplementable on machines which use descriptor-based memory organization, and in particular on the Burroughs B6700.

Arrays and records in this machine are not stored on-stack, but are allocated memory on demand, this memory being described by an on-stack descriptor word. In the B6700, this descriptor is memory protected by its tag-field (an extra 3-bit memory field), and the 'RETURN FROM FUNCTION' instruction (RETN) does not recognise a descriptor as a legal value to return: it expects an operand value as represented by an integer, real, etc (with tag-field = 0). Consequently attempts to return such values will always crash due to the 'INVALID OPERAND' interrupt.

The only way around this problem that does not totally destroy the architectural features of the B6700 (thereby slowing it manyfold), is to:

- (a) turn interrupts off,
- (b) shift the local PASCAL stack up by one word,
- (c) adjust the dynamic and static stack activation record pointers to correspond to the shift,
- (d) implant the return descriptor into the caller's stack in the free word created under the make-stack word, and
- (e) turn interrupts back on.

I have omitted some important features which maybe could be resolved, like how the returned memory get de-allocated, but this should suffice to show that this is really impossible.

(3) Allowing pointers to file-types and the use of new(file)

Observed in correspondance, on Univac PASCAL. The following will compile:

```
program P;
  var p : ↑ file of char;
      c : char;
begin
  new(p);
  reset(p↑);
  read(p↑,c)
end.
```

And it will execute. In the Burroughs B6700, file-descriptor-blocks (the operating system's status area for a real file) must reside in the task's stack. They cannot reside in some other area such as the heap as they will not then be closed and de-allocated by the operating system's block-exit procedure, and the ensuing mess will be horrible to contemplate. Moral: beware what you do with files.



THE CASE FOR A "BOOLEAN OPERATOR" INTERPRETATION

In some recent correspondence, I became aware that there might be some argument about the correct way to implement boolean expressions in PASCAL. This argument arises partially because the "standard" documents of PASCAL do not say the same things, and partially since the consensus area is perceived by some to be "restrictive". I wish here to argue two points:

- * that the consensus area outlined in the "standard" documents should be preserved, and
- * that a "boolean operator" interpretation be regarded as the generally preferred implementation method for evaluating boolean expressions.

The problem area

The essential problem is related to complex boolean expressions, for example:

(a < 0) or (b > c)

Are both relational expressions evaluated, followed by a logical or-ing of the result (the *boolean operator* interpretation), or is the first factor evaluated and only if it gives false is the second factor evaluated (the *sequential conjunction* interpretation)? In the example I give, if a, b & c are all simple variables, the two evaluations give identical results. If however there could be side-effects, whether intentional ones due to function evaluations, or unintentional ones due to undefinition of values, then the two possible implementations may give different execution results. Which is an implementor to choose?

The "standard" documents of PASCAL

The Revised Report does not specify the semantics of an expression, except by a rather remote implication. It simply states the syntactic rules for expressions, prefaced by an introduction which says:

*"Expressions are constructs denoting rules of computation for obtaining values of variables and generating new values by the application of operators.
 Sequences of operators of the same precedence are executed from left to right."* (Revised Report #8)

There is little semantic guidance here, except to imply that all operators are intended to be applied.

The User Manual, our second "standard" document is thankfully more explicit. It says:

"Boolean expressions have the property that their value may be known before the entire expression has been evaluated. Assume for example that x=0. Then

(x > 0) and (x < 10)

is already known to be false after computation of the first factor, and the second need not be evaluated. The rules of PASCAL neither require nor forbid the evaluation of the second part in such cases. This means that the programmer must assure that the second factor is well-defined, independent of the value of the value of the first factor. Hence if one assumes that the array a has an index ranging from 1 to 10, then the following program is in error!

x:=0;

repeat x:=x+1 until (x > 10) or (a[x] = 0) "

(User Manual, pp 20-21)

While this comment only covers some of the possibilities, it is quite explicit as far as it goes. If you accept the authority of the User Manual, implementations are at liberty to choose the *boolean operator* or the *sequential conjunction* approach. (Perhaps this freedom is regrettable, but that is a question of language or standard criticism.)

This would seem to dispose of my first question: the consensus area reserved for implementors in the standards. Now, which is preferable? This is a quite difficult question (impossible to resolve in abstract), and hinges around three points:

- * regularity,
- * efficiency, and
- * portability.

Regularity

One of the guiding principles of PASCAL is parsimony of concepts. In general, one expects expressions to be evaluated as they are written, and this is the practice for all expressions in existing PASCAL compilers, apart from the cases we are now considering. Regularity therefore suggests that treating boolean operators just like all other operators (both left and right side operands are evaluated) is the more regular approach. Certainly a naive student learning how to program is likely to assume this when tracking down an elusive bug, though a programmer conditioned by exposure to other languages might not. Particularly is this the case if the boolean expressions become even more complex than the ones I or Wirth give.

Time-efficiency

The evaluation of a complex boolean expression might be quite time-consuming. Obviously, minimizing the number of things to be evaluated will always pay off for sufficiently complex expressions, more especially if function evaluations are involved. Just where the break-even point comes (if it exists) depends upon whether the boolean expression is in a *jump context* as in if, while or repeat, or in a *value context* where it must deliver a boolean value. It also depends on the machine architecture of the target computer.

For example the sample expression I gave first is faster to evaluate on a Burroughs B6700 in toto, than by sequential conjunction. This arises because conditional branches flush the instruction buffers (lookahead), and delete the top-of-stack value. On a DEC PDP-11, this would be reversed. On the other hand, involving a function evaluation in the second factor as in

(gtyptr = realptr) and compatible(gtyptr, ltyptr)

would always be faster evaluated by the sequential conjunction method.

Space-efficiency

Unlike speed, which tends to be dominated by the properties of a small proportion of a program, space occupied by code depends upon the properties of all the code. Since typical PASCAL programs have a fair number of boolean expressions, compactness is also a desirable property of a compiled boolean expression evaluation. Here the decision could go either way, depending on the machine architecture, or could exhibit a cross-over (as with the Burroughs B6700's time-efficiency). Again to give examples, the B6700 always requires less code space for the boolean operator interpretation (the most extreme case "a:=b and c" requires 8 bytes or 18 bytes for the two approaches), and the reverse situation holds in the DEC PDP-11.

Portability

Any implementor must be cognizant of the effects of a compiler on the likely portability of the programs written under its umbrella. Is there a preference here? The answer is fortunately relatively clear: a compiler that implements boolean expressions just like all other expressions is likely to lead to the detection of more illegal PASCAL constructs (like that instanced by Wirth) by run-time errors before they leave the system supposedly debugged. Since part of the role of a compiler is detecting illegal constructs (as well as compiling legal ones), the *sequential conjunction* interpretation can be seen to reduce the portability of PASCAL programs.

Existing compilers

I cannot speak for all compilers, because I don't have access to them all. Of those I know, we have:

<u>boolean operator</u>	<u>sequential conjunction</u>
CDC 6000 series	ICL 1900
PASCAL-P (all of them)	
Burroughs B6700/7700	

Other factors

I have said before that PASCAL does not have a good standard, and this can be illustrated here too. Nowhere in the "standard" documents is it required that a compiler which implements sequential conjunction evaluate the leftmost factor first. I don't think any existing compiler is that clever (they are largely one-pass) but an optimizing compiler might quite easily decide to evaluate a simpler second (or third) factor before the leftmost one. It seems reasonably certain that some PASCAL compiler will attempt it if the language has a reasonable future to go.

Summary

	boolean operator	sequential conjunction
regularity	better	-
speed	?	better for complex expressions
space	?	?
portability	better	-

As happens so often, we end up comparing incomparables: efficiency vs desirable properties. Fortunately for me, the Burroughs B6700 architecture uniformly favours the method with the best properties, and there was no conflict. This will not be so in other computers where a comparison does not give a boolean result, but sets condition codes or some such hardware fudge.

Let me say, that if a compiler is intended for teaching purposes (so that regularity is important and efficiency not), or if the promotion of program interchange is regarded as important, then the choice seems quite clear to me: boolean operators should be implemented as such, and every factor evaluated.

Only in a compiler which is aimed at producing optimized code of high quality would it seem desirable to employ sequential conjunction. I think here of compilers that attempt strength reductions, or optimize loops and sub-expressions. All such rearrangements have their drawbacks and nastinesses. It may however be necessary or desirable to use sequential conjunction in a non-optimizing compiler simply because of the clumsiness of the machine architecture. This should be regretted and regarded as an unfortunate occurrence calling for a complaint to the computer architect!



Professor A.H.J.Sale
Department of Information Science
University of Tasmania

IMPLEMENTATION FEATURE - LONG IDENTIFIERS

PROBLEM

The definition of PASCAL allows identifiers to be of any length, provided they meet the compactness criterion, and are composed of an alphabetic and alphanumeric characters. It is natural therefore for the full identifier to be taken as uniquely identifying some object or concept, and this would normally call for the full identifier to be entered into a PASCAL compiler's symbol table.

Two features spoil this picture. One is the unfortunate implementation in PASCAL-6000 (CDC) of treating only the first ten characters of an identifier as significant, and the propagation effect this decision has had on PASCAL-P and other PASCAL compilers. The second is the statement in the portability note of the User Manual to the effect that identifiers should be chosen to be unique over their first eight characters. These have given rise to a common presumption in the PASCAL community that this regrettable handling is a standard part of PASCAL; I have been exhorted to reduce significant characters in the Burroughs B6700 PASCAL compiler to eight characters!

The problem arises as long as there exist commonly available compilers which treat different parts of an identifier as significant: some PASCAL programs may compile on system A but not on system B, or worse, some may execute differently on system A from system B. These arise from the combination of the Algol scope rules in PASCAL, and the significance conventions alluded to above.

THE BEST SOLUTION

The best solution is quite clear: all existing PASCAL compilers should be modified so that they retain *all* characters of identifiers in the symbol table. In the B6700 compiler this is done, and since the compactness criterion limits an identifier to a single input record, no identifier can exceed 72 characters (the size of the input record).

There are few valid reasons for the retention of the common practices used in current compilers. Probably the most significant argument is one of space, especially in mini- or micro-computer implementations. This is indisputable, but such implementations are often restricted in other ways as well. An oft-met argument is that PASCAL lends itself to fixed-length-strings (due to the type constraints) and that full significance would be very difficult to program.

Piffle! Any competent programmer can get around this with only a fractional increase in complexity, and that confined to three or four small routines (NEXTSYMBOL, ENTERINTOTABLE, SEARCHTABLE, and SEARCHLEVEL). An obvious idea that occurs to me is to chain overflow text in special records in the heap. The loss of speed is likely to be small if the base size of the names is large enough to cover most occurrences.

*RECOMMENDATION 1 (to compiler implementors and maintainers)
Please treat all characters of identifiers as significant, unless you have severe space constraints.*

A PRAGMATIC PATCH

Though the B6700 compiler does implement full significance, there nevertheless remains the problem of how to enable (i) B6700 programmers to modify their programs to achieve the same execution on other systems, and (ii) B6700 programmers who receive software from elsewhere to detect masquerade errors that might be persistent in that software. This difficulty has been tackled by making a change in the procedure that enters identifiers into the symbol table, and making the additional checking conditional on the setting of a compiler option 'STANDARD'.

Under normal circumstances, STANDARD is reset, and the compiler behaves similarly to other PASCAL compilers. If STANDARD is set by the user, the compiler issues warning messages for non-standard features (for example special pre-defined procedures, or ELSE in CASE), and it enables the identifier check.

This check operates as follows: before the identifier is entered into the

symbol table it is truncated to eight characters (if necessary), and the table is searched for a truncated match with this fragment. If no match is found, the entry is made in the normal way. If a match is found, one of a variety of nasty situations has arisen, and additional checks are necessary to identify the reason. The case depends on whether the match has occurred at the current topmost lexical level or not, and on whether a full-significance match of entry and identifier also succeeds or not:

		Lexical level	
		Top	Beneath
Does a full significance match succeed?	Yes	Clash	Redefinition
	No	Sibling	Masquerade

Let me examine each of these cases:

Clash:

This is a blatant direct name clash: every compiler should object to it. The B6700 compiler does nothing at this point as the error will be reported during the subsequent entry of the name into the table.

Redefinition:

No error, simply the Algol scope rules embedded into PASCAL causing an outer identifier to become inaccessible. Few compilers would even comment but the B6700 compiler will issue a 'NOTE' message as this is sometimes the cause of mysterious errors. Redefinition of I-s and J-s will have to be consciously ignored. (Those of a legalistic turn of mind may ponder on whether the portability note in the User Manual prohibits redefinition; an example of how careless wording can be interpreted.)

Sibling:

This is a distinct name which is so treated on the B6700, but will result in a compiler error message ('CLASH') on an 8-character PASCAL. A 'WARNING' is issued at this point.

Masquerade:

The most important case to detect: this may successfully compile on both the B6700 and on an 8-character PASCAL (depending on the objects identified) but would possibly execute with quite different effects. On a full-significance compiler the two names are quite distinct, but on an 8-character compiler the inner one re-defines the outer one in its scope. A 'WARNING' is issued.

In all cases, the normal entry of the full name into the table is then performed.

CAUSES OF PORTABILITY LOSS

There are two basic ways that portability of PASCAL programs can be affected

by this naming problem. One is common to all compilers that regard more than eight characters as significant (e.g. PASCAL-6000), and arises out of the natural tendency to miss possible naming problems on more restricted compilers. The B6700 compiler can detect these for its own offspring before they leave its protection, but it can also serve to detect these flaws in programs that are written under different PASCAL systems.

The second arises from an obnoxious practice that I have seen encouraged by the limited-significance convention. I recall one installation where it was the installation-standard to use the ignored text of an identifier in an Algol system as a heaven-sent opportunity for commentary! The only sure-fire cure for this is to do away with limited significance. Exhortation will do for those who listen, but the industry-at-large needs more definite and obvious constraints.

Recommendation 2 (to compiler implementors and maintainers)
 If your PASCAL compiler treats more than eight characters as significant, make the same checks as described above on identifiers.

Recommendation 3 (to users of PASCAL)
 Before you release a supposedly portable program, compile it on a system that checks for identifier problems (e.g. Burroughs B6700 PASCAL), and exert pressure on implementors and maintainers to eliminate this semantic snag.



INTERIM REPORT - IMPLEMENTATION OF FOR-STATEMENT - 2

This note adds details for the ICL1900 compiler to the set already detailed in Report 1.

ICL 1900 Pascal (test site: University of Southampton)

The implementation is equivalent to:

```

initial:=e1;
final:=e2;
v:=initial;
if (final - initial) > 0 then begin

```

```

repeat begin
  s;
  v:=v+1
end until v > final;
end;

```

The consequences are

(i) the order of execution is e1, e2, assign to v. Identical to PASCAL-6000 and B6700.

(ii) the exit value of v if the loop is never entered is e1.

(iii) the exit value of v if the loop is entered is (e2 + 1).

Additionally, the controlled variable can be altered from within the loop, and this alteration affects the progress of counting.

It is interesting, and sad, to note that all three compilers reported so far have different answers to my questions (ii) and (iii). The uniform answer to question (i) is perhaps a sign for the future...

Professor A.H.J. Sale,
Department of Information Science,
University of Tasmania.

FOR Statement - Robert A. Fraley

IBM 370 (UBC Version)
HP 3000 (Forthcoming)

The program behavior matches that of the CDC 6000 version, as printed in PUGN #9. There are no limits except for word size. (2^{31} on IBM. Currently 2^{15} on the HP, but double word integers may eventually be supported.)

For the IBM version, optimizations are made in several cases:

- If the limit is a constant, it is not stored in temp 2.
- In order of decreasing efficiency, the loop forms are:

downto 1

downto constant

to constant < 2^{24} (with initial value ≥ 0)

to constant

variable limit

MACHINE DEPENDENT IMPLEMENTATIONS

Alpha Micro Systems AM-11

See DEC LSI-11 (San Diego).

Andromeda Systems 11/B



November 21, 1977

Mr. Timothy M. Bonham
D605/1630 S. Sixth St.
Minneapolis, MN 55454

Dear Timothy:

Thank you for your inquiry about our 11/B System. You will find our standard product literature enclosed.

In regard to your specific questions:

- We are considering offering Pascal with the 11/B. No final decision has been made yet.
- We have been in communication with Ken Bowles and his assistant Mark Overgaard. Based on our discussions, we have no reason to believe the UCSD Pascal won't work on the 11/B. (If we were to offer Pascal directly, it would be UCSD Pascal.)
- I'm not familiar with the systems mentioned so I can not comment on them.

If you should have any further questions, please don't hesitate to contact me.

Regards,

Les LaZar
Les LaZar

14701 ARMINTA STREET #J PANORAMA CITY, CALIFORNIA 91402 (213) 781-6000



Heriot-Watt University

Department of Computer Science

37-39 Grassmarket, Edinburgh EH1 2HW
Tel 031-225 8432 031-226 5601 Ext

Head of Department
Professor A Balfour, MA, FIMA, FBCS

your ref
our ref AB/KM

date 30th November 1977.

Mr. A. Mickel,
PASCAL Users' Group,
277 Experimental Engineering Building,
208 Southeast Union Street,
University of Minnesota,
Minneapolis,
Minnesota, 55455,
U.S.A.

Dear Mr. Mickel,

I thought it was about time that I let the PASCAL Users' Group know about the implementation of PASCAL that we are running at Heriot-Watt University on our Burroughs B5700. We have a PASCAL translator that produces a Burroughs XALGOL program. It is based on the CDC 6600 compiler written by U. Amman & R. Schild, and conforms as closely as possible to the definition by Jensen & Wirth. We have been using it for almost 2 years now very successfully. Indeed, one colleague has almost completed writing a MODULA compiler in PASCAL using it. The only major extension to the compiler is a cross-referencing option.

We also have a frequency analyser that can be run in conjunction with the compiler. This is still being worked on to tidy it up, but works very well.

The PASCAL compiler was produced for us by an M.Sc. student from Oslo, Norway called Dag Langmyhr, to whom we are extremely grateful as he continues to maintain it for us. The frequency analyser was written by another M.Sc. student Mike Staker.

Recently we obtained a PASCAL compiler for our Burroughs B1726 from Paul Schultess of the University of Zurich. This has so far proved very successful with no problems. I have extended it slightly to allow the slightly odd character set from our B5700 PASCAL programs to be acceptable to it. We look forward to the implementation of real arithmetic by Herr Schultess.

Anybody wishing any information about our efforts with PASCAL can contact me at the above address. I will be only too pleased to help if I can. We already have several universities using our compiler in the USA, Japan and South Africa.

Yours sincerely,

David A Cooper
David A. Cooper.

Burroughs B6700/7700 (Tasmania)

Arthur Sale sent us a very impressive (about 170 pages) manual for his B6700/7700 implementation: B6700/7700 PASCAL Reference Manual (R77-3; July, 1977). Arthur also sent an interesting report entitled "The Use of Tag Six in a Pascal Compiler" (R77-4), which "discusses the use of words with tags of six in the PASCAL implementation for the Burroughs B6700 developed at the University of Tasmania. It involves questions of language definition, undefined values of variables, and machine design." (From the abstract.) Both of these reports are published by the Department of Information Science, The University of Tasmania, G.P.O. Box 252C Hobart, Tasmania 7001. (* Thanks Arthur! *)

Burroughs B6700/7700 (San Diego).

In a letter dated 3 November 1977, Thomas J. Kelly, 58-B Meadowlake Drive, Downingtown, PA 19335, wrote:

"I've changed jobs: I now work for Burroughs Corp. I exerted some effort and we are now using the UCSD implementation of PASCAL on our B7700. For any other B7700 users who need a compiler, you can get one from UCSD, I suppose. It's okay, but we have discovered several bugs. There is also a fix that needs to be installed to allow the generated code to run properly on a B7700 (as opposed to B6700). I will send that fix on to UCSD. I don't know if they'll put it in. If anyone wants to, they can get one directly from me (at the above address)."

Also see Tom Shields in the Here and There section.

CDC Cyber 18 and 2550 (Santa Ana)

Jim Fontana reports that the Implementor/Maintainer of this implementation is Gordon Wood, CDC, La Jolla, California, and that the Distributor is Control Data Corporation, Professional Services Division, Sunnyvale, California.

CDC Cyber 18 (Berlin)

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Implementors: Lutz Christoph (Kernel) and Thomas Wagner (Interpreter); Technische Universitaet Berlin; DV-Grundausbildung; Institut fuer angewandte Informatik; VSH 5; Otto-Suhr-Allee 18/20; D-1000 Berlin - 10; Germany. Phone: secretariate 30-314-4893 from 8 to 13 MEZ except Tuesday.

2. MACHINE. CDC Cyber 18, Models 05, 10, and 20. Model 17 is not supported due to use of the Enhanced Instruction Set, and Model 30 because it is a dual processor machine. Floating point firmware and software will both be supported.

3. SYSTEM CONFIGURATION. In interpretive form will need less than 48KW for self compilation. Requires a disk (cartridge or SMD).

4. DISTRIBUTION. Not yet. Later will be distributed by CDC. (Agreement with the German branch.)

5. DOCUMENTATION. A supplement to the Brinch Hansen articles is planned.

6. MAINTENANCE POLICY. Not determined yet.

7. STANDARD. Brinch Hansen Pascal. Some extensions are to be discussed.

8. MEASUREMENTS. Will be provided when system is operative.

9. RELIABILITY. ?

10. DEVELOPMENT METHOD. Kernel/Interpreter rewritten and cross-assembled on CDC 6500. Compiler: Development not started.

11. LIBRARY SUPPORT. As with all Concurrent Pascal implementations. (poor)

CDC 3200 (Minneapolis)

Unfortunately, the rumor printed in Pascal News #9-10 regarding John Urbanski (West Bank Computer Center, 90 Blegen Hall, University of Minnesota, 269 19th Ave. South, Minneapolis, MN 55455) and the CDC 3200 implementation is no longer viable. John's 3200 is being replaced by a different machine.

CDC 6000, Cyber 70, Cyber 170 (Zurich)

George Richmond has announced: (1) that Release 2 of Pascal-6000 will be distributed for \$60, and (2) that buyers may no longer send their own tapes to receive Pascal. Also, John Strait and Andy Mickel sadly announce a six month delay in the long awaited Release 3.

CDC 6000, Cyber 70, Cyber 170 (Bethlehem)

PASCAL-I Interactive, Conversational PASCAL-S

PASCAL-I is a version of the Wirth PASCAL-S (PASCAL subset) system designed to interact with a terminal user. The system compiles and interactively interprets PASCAL-S programs. It automatically formats source text on compilation and allows the user to edit his program.

Text editing functions include GET and SAVE a file. All other editing operations follow PASCAL scope rules (i.e. the command LIST defaults to listing only the procedure being edited). Statements can be inserted, erased, replaced, moved, and printed. Strings can be searched for and changed. The REPEAT command reapplies the last edit command. There are no line numbers; the editing scope is always very local, and none seem needed or desired. The edit pointer can be moved from procedure to procedure, up and down within a procedure, and to the top and bottom of a procedure. Entire procedures can be moved or deleted. Their levels can be easily changed. A tree structured listing of procedure relationships is produced by the STRUCTURE command.

The HELP command gives detailed instructions and examples for each command. The MESSAGE command gives a full descriptive statement of the cause of a syntax error and sometimes includes recommendations for possible fixes. "LIST,E,M" will list only the erroneous lines of a procedure with the full descriptive messages for each error.

The compiler automatically formats the user program on input and re-compilation. Statements are placed on single lines and properly indented. Data structures are neatly laid out. Comments are well handled, and inter symbol spacing is rationally done.

The internal text is efficiently stored as linked, variable length, blank suppressed lines. After program changes, the compiler recompiles only the minimal set of affected procedures. The internal code is a dynamically managed, linked group of variable length code blocks. This internal code resembles PASCAL-S code, is position independent, highly compressed, and exceptionally efficient for interpretive execution; the linked data structures allow the system to perform efficient automatic dynamic garbage collection. Code and text for statements are mutually linked so that the run time system displays user program statuses with actual text line references.

A user may EXECUTE or CONTINUE his program. Single statement stepping is available. Statements or procedures may be traced. Fifteen breakpoints may be set, unset, or ignored. Limits on output lines, statements or PASCAL-S instructions to be executed can be set to halt execution. The system also recovers from user aborts and system timeouts. On time out, the system saves the user program to a file. A user abort behaves as an instantaneous breakpoint invocation. The system stops interpreting the user's program at a statement boundary and allows him to execute commands and later to restart his program.

A user may request a PMD (post mortem dump) of his variables. He gets a complete statement of his I/O buffers, simple variable names and values, and recent execution history. The system accepts PASCAL code for immediate execution. Code for immediate execution is delimited by dollar signs (\$) and can be anything from a statement to an entire procedure body (i.e. declarations and code but no procedure or function declarations). The user has complete access at the source level to his current dynamic program environment. One block of immediate code can be saved with each procedure and can be reinvoked by typing \$\$.

User program I/O is fully integrated into the system. The user can signal EOF on his program's input file; PASCAL-I passes this status to his program and accepts more input. Recovery of any user I/O error is automatically performed. The system gives the user three tries to pass valid data to his program. Detailed error messages with an annotated display of the offending input are presented. Anyone who thinks PASCAL doesn't have adequate facilities for interactive I/O should see PASCAL-I in action.

This system has the interactiveness of APL, BASIC, and the PL/I Checkout Compiler and does it for PASCAL.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER.
 - Richard J. Cichelli, Department of Mathematics
Bldg. # 14, Christmas-Saucon Hall
Lehigh University, Bethlehem, PA
(and ANPA/RI, P.O. Box 598, Easton, PA 18042 215-253-6155)
 - J Curtis Loughin, Lehigh University and ANPA
 - John P. McGrath, Lehigh University
2. MACHINE. Written entirely in PASCAL using some features of PASCAL 6000 (segmented files for terminal I/O to flush buffers and read past EOF on terminal input). Operational under SCOPE 3.4 using the CDC segmented loader.
3. SYSTEM CONFIGURATION. SCOPE 3.4 with INTERCOM.
4. DISTRIBUTION. Magnetic tape. SCOPE format, 7 track, 800 bpi. \$100. Other media by special arrangement.
5. DOCUMENTATION.
 - System level: Very readable code (guaranteed)
 - User level : System explains itself in response to the HELP command (full details - oriented towards novice programmers)
6. MAINTENANCE. Accepting bug reports. (Concise, fully documented, unreported bugs from Lehigh users are rewarded at \$1 each.)
7. STANDARD. Supports PASCAL-S. Differences from standard PASCAL - files - only INPUT and OUTPUT, no sets, user defined scalar types, pointer variables, case variants, labels, goto's, or with statements. Any PASCAL-S/PASCAL-I program is a valid PASCAL program.

We now have a fairly complete checklist for you, which I enclose, and some details of timings, both execution and compilation, which Bob Berry put together. I think PUG readers might find them interesting.

Thanks,

A. Foster.

Arthur Foster.

8. MEASUREMENTS. Interpreter and overlaid. Beats CDC BASIC hands down in cpu efficiency (size and time), features, etc. It is the fastest compiler on campus (because of partial compilation it outperforms PASCAL-S), and it does very well at run time. The compiler forms the largest overlay segment and runs at 358K. The editor segment runs in about 248K. PASCAL-I will compile and interpret PASCAL-S programs of up to about 500 lines as the system is currently configured.

9. RELIABILITY. Runs just great.

10. DEVELOPMENT METHOD. Started with PASCAL-S and Wirth-Jensen I/O routines. Built suitable data structures for storage of compressed program source and interpreter code. Modified PCSYSTEM to fully recover from user aborts and system timeouts. Also added file access primitives and moved stack and heap to low core to enable the segmented loader to vary field length. Development took place during April, 1977 to October, 1977 and was a very part-time effort. The system is about 7000 lines of tightly formatted PASCAL.

Implementor responsibilities:

- Curt Loughin
 - Editor
 - Formatter
 - PASCAL-S compiler rewrite
 - PASCAL-S interpreter rewrite
 - Immediate code routines
- John McGrath
 - I/O routines rewrite
 - HELP command
 - PCSYSTEM mods
- Richard Cichelli (project leader)
 - Post mortem dump
 - and other run-time control and status routines

CDC 7600 (Manchester)

Peter Hayes (UMRCC; Oxford Road; Manchester M13 9PL; England; 061-2738252) reported on 22 November 1977 several corrections to the Checklist printed in Pascal News #9-10:

- Price: 30 pounds sterling (instead of 50).
- Core requirements (octal): 42402 SCM, or 36045 if segment loaded (using a simple segment structure. Self-compiles in less than 60000.
- First released in July 1976. Now used at four sites, running under Scope 2.1.4.

Data General Nova (Lancaster)

University of Lancaster

Department of Computer Studies
Bailrigg, Lancaster
Telephone Lancaster 65201 (STD 0524)

Head of Department: J. A. Llewellyn B.Sc., M.Phil., F.B.C.S., F.I.M.A. 27th October 1977.

Dear Andy,

Here at Lancaster we are just about ready to distribute the second release of Nova PASCAL Compiler. We are aiming to do this on or about 25th November 1977. We have already sent details to Dr. Macnuskowski and Jim Hebert. They should also have copies of the new compiler shortly.

1. Distributors/Implementors : R.E.BERRY and A.FOSTER
Computer Studies Department
University of Lancaster
Bailrigg
Lancaster
2. Data General Nova
3. Developed with RDOS 4.02, we also use it under RDOS 5.00, and one user tells us he uses it under RDOS 6.10 without any trouble. We assume at least 32K words of core, disc backing store. The present system assumes no hardware Multiply/Divide or floating point.
4. Distribution : from Lancaster 2.5 Mbyte cartridge disk
Data General Cassette
in U.S.A. two people who have Lancaster Pascal have offered to act as distributors thus increasing the range of distribution media.
Dr.H.S. Magnuski Jim Hebert,
Gamma Technology 51 Thomas Road
800 Welch Road Swampscott
Palo Alto Mass. 01907.
California 94304. (9 track 800 BPI mag tape)
(9 track 800 BPI mag tape)
5. User manual provided, Pascal User Manual and Report assumed.
6. Maintenance policy : No formal commitment to provide support can be given, however, bug reports welcome. To date all known bugs have been fixed and this policy will continue for as long as is practicable.
7. Pascal P4 subset accepted.
8. Measurements : P-code is generated, assembled and then interpreted
Compiler core requirements (n.b. compiler uses overlays)

	<u>Release 1</u>	<u>Release 2</u>	
Compiler NMAX (decimal)	14,055	15,505	(words)
Additional fixed table space	1,092	1,197	(words)

The workspace remaining depends upon size of the RDOS system used. The size of program which can be compiled depends upon the number of user defined symbols (dynamic area used) and the depth of nesting of procedures/statements. Thus it is difficult to make any general statement about the size of program which can be compiled, however, we observe that the assembler for the system is of some 1,100 lines of Pascal source generating 7,400 Pcode instructions and we can compile this on our 32K system. We cannot compile the compiler but would expect to do so with more than 32K of core.

Timing : See attached sheet.

- 9. Reliability : Release 1 was made available to our own department users in January 1977 and at the time of writing has been distributed to eight known sites. No significant bugs have been reported from external users. Release 2 is at present available to our own users and will be available to others by the time this appears in PUG newsletter.
- 10. Development : Originally cross-compiled from a CDC 7600. The P-code assembler was written from scratch in Pascal; the P-code interpreter was implemented in Nova Assembly language.
- 11. Libraries : no library support in Release 1. Under Release 2 user procedures may be separately compiled enabling the user to set up his own libraries. It is not possible to link into any other libraries.

Timing information for Nova Pascal Release 2

We have not as yet compiled the compiler with our system so cannot give figures for that. Instead to provide the basis for our statement that the performance of our Pascal "compares favourably" with DG Algol a list of times obtained by running some well known small, and often uninteresting programs are given. The timings are taken from a Nova 2/10 running under RDCS 4.02 with 32K of core and no hardware multiply divide and no floating point unit. They were (rather crudely) obtained by using the GTOD command to prefix and postfix the CLI command necessary to load the appropriate program. "Compile" should be taken to mean the production of a save file (.SV) from the appropriate source program.

Program #	Algol		Pascal	
	<u>Compile</u>	<u>Run</u>	<u>Compile</u>	<u>Run</u>
	m:s	m:s	m:s	m:s
1.	0:55	0:06	1:21	0:07
2.	1:15	1:54	1:39	2:35
3.	1:16	14:32	1:40	11:59
4.	1:10	2:06	1:38	5:56
5.	1:09	2:52	1:37	1:55
6.	1:06	3:18	1:35	1:11
7.	1:08	1:28	1:36	1:03
8.	1:36	1:56	1:57	3:13
9.	"	4:46	"	4:30

More about the test programs

- 1. A program consisting simply of BEGIN END
- 2. Matrix multiply of two 50*50 matrices of integers.No I/O
- 3. Matrix multiply of 50*50 matrices of reals. No I/O
- 4. Sort an array of 1,000 integers into descending order of magnitude from ascending order of magnitude. No I/O
- 5. Ackomans function (3,6). No I/O
- 6. Write 10,001 integers into a file.
- 7. Read 10,001 integers from a file.
- 8. Generate 5,000 random integers (printing only the last).
- 9. Generate 5,000 random integers and write to a file.

Timings such as these offer much scope for debate. It is safer to let others draw what conclusions they will from these figures (and any other figures that may be produced). I simply wish to observe that interpretive Pascal "compares favourably" with the code produced by DG Algol. In the programs used above the Algol and the Pascal looks very much the same. No attempt is made to exploit one feature of a particular language or implementation, and no tuning has been done. If anyone has other examples to contribute to such timing comparisons I would be pleased to hear about them.

R.E.Berry.

Note - A.Foster believes that larger programs both compile and execute more efficiently in PASCAL than Algol, due to the nature of the language. He has experience writing large programs in Nova PASCAL and Nova ALGOL over the last year and believes that the real advantages of PASCAL are that Nova PASCAL programs are considerably easier to develop than Nova Algol programs, in his view.

GAMMA TECHNOLOGY

October 6, 1977

Dear Andy:

Thanks for the pre-release of the DG notes. Here's some new information for you.

I have made an agreement with R.E. Berry of the University of Lancaster to be a secondary distribution point for his NOVA PASCAL. We will supply the program on 9-track 800-bpi magnetic tape in RDOS dump format, and the manuals. The distribution charge is \$70 for the binaries of Release 1. We will collect trouble reports and ship them to England, but all corrections will have to come from U.L. Also, we cannot supply the program on any other media, on disks, on floppies, on cassettes, in bottles or cans. California residents must add the appropriate sales tax.

During September I had a chance to run a comparison of the Lancaster PASCAL against the PASCAL written by Ted Park now with Medical Data Consultants (formerly Loma Linda University). First, I must compliment both authors on the ease of use of both of these compilers. For both systems I had compiled and run a program in less than an hour after loading the tape, and there were no mysterious system crashes or wipeouts as one would expect with the first release of a new software package. The system used (courtesy of ROLM Corp.) was a top-of-the-line ECLIPSE C330 with DG's ZEBRA disk. The Lancaster PASCAL was also run on my 32KW NOVA 2/10 with a Diablo 30 disk drive. Five short PASCAL programs were compiled (a total of 5319 characters), and three of the programs were executed. These are the results:

	Lancaster NOVA 2/10	Lancaster C330	Loma Linda C330
Compile time (seconds)	515	242	582
Run time (seconds)	86	47	299

After these tests I called Ted to ask him what his program was doing with all that CPU time, and he described to me a long list of problem areas where performance could be improved (changes in his macro facility, use of variable size data structures, more efficient use of the RDOS I/O structure, revision of his paging algorithm, less runtime error checking in the compiler's P-code interpreter, etc.). He has implemented a virtual memory scheme and makes heavy use of the ECLIPSE instruction set, so with a little tuning up of his software I predict he'll be able to run larger programs faster than the Lancaster group.

The Lancaster PASCAL has a few quirks of its own, primarily related to problems of the compiler and run-time system knowing about their environment: a subdirectory is difficult to use, long filenames cause errors, and it doesn't work at all in the foreground partition.

Nevertheless, my conclusion is that we have two good implementations now available on DG equipment.

Yours sincerely,

Hank

H.S. Magnuski, President
Gamma Technology, Inc.

800 Welch Road ■ Palo Alto ■ California 94304 ■ 415-326-1661 ■ TWX: 910-373-1296

You have received a letter from Hank Magnuski comparing my system with the Lancaster system. He pointed out several problem areas (speed-wise, we still have found no bugs!) I mentioned to him. Since his letter to you I have cleaned up all these problems and am putting together a 'final' version. I am obtaining a copy of the Lancaster compiler to do my own internal benchmarks; I want to make detailed comparisons of the two systems. The Lancaster version will always compile a little faster than mine and maybe run character-oriented or integer-oriented programs faster. I feel the strong points of my compiler are:

1. Total integration into the RDOS environment
2. The exclusive use of double precision (32-bit) integers
3. The exclusive and fast use of double precision (64-bit) reals
4. Ease of modification and extension.

I will share my benchmark results with you when I complete them.

I am still happy to distribute my preliminary version free. I am negotiating with Hank to market the new version for a cost. The exact price and distribution methodology have not yet been established.

By the way -- It is no secret that PASCAL-P4 is a subset but has anybody compiled a complete list of its differences? I have begun one and will enclose it for your information. I'm sure it is not complete and would like someone to add to it and perhaps publish it in PUGN.

Sincerely,



Ted C. Park
Director, Systems Development

TCP:map

cc: H.S. Magnuski

1894 Commercenter West, Suite 302, San Bernardino, CA 92408

Data General Eclipse (San Bernardino) (previously Loma Linda)

MEDICAL DATA CONSULTANTS



(714) 825-2683

DEC PDP-8 (Minneapolis)

October 19, 1977

Dear Andy,

I received your pre-release of the D.G. notes for #9 and #10 -- thanks. I have some updated information for you concerning my ECLIPSE PASCAL system:

The SSRFC Pascal Group (SSRFC; 25 Blegen Hall; University of Minnesota; 269 19th Ave. South; Minneapolis, MN 55455 (612/373-5599)) announced on 31 December 1977 that: "The project is undergoing extensive redesign and development. We apologize to all those who have received no response to inquiries regarding distribution. We are not distributing any version presently. A mailing list is being maintained so that interested parties can be informed of changes in status."

Good News: DECUS Pascal SIG was resurrected by John Barr (Pascal SIG; c/o DECUS; 146 Main Street, PK-3/E55; Maynard, MA 01754; -OR- Hughes Aircraft Co.; Box 92919; Los Angeles, CA 90009; Attn: John R. Barr 377/C209) who edits the bimonthly DECUS Pascal SIG Newsletter. So far (31 Dec. 1977) we have received two issues: Vol. 1, numbers 1 and 2, dated Sept. 1977 and Dec. 1977. Number 3 is on the way. To quote "From the Editor" in issue number 1: "Welcome to the Pascal SIG. The Pascal SIG has been in existence in name only since Atlanta (Spring 1976 DECUS). It was started again in Boston (Spring 1977 DECUS) with a commitment from DEC (Larry Portner) to aid the SIG in the development of a Pascal compiler for the PDP-11 computers and operating systems." And from the same issue:

"The main goal of the Pascal SIG is to provide DECUS members with common Pascal compilers for all DEC equipment. One of the highest priority prerequisites of the effort is that such compilers will be available to users for only the cost of reproduction through the DECUS library. Support of the compilers will rest with the Pascal Implementation Working Group who will use this newsletter to report bugs and fixes to these bugs.

"A secondary goal of the Pascal SIG is to report the existence of commercially available Pascal compilers and/or compilers developed outside of the SIG which may be of interest to users. In accordance with DECUS policies we cannot report the cost of such compilers but can only indicate who the users can contact for information on them. We will publish at most one announcement per year from any company which has a Pascal compiler available for DEC computers."

John Barr reports that the following persons are Officers of DECUS Pascal SIG:

- Program Librarian: Tom Tyson (PUG member).
- PUG/SIG Liaison: John Iobst (PUG member).
- Standards Committee: Ken Bowles (PUG member) and John Iobst.
- DoD-1 Language Committee: Fredreck Bartlett and Don Chaney.
- Concurrent Pascal Committee: Roger Vossler.

Anyone may join DECUS by contacting one of the following offices:

AUSTRALIA and NEW ZEALAND:
DECUS
P.O. Box 491
Crows Nest, N.S.W. 2065
Australia

EUROPE and MIDDLE EAST:
DECUS
Case Postale 340
CH-1211 Geneva 26
Switzerland

CANADA:
DECUS
P.O. Box 11500
Ottawa, Ontario K2H 8K8
Canada

ALL OTHERS:
DECUS
146 Main Street
Maynard, Massachusetts 01754
U.S.A.

DEC PDP-11 (Stockholm)

Seved Torstendahl's implementation has become quite widespread -- it was reported (in PUGN Checklist form!!) in the first issue of DECUS Pascal SIG Newsletter, and we have heard from several PUG members who have used it: Jon Gross (Minneapolis), Steve Schwarm (Wilmington), and Rich Cichelli (Bethlehem).

See also Ken Bowles' article "Status of UCSD Pascal Project" in this issue for information about Z80's, 8080's, and CP/M.

* OFFICIAL NOTICE OF RELEASE OF UCSD PASCAL *

January 1978

Dear Mini-Micro User;

This PASCAL system is our first system intended for general purpose use away from UCSD. We are still making major modifications to the software, and expect that you will find errors. If you do find an error in our system, please contact us before attempting to fix the problem yourself. We may have already found the error, and solved it. There are many expansions yet to be made to our PASCAL system, and hope that you will let us know what is needed. Recompilation of object code files will be necessary on major updates of the system. Please remember that our project is mainly staffed by students. We intend to support users who have paid the distribution fee within our resources, but may not respond as quickly as a commercial vendor might, due to conflicting university schedules.

We are offering two kinds of subscriptions:

Complete subscription: Floppy disks with all sources and code. Compiled listings of all sources. User and system maintenance documentation as complete as it exists. Updates at least three times during the next 12 months.
\$200. (paid in advance)

Code subscription: Floppy disk with system code. Users manual but no detailed system documentation. No continued support for later subscriptions. Only minimal assistance in response to telephone inquiries.
\$50. (paid in advance)

The complete subscription is a years bond with our project. We will do our best to answer your questions, and keep you up to date on new subscriptions. The code subscription is for the user who is not concerned with our development, and just wants a running PASCAL system. A user of the code subscription may upgrade to the complete subscription upon payment of \$175.

Please make your check payable to: Regents, University of California

Please return the completed order form, and your check to:

PASCAL Group
Institute for Information Systems
UCSD Mailcode C-021
La Jolla, CA 92093

In a conversation with Andy Mickel on 17 December 1977, Ken Bowles mentioned that DEC is coming out with Writable Control Store (WCS) for the LSI-11. Ken expects that use of this feature will speed up the UCSD system by a factor of five -- allowing it to compile 3000 lines per minute!

* REQUEST FOR COPY OF UCSD PASCAL *

Processor configuration system is to run on:

- LSI-11
- PDP-11 model _____
- TERAk B510
- TERAk B510A
- other (please specify) _____

- The above mentioned processor has PDP11/40 type (e.g. LSI-11 EIS/FIS chip) floating point hardware

Memory space available on above processor:

_____ words.

Mass storage to be used with above system:

- RX11 (DEC floppy disk drives)
(or hardware equivalent / software transparent)
- TERAk
- AED 3100
- RK05 (we may not be able to support all formats)
- Other (please specify, we may not support it, someone else might) _____

Other peripherals to be supported:

- The above system has a line printer with an LP11 (or equivalent) interface.

Terminal(s) to be attached as console device to above system:

- Decscope series (please specify model) _____
- Datamedia product (please specify) _____
- Other (please specify) _____

Type of release wanted:

- Complete release (\$200 paid in advance)
- Code release (\$50 paid in advance)
- No release, but keep us up to date on new developments.

Please complete this form, and return with your check to:

PASCAL Group
Institute for Information Systems
UCSD Mailcode C-021
La Jolla, CA 92093

SHIP TO: {i.e. where do you want your PASCAL sent}

DEC PDP-11 (Amsterdam)

-
1. DISTRIBUTOR/IMPLEMENTOR/MAINTAINER.
DISTRIBUTOR: Sources, binaries and documentation are part of the third UNIX software distribution.
IMPLEMENTOR: Johan Stevenson, Vrije Universiteit.
MAINTAINER: Andrew S.Tanenbaum, Wiskundig Seminarium, Vrije Universiteit, De Boelelaan 1081, Amsterdam, The Netherlands, tel: 020-5482410.
 2. MACHINE. DEC PDP 11, all models on which the UNIX operating system version 6 will run.
 3. SYSTEM CONFIGURATION. see 2.
 4. DISTRIBUTION. The UNIX software distribution center takes care of distribution.
 5. DOCUMENTATION. Short manuals for the compiler and interpreter in UNIX MAN format. A 12 page description giving details about the implementation.
 6. MAINTAINANCE POLICY. Bug reports are welcome. There will be no improved release of the current system. However, we are working on a totally new one. Main differences with the old one are:
 - a new hypothetical stack computer named EM1 (see Tanenbaum, A.S. "Implications of structured programming for machine architecture" CACM Dec. '77). This intermediate machine allows very compact code (only 15000 8-bit bytes for the compiler itself) and fast interpretation. Emulating EM1 on a microprogrammable computer must be easy. Moreover, this EM1 machine allows compilation of other high-level languages as well.
 - a new interpreter with all kinds of runtime checks and debugging aids.
 - expansion of EM1 code into PDP-11 instructions.
 - less restrictions on the language Pascal.
 7. STANDARD. Main differences with full Pascal are:
 - no goto's out of procedures and functions.
 - procedures and functions can not be passed as parameters.
 - extern procedures and functions not implemented.
 - mark and release instead of dispose.
 - only non-local textfiles (up to 15).
 - an explicit get (or readln) is needed to initialize the file window even for input.
 8. MEASUREMENTS.
Compilation speed: 40000 chars per minute on a PDP-11/45 with cache.
Compilation space: 48k bytes to compile the compiler. Very big programs can be compiled.

execution speed: You lose a factor 8 by interpretation. However I/O is relatively fast. Compared to not interpreted Pascal on a big machine (Cyber 73) it is ten times slower.
 execution space: The size of the complete interpreter is 5300 bytes. The binary code for the compiler is 23000 bytes.

- 9. RELIABILITY. The compiler and interpreter are good. However the runtime checking of the interpreter is poor.
- 10. DEVELOPMENT METHOD. The compiler is based on the PASCAL-P2 compiler. A Cyber-73 was used for bootstrapping. The time needed by an unexperienced implementor was about 6 months.
- 11. LIBRARY SUPPORT. No library support at all.

DEC PDP-11 (GTE SYLVANIA)

1. IMPLEMENTORS:

<p>Larry Drews (Address Unknown)</p>	<p>David Miller (PUG member) GTE SYLVANIA 11203A Avalanche Way Columbia, Maryland 21044 (301) 992-5665</p>
--	--

<p>Sharlene Wong (almost PUG member) GTE SYLVANIA P.O. Box 205 Mountain View, California 94042 (415) 966-3373</p>	<p>David Shaw (PUG member) STRUCTURED SYSTEMS CORP. 427 Embarcadero Road Palo Alto, California 94301 (415) 321-8111</p>
---	---

2. MACHINE:

Digital Equipment Corporation
PDP 11/45 and up I suppose

3. SYSTEM:

DOS/BATCH-11 V9.20C

4. DISTRIBUTION:

Corporate policy has not been decided at this time. If there is enough community pressure it could be arranged. It must be emphasized that this compiler was developed to support a single application.

5. DOCUMENTATION:

There is lots of good comments in the source compiler and supplementary information on the internals of the compiler system have also been written. There isn't much good information on the usage or details on the extensions however. We used the first edition of JENSEN and WIRTH user manual.

6. MAINTENANCE:

The compiler probably will be maintained for several years, to support the application. It is not expected to change much over the years -- it is quite satisfactory as it stands. Any problems which may turn up will probably be worked around rather than fixed. For instance, LABEL doesn't appear to work altogether correctly, but we don't use it anyhow.

7&8 IMPLEMENTATION:

As described in our paper, we probably have the most non-standard PASCAL in the world. We are therefore its most fearsome friend. (PN8 - p33). The following shopping list is a concatenation of the University of Illinois PASCAL (PN5 - p53, PN8 - p51-52), and of our own modifications.

EXTENSIONS:

- Overlay: the compiler is itself an overlay with nine segments. The application doesn't use this feature.
- Identifiers may contain an underscore as a break character.
- It recognizes octal constants.
- Multiple assignments and imbedded statements are supported.
- McCarthy boolean expression evaluation has been implemented.
- WRITE formats may specify; octal, hexadecimal, binary, unsigned decimal; field width; with or without leading zeros.
- Pretty print/line editor pass may be included in the compilation. The editor line numbers are carried through the compiler.
- A cross reference pass may be included in the compilation.
- Compiler directives include:
 - list suppression
 - page eject
 - statement number code suppression (used for post mortum trace)

3. Operating systems: SOLO, Experimental Development System, and user written systems.

Minimal hardware configuration: 48 KW
 memory, 1 - RK05 disk drive, 1 - 9-track
 tape drive, FP11 floating point option.
 Future versions will run on smaller PDP-11 machines without floating point.

4. Method of distribution: no established policy.
5. Documentation available: (a) machine retrievable user manuals are available for all utility programs; (b) several books and papers have been published by Per Brinch Hansen and Al Hartmann on the operating systems and compilers.
6. Maintenance: no established policy.
7. Two compilers are available: CPASCAL (Concurrent) and SPASCAL (Sequential). SPASCAL implements most of standard Pascal. The main exceptions are that procedures may not be nested, and only a few of the standard predefined Pascal functions are available. CPASCAL has further restrictions on scope rules and does not implement the pointer type. Extensions to CPASCAL include processes, classes, and monitors, and compile time checking of access rights.
8. Both compilers generate virtual code which is interpreted. The interpreter and operating system kernel are written in MACRO-11, the PDP-11 assembly language. Together they comprise less than 7000 statements, and are the only assembly language components of the operating system. Work is currently being done to reduce the size of the kernel. The SOLO operating system is written in Concurrent Pascal, consisting of about 1200 statements. All other utilities, including compilers and the file system, are written in Sequential Pascal. The compilers consist of seven passes each, totaling about 9200 statements per compiler. Compilation speed is about 240 char/sec on a PDP-11/45.
9. Reliability of compilers: very good; only one bug has been found in nearly a year of heavy use.
 Reliability of interpreter: excellent; no bugs have been found to date.
10. Method of development: developed by Al Hartmann and Per Brinch Hansen.
11. Libraries of subprograms are not available. Procedures and functions to be referenced by a program must either be included in the source file or accessed through a procedure entry in the operating system. Facilities for using procedures written in other languages are not currently available. Separate compilation is available in a limited sense in two ways: (a) commonly used procedures may be compiled and included in the operating system; (b) a sequential program can call another sequential program.

DEC PDP-11 (PAR)

The implementation started by Michael N. Condict (PAR Corporation; On the Mall; Rome, NY 13440; 315/336-8400) and reported in Pascal News #9-10 has been discontinued.

DEC PDP-11 (Vienna)

Distributor and Maintainer:

Österreichische Studiengesellschaft für Atomenergie Ges.m.b.H., Lenau-gasse 10, 1082 Wien, Physikinstitut

Implementor:

A portable Sequential and Concurrent Pascal compiler has been designed and implemented by Prof. P.B. Hansen and A.C. Hartmann, California Institute of Technology. The Compilers have been modified and a new runtime system has been implemented by D.I. Konrad Mayer, Österreichische Studiengesellschaft für Atomenergie.

Machine:

Digital Equipment's PDP11, all types

System Configuration:

Operating System

a) RSX11-M (version 2 or later)
 or compatible systems (RSX11-D, IAS)

b) RT11 (version 2C or later)

Minimum memory requirements for self-compilation of the compiler.

a) RSX11M: 23 k words-partition

This amount can be reduced by using overlays.

b) RT11: 24 k words (including the operating system).

Distribution:

Distribution medium: DEC-tape or floppy disk or magnetic tape (9 track, 800 bpi).

Format

Files 11 or RT11 or DOS for DEC-tape and floppy disk
 A "PRESERVE"-copy of a RK05 disk for magnetic tape

The package contains

- All sources and codes of the Sequential and Concurrent Pascal compiler.
- All sources and object code of the runtime system and the I/O routines.
- All indirect command files (RSX11 only) for assembling and building the runtime system.

The runtime system can execute only Sequential Pascal Programs! A system for Concurrent Pascal programs will be available later.

- A disassembler for analyzing the generated code of Sequential and Concurrent Pascal programs.
- Documentation

Distribution costs:

5.000.-- Austrian Schilling
 including distribution media.

Documentation:

The compiler and the language is described in

Hartmann, A.C., A Concurrent Pascal Compiler for Minicomputers.
Lecturę Notes in Computer Science 50, Springer Verlag, New York, NY, 1977

and in the Concurrent and Sequential Pascal Reports of California Institute of Technology.

The distribution package includes a supplement to the reports and a description of the runtime system.

Maintenance and Warranty:

Although every attempt has been made to achieve accuracy and freedom from errors, Österreichische Studiengesellschaft für Atomenergie, makes no warranty of any kind and does not guarantee correctness or maintenance of the system. But reported errors will be corrected, if possible. Please report errors to D.I. Konrad Mayer (address see above).

Language Standard:

Note, that Sequential Pascal is different to standard Pascal. The Sequential and Concurrent Pascal compilers have been extended slightly (a different lexical analyzer) to meet RSX11 and RT11 standards. I/O routines for handling sequential and random access files are provided.

Measurments:

Compilation speed: 150 - 200 characters per second (PDP11/45)
Execution time: 1 to 3 times of equivalent Fortran programs

Reliability:

The reliability of the system is really excellent. We have the system in use since one year. Up to now we have had only one minor problem with the system.

The portable compilers of P.B. Hansen/A.C. Hartmann are in use on some sites on different machines. The first release was January 1975.
(See Pascal News, Vol. 6-10)

Development method:

We started with the distribution kit of the Solo system. Then we wrote a kernel-interpreter, which can execute Sequential Pascal programs without needing the Solo operating system. The interface to DEC's operating system RSX11 (RT11) is the virtual machine instruction CALLSY.

External routines:

External assembler routines can be called via the "program prefix". Every external routine must be defined in the program prefix and can be linked using the standard Fortran call conventions.

DEC PDP-11 (Los Angeles)

(* The following was obtained from the DECUS Pascal SIG Newsletter vol. 1, number 1. *)

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. John R. Barr; 377/C209 - Hughes Aircraft Company - Box 92919 - Los Angeles, CA 90009 (213/648-8295).
2. MACHINE. PDP-11/35 and higher. Requires EIS.

3. SYSTEM CONFIGURATION. RSX-11D, IAS, RSTS, and RSX-11M.

4. DISTRIBUTION. Send 600' mag tape. No cost.

5. DOCUMENTATION. (* no information given *)

6. MAINTENANCE POLICY. Bug reports accepted with corrections in the DECUS Pascal SIG Newsletter.

7. STANDARD. Implements most of Standard Pascal. Exceptions are no for loops, real arithmetic, or set operators "<=" or ">="; set operators are represented by "!" (union) and "&" (intersection); logical operators are "~" (negation), "!" (disjunction), and "&" (conjunction); functions and procedures not supported are: abs, arctan, cos, eoln, exp, ln, round, sin, sqr, sqrt, trunc, pack, page, readln, unpack, and writeln; predefined terminal input and output files are INP and OUT; reset and rewrite require a second parameter of type array [0..N] of char representing a file specifier terminated by a blank. Extensions are structured constants and the ability to mix the order of const, type, and var declarations.

8. MEASUREMENTS. The compiler is currently written in Block Structured Macros, and is in the process of being converted to Pascal. It is a two pass compiler with each pass requiring approximately 20K words of task space. Compile speed is approximately 450 characters per second. PASS2 of the compiler written in Pascal is approximately 2200 lines which compiles in less than four minutes on a PDP-11/45.

9. RELIABILITY. At present the BSM compiler will loop and possibly abort under several conditions. Syntax errors are reported using references to the error table in Jensen and Wirth. The Pascal versions are expected to be much better.

10. DEVELOPMENT METHOD. The major portion of the compiler was taken from Brian Lucas' DOS version of the compiler. The PASS2 Pascal code has been extensively debuged in about three months of part time effort.

11. LIBRARY SUPPORT. Several Pascal programs which aided in the development effort are available. At present there is no facility for separately compiling procedures or for linking to non-Pascal subroutines other than the run-time library.

DEC-10 (Hamburg-DECUS)

UNIVERSITÄT HAMBURG

INSTITUT FÜR
INFORMATIK

Prof. Dr. H.-H. Nagel

Institut für Informatik
2 Hamburg 12, Schillerstraße 66-72

Dear Mr. Mickel,

in February 1977 I made our PASCAL compiler for the DECSystem-10 available to Digital Equipment Corp. to whom I had ceded the distribution rights. Since the envisioned distribution through DECUS took some time I received numerous enquiries concerning our compiler system. I referred people interested in using PASCAL on the DECSystem-10 to DECUS. I would be interested in learning whether the distribution of our compiler version through DECUS is now running to the satisfaction of interested users. Experiences with the distribution of our current compiler version might influence the way that a further improved version might eventually be distributed.

Since I no longer know who intends to use our latest compiler version and who is actually using it I would appreciate if you could include this letter in the next PASCAL Newsletter.

With many thanks in advance I remain

yours sincerely,

A. H. Hagel

Hewlett Packard HP-21MX (Dallas)

David McQueen from Analyst International, Dallas, Texas, reports that he has a Pascal system for the HP-21MX implemented in Fortran which provides direct access files.

Honeywell 6000

(* Please also see the article in this issue by James Q. Arnold entitled "A Novel Approach to Compiler Design". *)

THE UNIVERSITY OF KANSAS / LAWRENCE, KANSAS 66045

Dear Andy,

29 October 1977

Department of Computer Science
18 Strong Hall
913 864-4482



Thanks for entering my membership in PUG. I have read my backissues of the PUG newsletter and find only passing reference to the implementation of Pascal on the Honeywell 66/60 under GCOS. Pascal is a "software product" offered by Honeywell to installations with Series 6000 and Series 60 Level 66 computers. The compiler originated at the University of Waterloo, Waterloo, Canada, and is distributed through Honeywell (shades of WATFIV). There are two aspects of the implementation which will be of interest to PUG members: what the implementation should do, and what it does do.

The Honeywell version of Pascal is an independent implementation (unrelated to CDC-Pascal or Pascal-P) and is written in "B," an implementation language and successor of BCPL. It is completely time sharing based (compiled programs can be run in either batch or TSS). The compiler has one restriction on Standard Pascal and several extensions.

The restriction is that the program statement is replaced by "procedure main;" and the dot terminator at the end of the program is eliminated. This has a curious side-effect. Constants, types, variables, and procedures (functions) may be defined outside of "main" and thus allows the definition of a global environment which may be replicated for externally (separately) compiled procedures, thus allowing the global variables to be accessed both from the program ("main") and from the externally defined procedures. Other than that the compiler accepts Standard Pascal programs.

Of the implementation dependent features, several are worth mentioning here. Identifiers need to be distinct in the first 80 characters. The base type of a set is limited to at most 9,437,184 elements (36 bits x 262,144 words).

The compiler distinguishes between lower and upper cases and will recognize keywords in both cases or in lower case only. This is based on terminal command and means that you can use the upper case version of a reserved word as an identifier -- not suggested, but it can be useful. Our typical use is uppercase CONST and TYPE identifiers and lowercase variables. The standard character set is Ascii.

There are several extensions in the Honeywell implementation. An alternate I/O package is provided (in addition to the standard get, put, read, write, etc.) to make interactive use easier (it is based on the RATFOR input/output package described in Software Tools, by Kernighan and Plauger). External procedures are allowed and calls may be made to Pascal procedures or to procedures written in other languages. The declaration of external procedures is implemented as it is in CDC Pascal with the exception that "extern <language>" can be used to declare an external procedure in another language. Currently, <language> may only be FORTRAN. Extensions are planned to allow GMAP, COBOL, Algol, PL/1, B, C, etc. An else clause is added to the case statement (and in the record variant section) to allow default selection. Files may be of any type, except file of file. Files may be accessed and associated with a file variable at run time via a procedure call. The standard files "input" and "output" are normally directed to the terminal, but can be redirected to files by terminal command when invoking a Pascal program. Subscript checking is implemented by bounds checking on all variables of subrange type.

Now that I have decried the way the compiler is supposed to work, allow me to describe some of our experiences using it. We at KU have had access to the compiler for approximately 10 months (since January 1977). We are currently using version 5 of the compiler and expect a sixth version by late December. As stated earlier in this letter, our Pascal is distributed through Honeywell. The actual implementor is the University of Waterloo. This leads to several problems; most of them having to do with the communication of bugs, suggestions, questions, etc. The lack of fast and accurate communications would not be quite so important, however, if the compiler worked and if the documentation were complete. Unfortunately, the compiler has been plagued with several very serious bugs. When using compound structures, the compiler sometimes computes incorrect addresses thus resulting in memory address faults or operation faults (where the data is unintentionally stored over the program code and subsequently "executed"). The compiler aborts when it encounters some simple syntax errors (such as using a reserved word as an identifier). Resetting an output file garbages the file. Most of the serious bugs, and hence most of our complaints, center on the compiler either aborting with no indication of what is wrong, or producing object code which behaves similarly. At present, we have found more than 30 serious bugs in the compiler; the current version retains 11 of them (the others being fixed). Fortunately, we now have direct contact with Waterloo and are getting much better response to our complaints. We hope that by next January, most of the serious bugs will be fixed. As it is, a couple of people working on large Pascal programs have almost (?) given up using the compiler (one has written a letter to PUG to vent his frustrations). It is unfortunate that more extensive testing was not done before releasing the compiler for distribution.

We are currently using the compiler in three classes: Data Structures, Compiler Construction, and Programming Structures. This totals more than 150 students each using the compiler several times a week. Two graduate students are writing an assembler and a simulator (emulator?) for the Interdata 85 minicomputer to aid in teaching an under-graduate course. We are also trying to implement the BOBS System Parser Generator (our copy from the University of Aarhus, Denmark). Unfortunately, these projects are at a standstill because the compiler either won't compile them (the compiler aborts) or compiles them into incorrect object code. But, we're all hoping for the best, so until January and a new version.....

I am enclosing documentation which defines the differences between Standard Pascal and Honeywell Pascal -- this is in the form of an appendix to the User Manual and Report. You may or may not want to include it in the newsletter. It's up to you. I am sorry not to have written this in the form of the implementation checklist, but I am not the implementor -- I have talked with the folks from Waterloo and they assure me they will send you a letter. Thanks for spreading the word.

Sincerely,



Greg Wetzel
Research Assistant

Honeywell H316

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Robert A. Stryk, Honeywell Corporate Computer Science Center, 10701 Lyndale Ave. So., Bloomington, MN 55424; 612/887-4356.
2. MACHINE. Honeywell H316.
3. SYSTEM CONFIGURATION. 32K, dual cartridge disks, line printer, 7 track magnetic tape.
4. DISTRIBUTION. 7 track tape with programs to bootstrap from BOS 210.
5. DOCUMENTATION. Informal comments on 316 kernel implementation.
6. MAINTAINANCE. No known errors, no work planned.
7. STANDARD. P. B. Hansen's SOLO system with modified standard and Concurrent Pascal.
8. MEASUREMENTS. SOLO system needs minimum of 40K to execute compilers.
9. RELIABILITY. No known errors.

10. DEVELOPMENT METHOD. The H316 kernel imitates the PDP/11 reversed-byte addressing which makes it compatible with the distribution tape but slow in execution. The development was done under BOS 210. The kernel is written in DAP700.

11. LIBRARY SUPPORT. Those provided by the SOLO system.

IBM 360, 370 (Australia)



AUSTRALIAN ATOMIC ENERGY COMMISSION

NUCLEAR SCIENCE AND TECHNOLOGY BRANCH

RESEARCH ESTABLISHMENT, NEW ILLAWARRA ROAD, LUCAS HEIGHTS

TELEGRAMS: ATOMRE, SYDNEY
TELEX: 24562
TELEPHONE: 531-0111
IN REPLY PLEASE QUOTE: GWC.mwb

ADDRESS ALL MAIL TO:
AAEC RESEARCH ESTABLISHMENT
PRIVATE MAIL BAG, SUTHERLAND 2232
N.S.W. AUSTRALIA

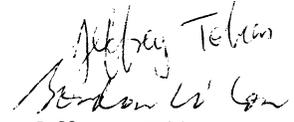
4 January, 1978.

Dear Andy,

Enclosed is up-to-date information on our implementation of Pascal 8000 for IBM 360/370 computers. Variations from previous information are:

1. Two versions are now supplied - the original compile-and-go version, with provision for saving compiled code, as well as a version which produces IBM-standard object modules and can link to externally compiled procedures written in Pascal or Fortran or Assembler language.
2. The price is now \$A100 for both versions of the system in source and object form, plus machine-readable documentation, all supplied on a 600 ft. magnetic tape. There is now no agreement to be signed.

Yours sincerely,



Jeffrey M. Tobias

Gordon W. Cox

Applied Mathematics & Computing Division

PASCAL 8000 FOR IBM 360/370

1. Implementors:

T. Hikita and K. Ishihata,
Dept. of Information Science,
University of Tokyo,
2-11-16 Yayoi
Bunkyo-ku TOKYO,
113 JAPAN

(HITAC - 8000 Version)

G.W. Cox and J.M. Tobias,
Systems Design Section,
AAEC Research Establishment,
Private Mail Bag,
SUTHERLAND, 2232,
N.S.W. AUSTRALIA

(IBM 360/370 Version)

Distributors/Maintainers:

G.W. Cox and J.M. Tobias
address as above

2. Machines:

IBM360 and IBM370 - compatible machines

3. System Configuration:

The compiler runs under any of the OS family of operating systems - i.e. MVT, MFT, VS1, VS2, SVS and MVS. A minimal program can be compiled in 128K; the compiler requires about 220K to compile itself.

4. Distribution:

Write to G.W. Cox and J.M. Tobias at AAEC to receive an order form. The cost is \$A100; there is no agreement to be signed. Two systems are supplied: a "compile-and-go" system which has its own compiled-code format, and a "linkage-editor" system which produces IBM-standard object modules. Both source and load modules for these systems are supplied - the compilers are written in Pascal and the runtime support in 360 Assembler.

An implementation guide, plus machine-readable implementation JCL, and machine-readable documentation are also supplied.

The system is distributed on a new 600 ft. magnetic tape at a density of 800 or 1600 bpi; the tape is supplied by the distributor.

5. Documentation

Machine-readable documentation is in the form of a report comprising a summary of extensions to Standard Pascal plus a complete specification of the language as implemented.

6. Maintenance Policy.

No guarantee on maintenance is given; however we are anxious to receive bug reports and suggestions, and will do our best to fix any problems which may occur.

7. Standard.

The full standard is supported with finiteness in a few areas:

- maximum static procedure nesting depth is 6.
- maximum set size is 64. (this precludes set of char.) It is hoped to increase this soon.
- maximum number of procedures in a program is 256
- maximum size of compiled code in any one procedure depends on its static level: the main program may be up to 24K, and this is reduced by 4K for each increment of static nesting level.

Significant extensions to the standard are in the following areas:

- Constant definitions for structured types. It is therefore possible to have arrays, records and sets as constants.
- A 'value' statement for variable initialisation
- A 'forall' statement of the form:
forall <control variable> in <expression> do <statement>
where <expression> is of type set.
- A 'loop' statement, specifying that a group of statements should be repeatedly executed until an 'event' is encountered. Control may then be transferred to a statement labelled by that event.
- The types of parameters of procedures or functions passed as parameters must be specified explicitly, and this enables the compiler to guarantee integrity.
- The 'type identifier', restriction in a procedure skeleton has been relaxed to allow 'type'.
- Functions 'pack' and 'unpack' are supported, as are packed structures in general.
- Exponentiation is fully supported, and is used via the double character symbol '***'.
- A 'type-change' function has been introduced that extends the role of 'chr' and 'ord'.
- Case-tag lists may now range over a number of constants, without explicitly having to list each constant.

The range is denoted by

<constant> .. <constant>

Thus,

4,6..10,15,30..45

is now a valid case tag list

A default exit is also supplied via

else: <statement>

i.e. else: is a valid case tag in every case statement. This path will be used if none of the other tags match.

Other interesting features of the system are:

- Procedure 'new' is fully supported, obtaining the minimum heap requirements as specified by variant tags. Procedures 'mark' and 'release' are also supported.

Procedure 'dispose' is not supported.

- Files may be external or local. Thus, structures such as 'array of files' are available. External files are named in the program statement, local files are not. Both external and local files may be declared in a procedure at any level.

- Text-files with RECFM of F[B] [S] [A], V[B] [S] [A] and U[A] are supported. Non-text files must have RECFM = F[B].

- All real arithmetic is in double precision (64 bit floating-point format).
- Control of input and output formatting is as described in the Jensen and Wirth report. The form is

X[:n] [:m], where n and m are integer expressions.

Further, elements of type packed array of char may be read on input.

- Execution errors terminate in a post-mortem dump, providing a complete execution history that includes procedure invocations, variable values, type of error, etc.
- the use of separately-compiled procedures in Pascal, FORTRAN or other languages is supported by the linkage-edit version. Thus one can build up a library of Pascal procedures or use a pre-existing library of FORTRAN routines.

8. Measurements.

- compilation speed about 2,500 chars/sec on an IBM 360/65
- compilation space : 128K for small programs
160K for medium programs
220K for the compiler
- execution speed : comparable with Fortran G.
- execution space : about 30K plus the size of the compiled code, stack and heap
Compiled code is fairly compact - the compiler itself occupies 88K.

9. Reliability.

The system was first distributed in its current form late in 1977. It is currently used at about 30 sites. Reliability reports have been generally good to excellent. A few minor problems which have been reported are currently being fixed.

10. Development Method

The compiler was developed from Nageli's trunk compiler and bootstrapped using Pascal-P by Hikita and Ishihata, who got it running on a HITAC-8000 computer (similar instruction set to IBM360). This version was further developed by Tobias and Cox for use under the OS family of operating systems on IBM360/370 computers. The compiler is written in Pascal 8000 (6000 lines) and runtime support is in 360 Assembler (3500 lines). Cox and Tobias spent about 10 person-months on the system. Most of this time was spent improving the OS support and adding enhancements to what was already a very workable system.

11. Library Support.

The linkage-edit version has the ability to perform separate compilation of procedures or functions. These can be stored in a library and selected by the linkage editor as necessary. It can also link to routines written in FORTRAN or other languages which use a FORTRAN calling sequence. To use an externally compiled routine, one must include a declaration for it. Such declarations consist of the procedure or function skeleton followed by the word 'pascal' or 'fortran'. The linkage-editor then automatically searches for that routine when it is linking the program. Global variables are accessible to externally compiled Pascal routines. Pascal procedures cannot be overlaid.

A symbolic dump of local variables and traceback of procedures called is provided on detection of execution errors.

IBM 360, 370 (Williamsburg)

1. Michael K. Donegan
Dept. of Mathematics & Computer Science
College of William and Mary
Williamsburg, Virginia 23185
(804) 253-4481
2. IBM 360/370
3. OS/VS
Requires 192-256K to compile the compiler, 150K is sufficient to compile "normal" programs.
4. Distribution is in the form of object, source, and documentation on 9-track tape and is currently free if you supply a tape (we can't). This is subject to change.

IBM 370 (London)

(* The editors wish to thank the persons in Berlin and London for taking the time to fill us in so well on their activities. We never expected such an outpouring of information when we wrote #9-10! Once again, thanks! -Andy, Jim, and Tim *)

TECHNISCHE UNIVERSITÄT BERLIN

Fachbereich Informatik (20)

DVG Technische Universität Berlin (FB 20)
LEHREINHEIT DV-GRUNDAUSBILDUNG
1 Berlin 10, Otto-Suhr-Allee 18/20

BERLIN

TU Berlin - FB 20 - VSH 5 - Otto-Suhr-Allee 18/20, D-1000 Berlin 10

PASCAL News - Editor

University Computer Center
227 Experimental Engineering
208 Southeast Union Street

University of Minnesota

Minneapolis, MN 55 455 USA

Tel.: (030) 314- 4893

Telex: 1 84 262 tubln -d-

November 30, 1977

Dear Andy:

A necessary correction of # 8 and 9/10 : At the Technical University of Berlin, we never embarked on an implementation based on P4 , but we do not intend to drop our efforts at a new implementation of PASCAL for IBM /370 under CMS (Thomas Habernoll). In addition, we are starting an implementation of Concurrent PASCAL for the CYBER 18. We have added check lists for both projects, reporting the current status.

We are using the P4-based implementation of Imperial College London for purposes of teaching and bootstrapping. As these people seem to be very modest (humble programmers ?), a word of praise for their work seems appropriate:

1. Greg P u g h implemented a VM /370 CMS PASCAL-compiler in September 76 based on P4. Its P-code is converted to standard OS object modules in a separate pass. A number of extensions are helpful for systems programmers; the interface to the operating system is very convenient to use.
2. At Technical University of Berlin, the London compiler was ready to use within a few hours; we found some minor bugs within six months of heavy usage, and these have been corrected in the latest release from London.

5. Documentation is distributed on the tape. We are currently writing an implementor's guide.
6. Maintenance is currently in the form of a new distribution and reported bugs will be fixed on a time-available basis. We are continually working on the limitations of the system and have plans for:
 - 1) Additional debugging facilities, cross-reference, post-mortem dump callable as a built in procedure, frequency profile.
 - 2) Better interactive capability for TSO users.
 - 3) Double-precision arithmetic.
 - 4) Dynamic arrays—as soon as some one can come up with a decent syntax for them.
 - 5) A bottom-up version of the compiler.
7. The major discrepancies are:
 - 1) Differences in the handling of EOLN.
 - 2) Sets are limited to 64 elements (no SET OF CHAR).
 - 3) No subranges in sets, e.g. [1..4].
 - 4) Subscripts are enclosed in parentheses.
 - 5) Set brackets are (↓ and ↓).
 - 6) PACK and UNPACK not supported.
 - 7) Dynamic values cannot be freed.
 - 8) All files must be declared globally.
 - 9) Single precision REAL arithmetic.Additions include:
 - 1) Additional formatting capability.
 - 2) I/O for arrays, records, and all scalar types.
 - 3) Modified handling of labels and gotos.
 - 4) Direct Access Files.
 - 5) Formatting for READ and WRITE is different.
8. Compiler written in PASCAL (5500 lines). Compiler size is 108K + 8K runtime support. Compiles itself in ~78 seconds on IBM 370/158.
9. Runtime support for files has essentially no diagnostics. Real number formatting is less than ideal. Post-mortem of variables is not available on all errors. The compiler has been used for three semesters here with little difficulty.
10. The compiler was written in PL/1 using the unrevised compiler for the CDC6600 as a model. This was done by two undergraduates, Doug Dunlop and Mark Gillette, and required an academic year. The bootstrap took another year of less-than-intensive effort. Improvements have been added as more users have dictated. A major rewrite of the runtime system is in progress.
11. No, but should be available this year. No separate compilation planned at this time.

IMPERIAL COLLEGE OF SCIENCE AND TECHNOLOGY



DEPARTMENT OF COMPUTING AND CONTROL
180 Queen's Gate
London SW7 2BZ Telephone: 01-589 5111
Telegrams: IMPCOL London SW7 Telex: 261503
Head of Department Professor J.H. Westcott, DSc, FBCS, FIEE

21/12/77

- 3. Full Standard PASCAL is supported with several notable additions,
 - full file support, including in-store files, random-access files, explicit opening of files, non-standard opening of files for interactive use;
 - external procedures and functions;
 - alfa variables and returncode in program heading;
 - symbolic post mortem dump;
 - cross reference option.

The compiler is not very fast (about 65 lines per CPU-second), but we consider it now very reliable.

- 4. Along with the compiler comes a special batch system running in CMS environment. It has been adapted and extended for our purposes by Thomas Habernoll. It has drastically reduced time-consuming system overhead (re-initialization of the CMS batch machine).

By fall, we have switched - at last - our introductory course from ALGOL to PASCAL, with excellent results so-far.

THANKS TO LONDON !

A. Biedl
(A. Biedl)

L. Christoph
(L. Christoph)

Thomas Habernoll
(T. Habernoll)

Andy Mickel
Pascal Users Group
University Computer Center:227EX
208 S.E. Union Street,
University of Minnesota
Minneapolis.
MN55455
USA

Dear Andy,

We are using Pascal on our tiny 370/135 under CMS, to support our undergraduate course, for systems programming and in a number of research projects.

Greg Pugh, one of our research students implemented our compiler from the P4 compiler (* described in the attached letter and report *)

Our undergraduate teaching began using Algol on the 7094, and moved to Delft Algol when the 370/135 appeared. Because we wanted to use a ~~rich~~ set of data structures and types than those available in Algol 60 we changed our initial teaching language to AlgolW. After two years with AlgolW we have again changed our initial teaching language - this time to Pascal. There were a number of reasons for this change:

- 1) Some of us had regarded the choice of AlgolW as a stop-gap until a reliable Pascal compiler was available under CMS. This we now believe has happened with the P4 implementation at IC.
- 2) We use a number of different computers during the undergraduate course, and we found that 'standard' Pascal was the only programming language that we felt able to use on a Computer Science Course, which had similar implementation across these machines (IBM370/135, CDC Cyber 173, ICL 1900 & CDC 7600)
- 3) It was clear that 'standard' Pascal included a number of features which would "extend the life" of our initial programming language, putting off the time when any additional language need be taught to provide a practical tool for course support.
- 4) We had a number of bad experiences with the AlgolW Compiler. For example some large valid programs, would fail to execute, however seemingly arbitrary changes to them, got over the problem.

(* The Berlin implementation mentioned is listed separately, below. *)

We attempted to look at the compiler to fix the problems but found it an almost impossible task to understand the commentless PL360. The P4 Compiler (* the first pass anyway *) being written in Pascal (* although with few comments *) is generally understandable and therefore possible to correct. Greg tells me the main areas where he met problems was in the reliance of CDC character set ordering.

Its too early to say whether the change over has made any great impact on the course or on the students' programming ability - it can be said with some certainty that they are no worse than other years ! We believe that by starting off their programming in a well disciplined environment, that students make better and perhaps a more cautious use of other programming environments they meet later on.

Our final year students carry out group projects. Some of them are using Pascal to implement the nucleus of a multiprogramming operating system and others are implementing an algebraic manipulation package in Pascal. (* Attempts to persuade the Data Processing project groups to use Pascal Failed ! *)

In the area of systems programming we have made a great deal of use of Pascal. The P4 implementation is well integrated into the CMS system and behaves exactly like any of the other compilers. By accepting the basic architecture imposed by CMS, the P4 compiler has avoided any problems about "being special" or "being an exception". This has aided in user acceptability and made it much easier to use for systems programming. Work is now in progress on implementing out batch subsystem scheduler in Pascal. Various system utilities, and our central file manager which supports multi-user access to the file base, have been written in Pascal.

The research use of Pascal is quite diverse (* program development systems, micro-assemblers, multi-user editors *) and our research students have produced a number of useful tools to aid nice program production (* POLISH - a very powerful reformatting program, SPUP - a Pascal system similar to CDC Update *). A research group in the department piloted the ICL 2900 implementation of Pascal to use Pascal in research into operating system architectures.

Pascal, then is in wide use within the Department of Computing and Control here at Imperial College !

Please keep up the good work of PUG !

Best wishes to you all,

Iain Stinson

(* Greg Pugh wrote the following checklist. *)

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Iain Stinson (distributor) and Greg Pugh (implementor); Department of Computing and Control; Imperial College; London SW7; England; Phone: 01-589-5111, extensions 2700 and 2758.

2. MACHINE. IBM 370 (not 360s). Our machine is a 135 with 384k. It operates a student service within the department with some terminal work (up to 8 terminals) and a batch stream, Pascal is the main teaching language.

3. SYSTEM CONFIGURATION. Operates under the VM/370 system using the CMS subsystem. It would take quite a bit of work to convert the system to run under any of the other IBM systems. The compiler should run on any machine that runs VM/370 (i.e. model 135 and upwards, >= 256k).

4. DISTRIBUTION. The current version is available for distribution now. Distribution is by 9-track magnetic tape in standard CMS tape dump format. Cost is just postage plus cost of tape (or provide your own tape).

5. DOCUMENTATION. A user's manual is available replacing chapters 13 and 14 of the Jensen and Wirth User Manual and Report. This includes details of all departures from standard and additional features plus descriptions of error messages, invoking commands, etc.

6. MAINTENANCE. Currently we cannot offer any guaranteed support, since we are very short of manpower, however we are using it for teaching ourselves so problems will probably be fixed if you tell us about them.

7. STANDARD. The compiler provides a number of additional standard procedures and a couple of small extensions to the language syntax. The compiler gives warnings when non-standard features are used. There are a few minor restrictions. The following summarises the differences:

Extensions

Hex and Octal numbers are supported.

Expressions can be used to define the value of constants.

External procedures can be declared.

Some 20 additional standard procedures are available, including DATE, TIME, DAY, MONTH, CPU time, routines to do direct access I/O on files, and several routines allowing VM/370 commands to be issued from the program.

ALFA variables can be specified on the PROGRAM statement as well as files.

Restrictions

Sets are 0..255

Files of files are not allowed, but files can appear within other structures (i.e. arrays and records of files are o.k.).

Dispose doesn't work (the dread Mark and Release can be used).

Packed is accepted but ignored (character strings are always packed anyway).

Features

We accept upper and lower case identifiers with 30 characters significant.

Integers are 32 bits and reals 64 bits.

The compiler produces listings with bold printed keywords, titles, nesting level, etc.

8. MEASUREMENTS. The compiler is a much developed version of the P4 compiler. An assembler for the P-code runs as a second pass producing standard object code. The P4 compiler is now about 7000 lines of Pascal, the assembler is 5000 lines of PL370. The run-time system consists of about 90 small modules (in Assembler F) which are included by the loader on a by-need basis (a small program may only use about a dozen of these routines). I'm not sure about exact compilation speed, but it is faster than IBM Fortran and slower than AlgolW. The compiler generates re-entrant code and is shared between all users (which saves a lot of store since the compiler code is quite large).

9. RELIABILITY. Reliability seems pretty good. Currently the compiler is being used mainly for fairly large programs (4000 lines). A student version is now available and should put the system to a severe test.

10. DEVELOPMENT METHOD. The compiler was developed from the Zurich P4 compiler by writing an assembler for the P-code in PL370 in 1976. The run-time library was written at about the same time. A symbolic dump package dumping all variables (including records, arrays etc.) is available. A batching version is available for student use.

11. LIBRARY SUPPORT. Procedures can be declared to be external and compiled programs can be placed into object code libraries. Assembler and Fortran procedures can be accessed.

Modular PASCAL 370 Compiler	12/01/77
	Page 1

1. Implementor

Thomas Habernoll
Technische Universitaet Berlin
{Fachbereich Informatik}
{Institut fuer angewandte Informatik}
DV-Grundausbildung
VSH 5
Otto-Suhr-Allee 18/20
D-1000 Berlin 10

2. Machine

IBM /360, /370
{Implementation is done on a 370/158}

3. System Configuration

VM 370 (CP+CMS)
OS {with some modifications of the run time system}

4. Distribution

Not yet, probably starting fall 1978.

5. Documentation

A detailed reference manual will be available in machine retrievable form.

6. Maintenance Policy

Not yet determinated.

7. Standard

Full standard PASCAL. The compiler is the first step to a 'compiler family'. Later members of this family will include extensions (especially for system programming). But they will not be named 'PASCAL compiler'! (because PASCAL is the language defined in the report of Jensen/Wirth: User Manual and Report. At this level, enough difficulties arise for portability of programs.)

8. Measurements

Presently, no precise information can be given.

9. Reliability

Hopefully good.

10. Development method

The compiler is written in PASCAL. The syntax analysis is based on the PASCAL 6000 compiler, which influenced the method of code generation too.

The run time system is written in PASCAL and Assembler. I hope that the Assembler part can be reduced to a minimum.

As I started working we didn't have a PASCAL compiler useful for bootstrapping on the /370 at Technical University. The Stony Brook and Manitoba compilers were not sufficient. Therefore I wrote a PASCAL->Simula compiler (in SIMULA). It is slow (compilation speed 10 lines per CPU second). I have learned something due to this project: don't think that it is a simple and straightforward task to translate a high level language into another (relatively similar) high level language.

Just as I finished the PASCAL->SIMULA compiler, we got the PASCAL P4 compiler from Imperial College London. This facilitated my work.

I am trying now to write a 'modular compiler'; that means, the compiler architecture allows changing of the accepted language and/or of the generated code by exchanging only few modules. Therefore we will have not one compiler for Standard PASCAL and another one for a subset (e.g. for educational purposes) but a set of modules. Depending on the command, the modules necessary for a specific purpose will be loaded. Writing modules (in the sense of MODULA) in a block structured language makes fun - if one has grim humour.

Presently the compiler is a one pass compiler. But its structure allows splitting it into a multipass compiler.

The first version to be released will accept Standard PASCAL and produce (for several reasons) IBM/370-Assembler code. Later versions will produce relocatable machine code (in standard OS loader format). As a vision the following versions exist:

- PASCAL subset for educational purposes;
- PASCAL supersets.
First level: machine independent extensions like else or otherwise in case statements; a simple module concept.
Second level: extensions for systems programming.
- Generation of interpretative code. This is normally the first step to implement PASCAL, but I think that interpreters are a good tool to check out programs - even for large ones (one of these reasons is the fact, that additional check out code and test aids need additional space resulting in frustrated programmer of large programs, who fights against storage constraints.

Another reason is a possible step-by-step execution of interpretative code, that is more useful than executing step-by-step on machine instruction level.)

- Optimizing compiler.
- Processing of pre-scanned source programs (assume programs to be edited and analysed on a mini computer, the processed source being sent to the /370 for compilation and executing.

Someone somewhere interested in joining this project should please contact me. To have a chance, PASCAL must provide a better support than other languages. Therefore more software aids besides the compiler must be provided (for example the embedding of the compiler in an editor system is a useful task).

11. Library support

Linkage to FORTRAN subroutines is without any problem. Separate compilation is available. The first release provides no symbolic dump. In the event of a run time error the following informations will be given: error message, corresponding source line number and a back trace of procedure calls (with source line numbers of calls).

ICL -- Clearing House

(* David Joslin sent us the following announcement on 13 October 1977 *)

PCHICL - the Pascal Clearing House for ICL machines - has been set up for the purposes of:

- Exchange of library routines;
- Avoidance of duplication of effort in provision of new facilities;
- Circulation of user and other documentation;
- Circulation of bug reports and fixes;
- Organisation of meetings of Pascal users and implementors;
- Acting as a "User Group" to negotiate with Pascal 1900 and 2900 suppliers.

There are currently about 40 people on PCHICL's mailing list, mainly in Computer Science departments and Computing Centres of U.K. Universities and Polytechnics. Any User of Pascal on ICL machines whose institution is not already a member of PCHICL should contact

David Joslin,
University of Sussex Computing Centre,
Falmer, Brighton, Sussex, BN1 9QH, United Kingdom.

All ICL Pascal users are urged to notify David of any bugs they find, any compiler modifications they make, any useful programs or routines or documentation they have written, anything they have that may be of use or interest to other users.

ICL 1900 (Belfast)

(* Thanks to Judy Mullins and to David Joslin who wrote on 4 Nov. 1977 and on 13 Oct. 1977, respectively, to correct the information which we printed in issue #9-10. David provided the following revised checklist. *)

1. Implementor & Distributor: Dr. J. Welsh,
Dept. of Computer Science,
Queen's University,
BELFAST, N. Ireland, BT7 1NN.
2. Machine: ICL 1900 series.
3. Operating System: Any (although the Source Library facility is only possible, and the Diagnostics package only practicable, under GEORGE 3 or 4).
Minimum Configuration: The compiler needs at least 36K of core, and uses CR, LP, DA files.
4. Distribution: Free. Send a mag. tape (either 9-track PE 1600 bpi or 7-track NRZI 556 bpi) to Belfast.
5. Documentation: Belfast's Users' Guide (supplement to Revised Report) and implementation documentation is distributed with the compiler.
Glasgow's supplement to the Revised Report is available from
Bill Findley or David Watt,
Dept. of Computer Science,
University of Glasgow,
GLASGOW, Scotland, G12 8QQ
(who produced the Diagnostics package).
6. Maintenance Policy: No formal commitment to maintenance.
No plans for development in near future.
Send bug evidence to Belfast, and also a note of the bug to PCHICL (see separate notice) who circulate bug reports & fixes to their members.
7. Implementation Level: The language of the Revised Report, with:
Exceptions: There are no anonymous tag fields;
FILES cannot be assigned, passed as value parameters, or occur as components of any structured type;
Predefined procedures & functions cannot be passed as actual parameters;
The correct execution of programs which include functions with side-effects is not guaranteed.

ICL 2900 (Southampton)

(* Thanks again to David Joslin and Judy Mullins (see ICL 1900) for sending revised information. *)

Only the first 8 characters of identifiers are significant.
SETs are limited to $x..y$ where $0 \leq \text{ORD}(x) \leq \text{ORD}(y) \leq 47$.
The ICL 64-character graphic set is used for type CHAR.
PACKED is implemented, and TEXT = PACKED FILE OF CHAR.
ALFA = PACKED ARRAY [1..8] OF CHAR.

Extensions: VALUE and DISPOSE are implemented.
Integers may be written in octal.
Additional predefined procedures & functions DATE,
TIME, MILL, HALT, CARD are provided, and procedures
ICL, ADDRESSOF, allow use of inline machine code.

8. Compiler Characteristics: Compiler, producing absolute binary machine code.
The compiler is written as c. 14000 lines of PASCAL & c. 3500 lines of Assembler Language (Issue 3). Performance is better than most other ICL 1900 language processors (exceptions are incore compile-and-go batch systems of the the WATFOR type).
The postmortem analysis program is written as c. 2500 lines of PASCAL.
9. Reliability: Very good.
10. Method of development: Complete rewrite of original PASCAL 1900 compiler, which was bootstrapped from the original Zurich CDC compilers by Welsh & Quinn.
11. Libraries: Source Library facility available under GEORGE 3 & 4.
Efforts are to be made to develop a PASCAL version of the NAG library (which would be applicable to any machine, not just ICL).
No facilities for separate compilation or for inclusion of semi-compiled routines written in any other language.

D.A.Joslin
October 10th 1977



1(a). Implementation

ICL are providing support for an implementation group based on the University of Southampton.

Project Supervisor:

Dr.M.J.Rees,
Computer Studies Group
Faculty of Mathematical Studies,
The University,
Southampton, SO9 5NH,
England.

Telephone: 0703 559122 x2270

Implementors:

J.J.M.Reynolds,
Computer Centre,
Queen Mary College,
University of London,
Mile End Road,
London, E1 4NS,
England.

H.J.Zell,
New Huxley Building,
Imperial College,
University of London,
London SW7 2AZ,
England.

Telephone: 01 980 4811 x778

Telephone: 01 589 5111

1(b). Distribution

The Pascal compiler will be distributed as a standard ICL program product. Contact should be made to the nearest ICL sales office. Failing that, contact the project supervisor shown above.

2. Machine

The ICL 2900 series machines 2960, 2970 and 2980.

3. Operating System

VME/B and VME/K

4. Method of Distribution

As per ICL standard, see 1(b) above.

5. Documentation

Standard ICL manuals will be available:

- (a) Pascal Language Manual: operating system independent aspects of the Pascal Language.
- (b) Running Pascal Programs on VME/B and VME/K: information on how to run Pascal programs under the operating systems.

6. Maintenance Policy

Full maintenance will be provided by the implementation group and/or ICL while the compiler is offered as a standard ICL program product. The usual ICL procedures for bug reports will be adopted.

7. Language Standard

The compiler implements all features of the language as described in the User Manual and Report (Jensen and Wirth, 1974).

8. Translation/Execution Mechanism

The compiler is written in Pascal and produces Object Module Format (OMF) compatible with all standard ICL compilers. The OMF module may be directly loaded or linked with other OMF modules.

The source listing is approximately 10000 lines of Pascal and produces 80K bytes of code. Approximately 160K bytes of store are required to compile the compiler.

9. Reliability

Current reliability is moderate to good.

10. Method of Development

The compiler was bootstrapped using the 1900 compiler from the Queen's University of Belfast, Northern Ireland as a base. , Northern Ireland as a base.

Twenty four person-months of effort from experienced programmers were required.

11. Libraries

As the compiler produces OMF modules, separate compilation and the inclusion of external procedures will be possible providing the necessary operating system facilities are present.

Intel 8080 (San Diego)

For information, see Ken Bowles' article "Status of the UCSD Pascal Project", and for an order form see the DEC LSI-11 (San Diego) implementation note.

Intel 8080 (Oslo)

David Barron sent us a short piece that appeared in the 17 November 1977 issue of Computing describing an 8080 implementation distributed by the Norwegian company, Mycron, of Oslo. The compiler is said to run in 29K bytes. (* We would appreciate receiving any more information about this implementation that a PUG member could provide. *)

MITS Altair 8800

See DEC LSI-11 (San Diego).

Motorola 6800 (St. Paul)

Mark Rustad asked (30 October 1977) that we make the following updates to his checklist printed in Pascal News #9-10.

1. Work phone: 612/376-1143.
2. Developed using a Motorola 6800 Exercizor (48K, dual floppies) and MITS Altair 680b and SWTP 6800. Should be extremely easy to adapt to any system using the M6800 chip.
3. Requires 32K bytes and an ASCII terminal. Also, a high speed I/O device (floppy disk, cassette, or data cartridge) is highly recommended to reduce loading time to a reasonable amount.
4. The system is being distributed by Computer Depot (3515 West 70th Street; Minneapolis, MN 55435; (612/927-5601)) for less than \$100 for documentation, binary, and interpreter source.
5. Documentation is the responsibility of the distributors. Mark is providing a machine retrievable supplement to the Pascal User Manual and Report to the distributors.
6. Mark guarantees maintenance only to the distributors. They are expected to pass bug reports, etc., to him. Future plans include full acceptance of upper/lower case with mapping of reserved words to single case, and separate compilation of procedures.
7. The following are not supported: with and goto statements; real arithmetic and the transcendental functions; pack and unpack. The compiler handles real arithmetic but the present interpreter does not. The system is designed to make it easy to interface the interpreter to a floating point package or a hardware floating point chip. The following extensions have been made: predefined procedure exit and halt. As of 77/10/23, only lower case is recognized.
8. Interpreter requires 4K (with floating point package). No compilation speed was provided. The interpreter is unbuffered and can keep up with typing speeds of 10cps. Approximately 2-5K M-code instructions are executed per second. This is at least 5-10 times faster than SWTP BASIC.
9. As of 77/10 the compiler had successfully compiled itself in the 6800. Was released to two external sites for testing.
10. As of 77/11 about 3 person-months had been invested. The compiler source is about 2400 lines.
11. The compiler and interpreter are completely relocatable, may be located anywhere in address space where sufficient memory exists. Presently this memory must be contiguous, but it is planned to change this in the future. A crude but usable method for calling external (assembly code) procedures exists. No direct parameter passing is available - this must be done via the stack.

THE UNIVERSITY OF HULL

HULL HU6 7RX. ENGLAND

Telephone: Hull 46311

Department of Computer Studies

15th December, 1977

Mr. Andy Mickel, Editor, "Pascal News", University Computer Centre, 227 Experimental Engineering Building, University of Minnesota, MINNEAPOLIS. MN 55455 U.S.A.

Dear Andy,

The main purpose of this letter is to let you have a copy of the enclosed implementation notes. As we're only about half way through building our PASCAL compiler for the PRIME 300, we've left some of the details vague. We'll send you a fuller set of implementation notes sometime in 1978.

We started our compiler because we couldn't find a suitable version of PASCAL on PRIME 300 computers. The only other implementation that we know of is an implementation of Per Brinch Hansen's Sequential PASCAL. We did some work in early 1977 taking bugs out of this implementation and it is now available from PRIME Computer International, Bedford, England. We don't like it because: (a) Sequential PASCAL doesn't permit nested procedures, (b) it's very slow which makes it useless for teaching purposes.

We currently teach ALGOL 60 to our Computing Science undergraduates. After much debate within the department, we will be teaching PASCAL to our first years from October 1978. For this purpose we'll be using the Belfast Mk.2 compiler as the University's mainframe is an ICL 1904S.

The conversion to PASCAL would be easier if there was a book which properly teaches programming! We haven't found one yet. Addison Wesley are bringing out a book in early 1978 which we think may be suitable - it's entitled "Programming in PASCAL" and written by PUG member Peter Grogono.

Finally, we'd like to say what an excellent job you're doing with "Pascal News". It stimulated us into implementing PASCAL-P for PRIME 300s and is keeping us well informed of what's happening elsewhere. Thanks.

Yours sincerely,

Barry Ian Robson

Barry Cornelius. Ian Thomas. Dave Robson.

- 1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER: Barry Cornelius, Ian Thomas or Dave Robson; Department of Computer Studies, University of Hull, Hull, HU6 7RX, England; Hull (0482) 497951.
2. MACHINE: PRIME 300.
3. SYSTEM CONFIGURATION: Our own system is currently 48K words running under PRIMOS-3 Revision 10 (but Revision 14 sometime in 1978).
4. DISTRIBUTION: We hope to have a first release of the compiler completed by April 1978. No details about distribution have been arranged yet.
5. DOCUMENTATION: None - see details of PASCAL-P elsewhere in "Pascal News".
6. MAINTENANCE POLICY: We intend to correct reported errors for the next few years. Error reports and updates will be sent at irregular intervals to all those who receive the compiler.
7. STANDARD: PASCAL-P subset of Standard PASCAL.
8. MEASUREMENTS: No details are yet available.
9. RELIABILITY: No details yet.
10. DEVELOPMENT METHOD: The code generation parts of the PASCAL-P4 compiler are currently being rewritten to generate PMA which can then be assembled. The first version of the compiler is being tested using the Belfast Mk.2 ICL 1900 compiler. It will be bootstrapped on to the PRIME by using a P-code interpreter for the PRIME written in CORAL 66. The work has been done on and off since June 1977 by five people - some have now left and some learnt CORAL 66 and PASCAL during this time!
11. LIBRARY SUPPORT: No facilities for external procedures are currently available but they may be developed in the future.

Univac 1100 (Madison)

A short note appeared in the 24 October 1977 issue of the University of Wisconsin Madison Academic Computer Center newsletter, MACC NEWS, which stated that "The UW-Pascal compiler is now fully supported by MACC." Two versions, the relocatable, and the load-and-go, were mentioned.

Zilog Z-80 (San Diego)

For information, see Ken Bowles' article "Status of the UCSD Pascal Project", and for an order form see the DEC LSI-11 (San Diego) implementation note.

PASCAL NEWS #11 FEBRUARY, 1978 PAGE 103

INDEX TO IMPLEMENTATION NOTES

General Information

 #9&10: 60.
 #11: 70.

Checklist

 #9&10: 60.

Applications

 #11: 70.

Software Tools

 #9&10: 61.

Portable Pascals

Pascal-P

#9&10: 61-62.
 #11: 70-72.

Pascal Trunk

#9&10: 62.

Pascal J

#9&10: 62.

Pascal Variants

Concurrent Pascal

#9&10: 63.
 #11: 72-74.

Modula

#9&10: 63.
 #11: 74.

Pascal-S

#9&10: 63.
 #11: 72.

Feature Implementation Notes

Set of Char

#9&10: 64-66.

For Statement

#9&10: 66-69.
 #11: 79-80.

Default Case

#9&10: 69-70.

Variable Parameters

#9&10: 71.

Interactive I/O

#9&10: 71-72.

Unimplementable Features

#11: 75.

Long Identifiers

#11: 78-79.

Boolean Expressions

#11: 76-78.

Machine Dependent Implementations

Alpha Micro Systems AM-11

See DEC LSI-11.

Amdahl 470

See IBM 360, 370.

Andromeda Systems 11-B

#11: 80.

Burroughs B1700

#9&10: 73.

Burroughs B3700, 4700

#9&10: 73.

Burroughs B5700

#9&10: 74.

#11: 81.

Burroughs B6700, 7700

#9&10: 74-75.

#11: 81.

CDC Cyber 18 and 2550

#9&10: 75.

#11: 81-82.

CDC 3200

#9&10: 75.

#11: 82.

CDC 3300

#9&10: 75.

CDC 3600

#9&10: 75.

CDC 6000, Cyber 70, Cyber 170

#9&10: 76.

#11: 82-83.

CDC 7600, Cyber 76

#9&10: 76.

#11: 83.

CDC Omega 480-I, 480-II

See IBM 360, 370.

CDC Star-100

#9&10: 77.

CII Iris 50

#9&10: 77.

CII 10070, Iris 80

#9&10: 77-78.

Computer Automation LSI-2

#9&10: 78.

Cray Research Cray-1

#9&10: 78-79.

Data General Eclipse

#9&10: 79-80.

#11: 85.

Data General Nova

#9&10: 79-82.

#11: 83-85.

DEC PDP-8

#9&10: 82.

#11: 85.

DEC LSI-11 and PDP-11

#9&10: 82-88.

#11: 86-91.

DEC DECSYSTEM-10

#9&10: 89-91.

#11: 91-92.

Dietz MINCAL 621

#9&10: 91-92.

Foxboro Fox-1

#9&10: 92.

Fujitsu FACOM 230

#9&10: 92.

Harris / 4

#9&10: 92-93.

Heathkit H-11

#9&10: 93.

Hewlett Packard HP-21MX

#9&10: 93.

#11: 92.

Hewlett Packard HP-2100

#9&10: 93.

Hewlett Packard HP-3000

#9&10: 94.

Hitachi Hitac 8700, 8800

#9&10: 94.

Honeywell H316

#9&10: 94.

#11: 93.

Honeywell 6000

#9&10: 94-95.

#11: 92-93.

IBM Series 1

#9&10: 95.

IBM 360, 370

#9&10: 95-101.

#11: 93-100.

IBM 1130

#9&10: 101.

ICL 1900

#9&10: 101-102.

#11: 100-101.

ICL 2900

#9&10: 102.

#11: 100, 101-102.

Intel 8080, 8080a

#9&10: 102-103.

#11: 102.

Interdata 7/16

#9&10: 103.

Interdata 7/32, 8/32

#9&10: 103-104.

ITEL AS/4, AS/5

See IBM 360, 370.

Kardios Duo 70

#9&10: 104.

Mitsubishi MELCOM 7700

#9&10: 104-105.

MIT S Altair 680b

See Motorola 6800.

MIT S Altair 8800

See DEC LSI-11.

MOS Technology 6502

See DEC LSI-11.

Motorola 6800

#9&10: 105.

#11: 102.

Nanodata QM-1

#9&10: 105.

NCR Century 200

#9&10: 105.

Norsk Data NORD-10

#9&10: 106.

Prime P-300

#11: 103.

Prime P-400

#9&10: 106.

SEMS T1600, SOLAR 16/05/40/65

#9&10: 106.

Siemens 330

#9&10: 107-108.

Siemens 4004, 7000.

#9&10: 108.

Telefunken TR-440

#9&10: 108.

Terak 8510

See DEC LSI-11.

Texas Instruments TI-ASC

#9&10: 109.

Texas Instruments 9900/4

#9&10: 109.

Univac 90/30

#9&10: 109.

Univac 90/70

#9&10: 109.

Univac 1100

#9&10: 109-112.

#11: 103.

Univac V-70

#9&10: 112.

Varian V-70

See Univac V-70.

Xerox Sigma 6, 9

#9&10: 112.

Xerox Sigma 7

#9&10: 112.

Zilog Z-80

#9&10: 112.

#11: 103.

POLICY: PASCAL USER'S GROUP (77/12/30)

Purposes: Pascal User's Group (PUG) tries to promote the use of the programming language Pascal as well as the ideas behind Pascal. PUG members help out by sending information to Pascal News, the most important of which is about implementations (out of the necessity to spread the use of Pascal).

The increasing availability of Pascal makes it a viable alternative for software production and justifies its further use. We all strive to make using Pascal a respectable activity.

Membership: Anyone can join PUG: particularly the Pascal user, teacher, maintainer, implementor, distributor, or just plain fan. Memberships from libraries are also encouraged.

See the ALL PURPOSE COUPON for details.

FACTS ABOUT Pascal, THE PROGRAMMING LANGUAGE:

Pascal is a small, practical, and general purpose (but not all-purpose) programming language possessing algorithmic and data structures to aid systematic programming. Pascal was intended to be easy to learn and read by humans, and efficient to translate by computers.

Pascal has met these design goals and is being used quite widely and successfully for:

- * teaching programming concepts
- * developing reliable "production" software
- * implementing software efficiently on today's machines
- * writing portable software

Pascal is a leading language in computer science today and is being used increasingly in the world's computing industry to save energy and resources and increase productivity.

Pascal implementations exist for more than 62 different computer systems, and the number increases every month. The Implementation Notes section of Pascal News describes how to obtain them.

The standard reference and tutorial manual for Pascal is:

Pascal - User Manual and Report (Second, study edition)
by Kathleen Jensen and Niklaus Wirth
Springer-Verlag Publishers: New York, Heidelberg, Berlin
1975, 167 pages, paperback, \$5.90.

Introductory textbooks about Pascal are described in the Here and There Books section of Pascal News.

The programming language Pascal was named after the mathematician and religious fanatic Blaise Pascal (1623-1662). Pascal is not an acronym.

Pascal User's Group is each individual member's group. We currently have more than 1351 active members in more than 30 countries. This year Pascal News is averaging more than 150 pages per issue.

Return to:

University Computer Center
227 Experimental Engineering Building
208 Southeast Union Street
University of Minnesota
Minneapolis, Minnesota 55455 USA

return postage guaranteed

The University of Minnesota is committed to the policy that all persons shall have equal access to its programs, facilities, and employment without regard to race, creed, color, age, sex, national origin, or handicap.