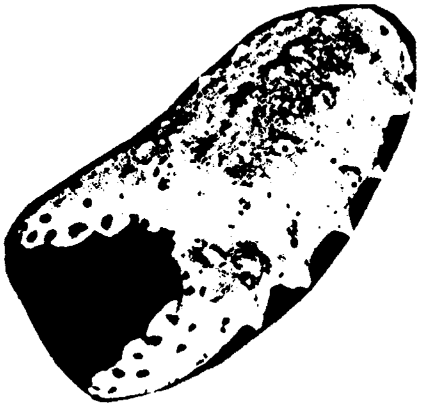
PASCAL USER'S GROUP

Pascal News

December, 1978 NUMBER 13 (oh, how unlucky...)

COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS



POLICY: Pascal News (78/10/01)

* 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 3 or 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 material (brevity is a virtue) for Pascal News single-spaced and camera-ready (use dark ribbon and 18.5 cm lines!

* 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 in the news, history, membership rosters, etc.

APPLICATIONS - presents and documents source programs written in Pascal for various algorithms, and software tools for a Pascal environment; news of significant applications programs. Also critiques regarding program/algorithm certification, performance, standards conformance, style, output convenience, and general design.

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 Portable Pascals, Pascal Variants, Feature-Implementation Notes, and Machine-Dependent Implementations.

* Volunteer editors are (addresses in the respective sections of Pascal News):

Andy Mickel - editor
Jim Miner, Tim Bonham, and Scott Jameson - Implementation Notes editors
Sara Graffunder and Tim Hoffmann - Here and There editors
Rich Stevens - Books and Articles editor
Rich Cichelli - Applications editor
Tony Addyman and Rick Shaw - Standards editors
Scott Bertilson, John Easton, Steve Reisman, and Kay Holleman - Tasks editors

PASCAL USER'S GROUP

USER'S

GROUP

ALL-PURPOSE COUPON

****

(78/10/01)

Pascal User's Group, c/o Andy Mickel ← *Clip, photocopy, or*
University Computer Center: 227 EX ←
208 SE Union Street ← *reproduce, etc. and*
University of Minnesota ←
Minneapolis, MN 55455 USA ← *mail to this address.*

/ / Please enter me as a new member of the *PASCAL USER'S GROUP* for ___ Academic year(s) ending June 30, ______ (not past 1982). I shall receive all the issues of *Pascal News* for each year. Enclosed please find ______. (* Please see the POLICY section on the reverse side for prices and if you are joining from overseas, check for a PUG "regional representative." *)

/ / Please renew my membership in *PASCAL USER'S GROUP* for _____ Academic year(s) ending June 30, ______ (not past 1982). Enclosed please find ______.

/ / 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.

(* The U.S. Postal Service does not forward *Pascal News*. *)

/ / 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.
- please enclose the proper prepayment (checks payable to "Pascal User's Group"); 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.
- please remember that PUG is run by volunteers who don't consider themselves in the "publishing business." We produce Pascal News as 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 minimize paperwork, because we have other work to do.

POLICY

- American Region (North and South America): Join through PUG(USA). Send $6.00 per year to the address on the reverse side. International telephone: 1-612-376-7290.

- European Region (Europe, North Africa, Western and Central Asia): Join through PUG(UK). Send £4.00 per year to: Pascal Users' Group/ c/o Computer Studies Group/ Mathematics Department/ The University/ Southampton S09 5NH/ United Kingdom. International telephone: 44-703-559122 x700.

- Australasian Region (Australia, East Asia - incl. Japan): Join through PUG(AUS). Send $A8.00 per year to: Pascal Users Group/ c/o Arthur Sale/ Dept. of Information Science/ University of Tasmania/ Box 252C GPO/ Hobart, Tasmania 7001/ Australia. International Telephone: 61-02-23 0561.

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 they reprint and distribute Pascal News using a proof copy and mailing labels sent from PUG(USA). Persons in the Australasian and European Regions must join through their regional representatives. People in other places can join through PUG(USA).

RENEWING? (Costs the same as joining.)

- 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. Renewing for more than one year saves us time.

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 £2 each.)
- Issues 9, 10, 11, and 12 (September, 1977 - June, 1978) are available from PUG(USA) all for $10 and from PUG(AUS) all for $A10.
- extra single copies of new issues (current academic year) are: $3 each - PUG(USA); £2 each - PUG(UK); and $A3 each - PUG(AUS).

SENDING MATERIAL FOR PUBLICATION?

- check the addresses for specific editors in Pascal News. Your experiences with Pascal (teaching and otherwise), ideas, letters, opinions, notices, news, articles, conference announcements, reports, implementation information, applications, etc. are welcome. "All The News That Fits, We Print." Please send material single-spaced and in camera-ready (use a dark ribbon and lines 18.5 cm wide) form.
- remember: All letters to us will be printed unless they contain a request to the contrary.

MISCELLANEOUS INQUIRIES?

- Please remember that we will use Pascal News as the medium to answer all inquiries, and we 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

(* This is going to be a long column. I apologize, but many important things need to be said. The future of PUG is one of them! *)

I would like to thank everyone who has helped with Pascal User's Group and Pascal News. Three far-sighted individuals to whom we owe special thanks are close by (at the University Minnesota). Pete Patton is our Computer Center director and Larry Liddiard is our associate director for systems. Phil Voxland is the director of the Social Science Research Facilities Center. Their encouragement and moral support gave us the chance to see Pascal through to its widespread acceptance through the medium of Pascal News.

0. FORTRAN

Being a member (just joking) of ACS and SHAFT (American COBOL Society--dedicated to the elimination of COBOL in our lifetime--and the Society to Help Abolish FORTRAN Teaching, see PUGN #5) I've always wanted to write a short essay like David Barron wrote below. I'd like to add that if the new FORTRAN compilers are written in assembler like most of the old ones, then we should see instability as well. Pascal may make its move on the large machines especially at Universities!

FORTRAN - The End at Last?

D.W. Barron

The apparent indestructibility of FORTRAN as the preferred programming language of users in the physical sciences has long been a discouragement to those of us who try to spread the use of Pascal. We have thought long and hard about ways to convert the FORTRAN faithful, and concluded regretfully that it can't be done. Readers of Pascal News probably don't follow the activities of the FORTRAN Standards Committee, and so will be unaware of recent developments which indicate that the Standards Committee is doing the job for us.

The specification of FORTRAN 77 has recently been published - a botpotch of "features" heaped indiscriminately on the existing FORTRAN language in a way that is not downwards compatible. The Committee has already started work on FORTRAN 82 and has published a draft list of features of the "central module". Since such sacred things as C in column 1 for comments, continuation in column 6 and statements starting in column 7 are apparently to go, the result will not even bear a superficial resemblance to the FORTRAN that present day users know and love.

If you ask a scientist why he uses FORTRAN, his answer will include some or all of the following reasons

i) it is efficient

ii) it is simple

iii) it is universal.

Editor's Contribution

The first reason is a red-herring - Pascal is probably more efficient than FORTRAN on many computers, but the typical FORTRAN user has been brainwashed into believing that nothing can be more efficient than FORTRAN. The simplicity of FORTRAN is superficial - true simplicity comes from logical cohesion. Certainly, FORTRAN 77 can't be described as simple: "feature-oriented" designs rarely are. The really strong card in the FORTRAN pack is its universality. Every computer centre has a FORTRAN compiler, they are reasonably compatible, and the scientist can move his FORTRAN programs from place to place with relative ease. The reason for this is that FORTRAN has been around for a long time, and has been stable for a long time. It is this stability that the new Standards are destroying. Now, FORTRAN may be ANSI or '77. '82 lurks in the wings. These new versions are appreciably different from the old, so FORTRAN loses its identity. What gives a language an identity? Partly its structure, mainly its stability. The FORTRAN philosophy seems to be that FORTRAN is anything to which a particular committee chooses to give that name: at a stroke they have destroyed FORTRAN's most valuable assett.

Whilst rejoicing over this development, we should not lose sight of the moral for Pascal. When it comes to determining the usage of a language, having a good language helps, but most important is to have a stable, widely available language. That way we can reach the situation where everyone knows Pascal and everyone can use Pascal. If we want Pascal to become a universal language, then we must deny ourselves the indulgence of changing it.

I. Recent Events (at least since PUGN #12)

A lot of people responded positively to the new Applications section started in #12. I hope this issue's Applications section is just as worthy. There were also a few comments in favor of regularly featuring "Pascal in Teaching."

It's been quite a while since #12 appeared and even though we have been flooded with renewals and the enthusiastic remarks of "keep-up-the-good-work!", here I am putting #13 out very late. I'm sorry. "13" is turning out to be unlucky indeed. Please see part II.

Part of the reason we're late is that it is hard to keep up with the swirl of events surrounding Pascal. So...

case ImportantEvent of

Employment:
Please see the letter from Chuck Beauregard in the Open Forum section. People have been calling me constantly on the phone for 6 months now trying to find Pascal people to fill jobs they offer. So get the word out: IF YOU KNOW Pascal YOU CAN EASILY FIND A JOB. Down goes another myth! (But, wow, it has been a struggle!)

ConcurrentPascal:
Per Brinch Hansen is trying to survey Concurrent Pascal users. Please respond to his letter in the Implementation Notes section before February 28.

NASA:
The United States National Aeronautics and Space Administration (NASA) is making a strong committment to Pascal. NASA Langley, NASA Ames, and NASA in Houston are all concerned with reliable software in deep-space probes (such as the upcoming Galileo project). PUG member John Knight (who is the CDC-Star Pascal implementor and convenor of the joint SIGPLAN/PUG session at ACM--see below) has been keeping us informed. Thanks, John!

ConventionalizedExtensions:
In #12 we described the formation of an International Working Group on Pascal Extensions which is supposed to decide on a handful of conventionalized extensions. Please see the section in Open Forum on Standards.

Editor's Contribution

Standards:

The British Standards Institute Working Group (DPS/13/4) work on a Pascal standard (see PUGN #14 for a working draft) should help lay to rest much of the standards controversy. A revised version of the document will be accepted by ISO (and therefore ANSI in the United States) in mid-1979. Politics with standards unfortunately keeps growing as knowledge and use of Pascal grows. And unfortunately ANSI has decided to refer Pascal standards business to a subcommittee called X3J9. See Standards in Open Forum.

PascalMachines:

One of our fondest wishes has been that hardware manufacturers help bridge the gap to Pascal by building machines with friendlier architectures. In fact, long-time PUG member Judy Mullins Bishop wrote her PhD thesis under Prof. David Barron investigating just such a Pascal architecture, which among other things, would require minimal storage requirements for object code. Some people keep saying that BASIC (BASICK) exists on small personal computers and that Pascal implementations are too big. The fact is that the popular small personal computers are now based on microprocessors such as the 8080, Z-80, 6800, and 6502 with dinosaur architectures (and with memories too small to do much useful anyway!). Thus a Pascal implementation is at a disadvantage having to emulate actions that should be performed in the hardware to begin with, and therefore consuming more code space. The significant aspect of the widely-known UCSD (University of Calif. at San Diego) Pascal project was to dispel the myth that Pascal couldn't run on a micro. However, in order to be small, this implementation had to be kept interpretive (there exist several cross-compilers of "hard code" for these micros from other sources). Also because UCSD Pascal is a Pascal-P derivative, the P-code had to be modified and packed (frequency-encoded). The result is slower execution.

But, recently, Western Digital built an inexpensive chip-set expressly for running the modified UCSD P-code, and a speedup of 5 or 10 is being realized. National Semiconductor will probably do the job even better in a few months by building something closer to standard P-code that will execute much faster. They have had something similar under wraps for over two years! After being disappointed by Zilog and the Z-8000 a year ago, it is good to see Western Digital take the courageous first step, made possible by the people at UCSD. (Unfortunately we have been receiving altogether too many reports from users that UCSD Pascal is not as stable as it should be, and that its non-standard extensions are particularly lacking in robustness. For an example, see the Implementation Notes section.)

PascalUsage:

The Western Digital product brochure for its Pascal "Microengine" apparently misquoted Ken Bowles of UCSD concerning: "there are more users of UCSD's Pascal today than users of all other versions combined." This patently false statement has caused Ken some embarrassment, and although it's hard to get an exact figure, I'd estimate that nearly 8% (1 in 12) of all Pascal usage is on UCSD Pascal. Most usage is on PDP-11's (non-UCSD) followed by IBM 370's followed by CDC machines and DEC 10's and 20's. To give a specific example, the venerable CDC-6000 implementation is running at over 300 (very large) sites, and at just one of them (our University of Minnesota computer center) the compiler was accessed over 272,000 times from 77/07/01 to 78/06/30 which represented a 68% increase over the previous 12 months. We have been trying to collect usage data through the checklists in the Implementation Notes section and will try to summarize them in one place in a future issue.

ExplosionInIndustryLiterature:

Byte, Electronics, Creative Computing (ROM), and others have run full-length articles on Pascal. In fact the August, 1978 Byte was almost entirely devoted to Pascal! This phenomenon is most encouraging because eventually the mainstream computer literature will have to help carry news about Pascal if it is to supplant BASIC and other crummy languages. Other computer journals (Computer World, for example) have kept Pascal in the news this last six months and we appreciate it. The only bad side effect is that the publicity has literally swamped us here at PUG central with mail and phone calls.

end (* case *) This leads to....

II. Pascal User's Group / Pascal News status

Running Out of Time

Just at the time when the mail was starting to build up last May, (it now runs between 10 and 30 pieces a day), our usually smooth-running operation became short-handed. Jim Miner started going to school full-time. Sara Graffunder delivered a 2.81 kg baby boy named David. (As an aside, Rich Stevens got married two days after Thanksgiving in November!) Not just that, but standards politics, conventionalized-extensions politics, and UCSD workshop politics all began to consume our time with very little in return (just working very hard to stay still). The cover of this issue depicts the situation.

As if that weren't all, I do have my own full-time job to do here at the University of Minnesota computer center. This summer we changed operating systems and character sets. Because I am also involved with the project to produce a new release of CDC-6000 Pascal, I was unable to work on PUG much at all this summer. Our mail went unanswered, and I apologize.

New Members

This is the first academic year (July, 1978 to June, 1979) for almost 1000 new members, and I wanted to assure them good service and information which would tell them what our style is like. But after catching up with the mail in October (and returning all $4 renewals arriving after August 1 thus allowing a 1-month grace period) and falling behind again, I think I have disappointed quite a few people and I'm sorry. We have stated that we are all-volunteer, and that we have little or no secretarial help, but you new members have yet to read this sentence because you have received nothing from us until now! I urge new members to get backissues from last year--see the section on backissues in Here and There.

Deadlines

We have received some sharp criticism from overseas PUG members (who, by the way contribute most of the material for publication!) about the publications deadlines for Pascal News. The fact is that we had no deadlines during 76-77 (issues 5-8) and everything went well. When we began setting deadlines in the first line of the Editor's Contribution (issues 8-12) we never actually met a single one. Because we were always late in producing an issue, overseas members often received issues after the deadline for the next issue.

Solution: let's go back to no deadlines. If you have material, simply send it in.

Confusion

Our mailing list has never been sold or given out. Any PUG members with issues of Pascal News from #9/10 onward has the mailing list, because we print the roster. We have however sent out a notice last month for the jointly-sponsored ACM SIGPLAN (Association for Computing Machinery Special Interest Group on Programming Languages)-PUG session at the national ACM '78 conference this December, and it is already causing confusion. We didn't bring all the renewals up to date, and for many people, this is the first thing to be received from us. If I were on the receiving end I would be confused too! We knew we were going to be late with this issue, and that is why we sent the notice out.

Summary

I hate to paraphrase someone like Winston Churchill, but he said that sometimes doing your best is not enough--sometimes you have to do what is required. Please read on in my open letter in the Open Forum section.

Andy

78/12/01.

TIDBITS

Ole Anderson, Corvallis, OR 97330: "I have a LISP interpreter that runs under the UCSD Pascal system- Would anyone be interested?" (*78/05/19*)

David A. Beers, Santa Ana, CA 92701: "I very much enjoy reading Pascal News. It is a refreshing exposure to rationality when compared to my job as a business systems programmer. ... I have talked to Joseph Mezzaroba of Villanova Universiy concerning his DOS/VS version of the AAEC Pascal 8000, and will be attempting to convert it to DOS unless I hear of someone else's successful endeavors in this area." (*78/10/25*)

C. Y. Begandy, Aluminum Company of America, Alcoa Center, PA 15069: "I recently obtained the Pascal compiler from the DECUS library. Because of daytime core usage restrictions at our installation, it is necessary to decrease the size of the executable program. Any information you might have on other users' experiences in implementing either a smaller version of this compiler, or a segmented version would be greatly appreciated." (*78/05/26*)

Gerd Blanke, Eschborn, Germany: "... MODULA will be running on a Zilog MCS with 64K under Rio near the end of this year!" (*78/10/27*)

John H. Bolstad, Department of Mathematics, Florida State Univ, Tallahassee, FL 32306: "We use Pascal here for almost all computer science courses. The system programmers also use it." (*78/07/11*)

R. T. Boute, Francis Wellesplein 1, B-2000 Antwerpen Belgie: "We are interested in a special hardware support for standard and concurrent Pascal, for example microprogrammed implementations of the P machine." (*78/10/17*)

Robert Boylan, Metromation, Princeton, NJ 08540: "I know a PDP-11 version of Pascal is in existence, but has anyone done one for a Modcomp mini?" (*78/07/26*)

David C. Cline, Westboro, Mass 01581: "Pascal is attracting a lot of attention here at Data General as a takeoff point for a SIL." (*78/05/11*)

Dennis R. Ellis, Cray Research, Boulder, CO 80303: "I have a COPYSF (copy shifted file) implemented on a CRAY-1 written in Pascal using 11 lines of code." (*78/08/07*)

Larry Ellison, Computer Assisted Bible Study, Willingboro, NJ 08046: "I am serving as coordinator for a group of Bible students who are going to use Pascal on various micro-computers to assist in the study of the Bible." (*78/08/09*)

John Fitzsimmons, Edina, MN 55436: "It seems that every issue of PUGN has a few pleas for insertions, deletions, or things they don't like about Pascal. Did it ever occur to those of you who complain that the rest of us like the language as it is?" (*78/06/30*)

Lee Fránk, BTI Computer Systems, Cherry Hill, NJ 08002: "... our Pascal is the systems programming language for our new BTI-8000 and all our compilers are written in it." (*78/06/16*)

Glen Fullmer, Tektronix Inc., Beaverton, OR 97099: "Dear Lord, won't you buy me a new programming language/ My friends all write Pascal/ I must make ammends./ P. S. We could call it 'LACSAP'." (*78/10/31*)

Steven J. Greenfield, Unicorn Systems Company, Los Angeles, CA 90010: "I have been using Pascal for the last six months to write an Assembler designed to generate code for any object computer. Pascal has provided a powerful method of writing a transportable piece of software." (*78/04/25*)

Dale H. Grit, Department of Computer Science, Colorado State University., Ft. Collins, CO 80523: "At CSU, we're using Pascal in all upper level courses... next year, the 2nd course will be Pascal." (*78/08/10*)

Here and There With Pascal

Marc Hanson, Hermosa Beach, CA 90254: "... I would appreciate learning about anyone's experiences with running Pascal on either Xerox or Honeywell equipment." (*78/05/04*)

Sam Hills, New Orleans, LA 70125: "I am implementing AUGMENT (from the last PN) on the DEC-10." (*78/08/14*)

G. Steve Hirst, Iowa City, Iowa 52240: "CONDUIT (a consortium distributing computer-based curriculum materials) is currently investigating including Pascal as a distribution language for new materials." (*78/08/07*)

Claes Hojenberg, University of Uppsala, Sweden: "UDAC is the computer center for the Univ. of Upsala, Sweden's biggest university, and we hope to be able to use {UCSD} Pascal for implementing a data-base management system on microcomputers." (*78/10/06*)

K. B. Howard, College of the Sequoias, Visalia, CA 93277: "We're interested in looking into the possibility of using Pascal (in instucting beginning programming course) for students aiming toward engineering and computer science fields, and are particulary interested in learning of sources for compilers for the language, for PDP-11, HP-3000, and/or Altair 8800 micro if possible." (*78/09/29*)

L. C. Hutchinson, Mentor, OH 44060: "... I would appreciate knowing if there are any Modcomp Pascal users..." (*78/05/15*)

Jose I. Icaza, Universidad Autonoma Metroplitana - Azcapotzalco, Mexico D.F., Mexico: "At this University, we are just starting to use Pascal and giving some optional mini-courses about it. People seem to love the language. Hopefully, soon it will replace FORTRAN as the first language students learn." (*78/10/24*)

Dennis Kalthofer, Philadelphia, PA 19103: "I am starting a workshop in computer science stressing the social aspects of the field. ... I plan to use Pascal as the basis for these systems and any further systems we develop, to organize our programming technique and understanding, and for teaching people about computers and programming in general, as it illustrates many important computer concepts." (*78/07/11*)

Richard H. Karpinski, San Francisco, CA 94114: "Request that software tools' or 'applications' solicit Pascal program modification tools, such as macro generators and programs to make names unique among the first N characters, etc. Praise for UCSD system." (*78/04/18*)

Tom Kelly, Downingtown, PA 19335: "With regard to 'improvements', 'extensions', etc, i wish people would engage brain before putting mouth in gear'. My (substantial) work with several Pascal compilers over past year has shown me what a fantastic job Wirth did!" (*78/07/07*)

Neb Lafert, Hewlett-Packard (Schweiz) AG, Geneva, Switzerland: "... we think that a good relationship should be established between our two organizations, enabling us to help every new request for Pascal in our country." (*78/09/25*)

Jerry LeVan, Dept Math Sc, Eastern Ky Univ, Richmond, KY 40475: "All of our CS majors will be started on Pascal. We are using OMSI's Pascal. I am reasonably happy with the implementation (it will compile and run Pascal-S)." (*78/07/11*)

Stephen A. Locke, Beloit Corporation/Paper Machinery Division, Beloit, WI 53511: "I am interested in Pascal for real-time control of an industrial process... Is there anyone you know working in such a direction?" (*78/06/05*)

Richard C. Lound, San Francisco, CA 94114: "I am an independent software consultant, primarily in communications systems. My interest in Pascal is in its applicability to use for generation of specialized message switching and front-end software."(*78/08/02*)

Wilf Overgaard, Worldwide Evangelization Crusade, Fort Washington, PA 19034: "Where could I locate a general ledger-bookkeeping program, in Pascal, for non-profit organization? ... Where can one find a good word processing program in Pascal?" (*78/08/31*)

Bill Marshall, Sanders Associates Inc., Nashua, NH 03060: "I had hoped to be the first one on my world to implement Pascal on the VAX-11/780, but discovered a group at Univ.

Here and There With Pascal

of Washington already well along toward that goal." (*78/08/28*)

Jim McCord, Goleta, CA 93017: "I am acting as the distributor for UCSD Pascal for hobby users of the LSI-11. Cost is $50, of which $35 goes to UCSD for continued work. Other $15 pays for documentation and postage, if user sends me four floppies. (Else I will provide for $3 each). This includes all source code for everything, including the interpreter. Anybody interested should get in touch with me (we already have 7 users)." (*78/07/17*)

Michael Robert Meissner, University of Minnesota, Minneapolis, MN 55455: "Everybody talks about portability of programs. This summer I ran into the portability of programmers. I found that we can all get locked into thinking and depending on special features of Pascal compilers, and have to 'relearn' Pascal whenever we switch computer systems or compilers." (*78/10/20*)

Anne Montgomery, Lowry AFB, CO 80230: "McDonnell Douglas has developed a CMI/CAI system here on Lowry Air Force Base called the Advanced Instuctional System (AIS). The AIS, as its name implies, is used primarily for technical training. ... The system currently manages approximately 1500 students in four courses over a 12-hour production shift." (*78/10/16*)

Roderick Montgomery, Somerville, NJ 08876: "I am coordinating distribution of UCSD Pascal to amateurs in the Amateur Computer Group of New Jersey, largest surviving hobbiest club in U. S. September meeting of ACG-NJ will be devoted to Pascal." (*78/07/20*)

William Moskowitz, The California State University and Colleges, Los Angeles, CA 90036: "I might add that Pascal at CSUC has been tremendously successful. During the past twelve months we have had 68,603 accesses and usage continues to grow." (*78/07/17*)

David Mundie, 104-B Oakhurst Circle, Charlottesville VA 22903: "I would like to correspond with anyone having first-hand experience with the S-100 bus TI 9900 Pascal system being offered by Marinchip Systems." (*78/10/06*)

John E. Newton, Randolph AFB, TX 78148: "I am specifically interested in identifying members that have implemented Pascal on Burroughs 6700 hardware." (*78/07/20*)

Dave Peercy, BDM Corp., Albuquerque, NM 87106: "We at BDM are becoming a very interested group of Pascal users." (*78/08/28*)

Sergi Pokrovsky, USSR Acad. Sci., Novosibirsk, USSR: "I hope that S. Pitin of the Moscow Computing Center will shortly report to you on his (not so recent) implementation of Pascal for the BESM-6 computer." (*78/10/31*)

Darrell Preble, Georgia State University, Atlanta GA 30303: "GA State Univ. has converted a Pascal compiler from SUNY at Stony Brook. Originally written in XPL, it uses either of two monitors to support interactive or batch use." (*78/09/05*)

David Rosenboom, York University, Downsview, Toronto, Canada M3J 1P3: "My particular interest in Pascal is in obtaining or developing a compiler for use on the 16-bit Interdata machine... Do you know of anyone who has developed a Pascal system for Interdata 16-bit machines?" (*78/09/01*)

Axel Schreiner, University of Ulm, W-Germany: "Using (in Ulm) Torstendahl's RSX-11 Pascal (love it) and Petersen's TR440 Pascal (not quite as stable) in beginner's courses." (*78/06/19*)

Joeseph C. Sharp, Varian, Palo Alto, CA 94303: "I will introduce Pascal to the North Star Users Group this month. A 30 minute talk is scheduled." (*78/10/30*)

Robert J. Siegel, Brooklyn, NY 11215: "Would like to see an article on the relationship of Pascal to ALGOL." (*78/06/23*)

Seymour Singer, Hughes Aircraft Co., Fullerton, CA 92634: "We have installed the SLAC-Stanford Pascal compiler on our twin Amdahl 470 computers." (*78/07/09*)

Jim Smith, Computer Science Dept., School of the Ozarks, Pt. Lookout, MO 65726: "We have recently implemented a Computer Science Department here as the School of the Ozarks, and there is a need to increase the software library in the computer center. We feel that Pascal would be an important language to present in the curriculum." (*78/09/08*)

Craig A. Snow, TRW Communications Systems and Services, San Diego, CA 92121: "We are very interested in using Pascal to implement our future software products." (*78/05/09*)

James A. Stark, Oakland, CA 94609: "Interactive Pascal via UNIX is way ahead of a batch compiler on UCSF's 370/148 but I have no comparison on routine production jobs on either." (*78/07/17*)

Ed Thorland/Walt Will, Computer Center, Luther College , Decorah, IA, 52101: "We are still looking for information on an HP3000 implementation of P-code Pascal. Also need documentation of P-code instruction-format and functions." (*78/07/11*)

P. J. Vanderhoff, Berkel En Rodenrijs, The Netherlands, "What happened to Stony Brook Pascal release 2?" (*78/10/27*)

Eiiti Wada, Division of Engineering, University of Tokyo Graduate School: "In my class, all the examples were switched to Pascal since the fall semester of 1972, and the first Pascal compiler became available in the summer of 1974. Since then at the University of Tokyo, three versions of Pascal compilers have been installed, and all the compilers are intensively used." (*78/09/08*)

Anna Watson, Panama City, FL 32407: "Very fascinating reading in News - must obtain magnifying glass before I go blind though." (*78/05/15*)

Anna Watson, Panama City, FL 32407: "Is there a Pascal for a SEL 32/75?" (*78/10/07*)

John West, Digital Systems Design Group, Atlanta, GA 30327: "Would like any information about latest Pascal-P implementations on Interdata 7/16, 7/32, NCR 8100, 8200." (*78/05/01*)

James A. Woods, Berkeley, CA 94703: "What's wrong with C?" (*78/08/24*)

PASCAL IN THE NEWS

Byte, May, 1978: "Comments on Pascal, Learning How to Program, and Small Systems"; A short article by Gary A. Ford, Arizona State University, which talks about Pascal's advantages and drawbacks with regards to personal computing. "I have used Pascal for at least 95% of my own programming and I cannot recommend it too strongly."

Byte, August, 1978: "Pascal: A Structurally Strong Language"; A 6-page article describing Pascal. Procedures for Infix to Polish conversion, and subsequent code generation for a hypothetical micro are listed and explained.

Byte, August, 1978: "In Praise of Pascal"; A quick survey of Pascal, with descriptions of user-defined scalar types, sets, and pointer type variables. A comparison of a Pascal program and a BASIC program to its corresponding Warnier-Orr logic diagram is given.

Byte, August, 1978: "Pascal Versus COBOL"; Ken Bowles shows how Pascal can be applied to the traditionally COBOL-infested business environment.

Byte, August, 1978: "Pascal Versus BASIC"; A comparison of a program 'MASTERMIND Codebreaker' written in both BASIC and Pascal. Mastermind is similar to the number guessing game 'BAGELS', using colored pegs instead of digits.

Byte, September, 1978: "A 'tiny' Pascal Compiler, Part 1: The P-code Interpreter"; The first in a series of articles describing a Pascal compiler written for an 8080. The first talks about parsing, and grammers, etc. Parts of the P-code interpreter are listed.

Byte, October, 1978: "A 'tiny' Pascal Compiler, Part 2: The P-compiler"; The second part of the previous, this describes the compiler portion.

Byte, November, 1978: The third part of the 'tiny' Pascal series is to be on generating executable 8080 machine code.

Computer Week, May 12, 1978: "Pascal- Everybodys Language?"; A short description of What, Where, and Why of Pascal. "Pascal is named after the 17th century French philosopher, Blaise Pascal. It is not an acronym and is written in lower case."

Computer Weekly, August 24, 1978: "GEC's Pascal"; "A Pascal compiler is being developed by General Electric Computers (GBr) for its 4000 series machines. ... will be available in 1979."

Computer Weekly, September 9, 1978: "Motorola to offer Pascal on MACS"; "Giving futher credence to the view that Pascal could become the dominant high level language of microcomputing, Motorola Semiconductor has revealed that this software will be the prime language supported on its new microprocessor MACS, due to be unveiled early next year."

Computerworld, April 17, 1978: "TI adds Pascal to Mini's Repertoire"; "A Pascal software package said to be suitable for systems applications because its compiler and several software modules are themselves written in Pascal has been introduced by Texas Instruments Inc. for the firm's DS990 packaged disk-based minicomputer systems." 1 year software subscription costs $1,500 to $2,000.

Computerworld, April 24, 1978: "Growth in Use of Pascal called Revolutionary"; A short report, by Richard Cichelli, mentioning that Pascal is available "for the Zilog, Inc. Z80 micro to the Cray Research, Inc. Cray-1 supercomputer and for nearly everything in between." Also, it gives the addresses of the PUG and DECUS/Pascal groups.

Computerworld, May 8, 1978: "Pascal Attractive Anyway"; A Letter to the Editor from Saul Rosen, Purdue University, "Pascal is a very attractive language. Here and at many other colleges and universities, it is used extensively in computer science and computer engineering courses."

Computerworld, May 15, 1978: "Standard Pascal Compiler Runs on PDP-11's"; A description of Oregon Minicomputer Software, Inc. Pascal compiler, known as OMSI Pascal-1, which generates assembly code that can be assembled and linked with DEC system utilities. RT-11 can support this compiler.

Computerworld, May 22, 1978: "Pascal ready for DG users"; An announcement of Rhintek, Inc.'s Pascal compiler for Data General Corp. minicomputers running RDOS. Cost is approximately $1,000.

Computerworld, May 22, 1978: "Northwest Melds 8085A, Pascal"; "Northwest Microcomputer Systems, Inc. has announced a 'programmers workbench' that reportedly combines the throughput of the 3 MHz Intel Corp. 8085A and the power of Pascal." "The 85/P provides the 'full Pascal environment,' according to the spokesman, including random and sequential files, a screen-oriented editor, interactive source-linked debugger and full documentation plus a 90-day warranty." Cost is about $7500.

Computerworld, September 4, 1978: "University Working To Adapt Pascal For MDC-100 Use"; "Programmers here at the University of California are presently under contract to adapt Pascal for use on the American Microsystems, Inc (AMI) MDC-100 microprocessor development center, according to an AMI spokesman."

Computerworld, September 29, 1978: "The Waves of Change", "Implementation languages and the case for Pascal"; one section of the multi-part excerpt of Charles P. Lecht's book, The Waves of Change is devoted to a background of why Pascal is a successful language, and where it is being used. "Pascal is more interesting than other influential, new development languages such as Algol 68, because it is apparent that it was designed for software engineering purposes. (italics in original).

Computerworld, September 25, 1978: "Isam Logic, Disk Space Control Included in Micro-Based Pascal/Q"; An announcement of Pascal/Q, which is an enhanced version of Pascal which "includes support for Qsam, Queue's enhanced Isam file access method, and for automatic disk file storage allocation". Available for $300 plus $19/month for updates from Queue Computer Corp.

Computerworld, October 2, 1978: "DOD Expects Standard Compiler by 1981"; The U. S. Department of Defense's new compiler is planned to be based upon Pascal. There is a plethora of articles on this language (see July Sigplan Notices).

Computerworld, November 20, 1978: "Work on Pascal Progressing"; "A technical committee from the American National Standards Institute (ANSI)] ON Pascal has been approved to work under the X3 committee on computers and information processing. Identified as X3J9, the new groups' initial task is to prepare a proposal for standardization of Pascal and to obtain approval of the proposal ..." Justin Walker of the NBS will convene the first meeting at the CBEMA offices on Tuesday, December 19. "Interested people and organizational representatives are invited to contact Cathy A. Kachurik at Cbema/Standards, 1828 L St. N. W., Washington, DC 20036.

Computerworld, November 20, 1978: "DOD language named"; "'ADA' has been chosen as the name for the forthcoming Department of Defense (DOD) computer programming language. The language was named after the first programmer in history, according to Lt. Col. William A. Whitaker of the DARPA. Ada Augusta, Countess of Lovelace, was one of the few contemporaries of computing pioneer Charles Babbage who understood his work on calculating machines. ... the first funded compiler, produced for the Army is expected in May 1981."

Computing Europe, September 1978: "Steelman ready next April ..."; More on the DoD's new language. Some background on what has been happening, plus some comments by Edsger Dijkstra, who is a critic of the DoD's plans.

Electronic Design 19, September 13, 1978: "Pascal isn't just one more computer language. It promises to be simple, flexible and fast."; "This introduction to the Pascal programming language is the first part of a series, based on ESI's Pascal Instruction Manual. Future parts will deal in detail with Pascal statements, structured data, I/O procedures, advanced programming techniques and real-world applications." This is a good primer to the language. About 5 pages in length.

Electronic Products, July 1978: "As IC it"; bylined by Jerry Metzger. He mentions that several IC houses and minicomputer companies have announced intentions towards using Pascal. "But standards need to be established. The time is right to do this with Pascal."

Electronic Engineering Times, October 16, 1978: "Pascal Implemented in Code of WD's First Computer Offering"; "Pascal has been implemented in the microcode of a new computer from Western Digital Corp., the first in a line of system products to be announced soon, according to the company." "This new system includes a complete Pascal operating system-Pascal compiler, BASIC compiler, file manager, screen-oriented editor, debug program and graphics package- all written in (UCSD) Pascal." Price is about $2,500.

Electronics, October 12, 1978: "Pascal becomes software superstar"; "From the mountain fastness of Switzerland there came 10 years ago a programming language called Pascal. For the first few years of its life it created little stir, but then it began to gain popularity in academia and eventually industry. Today, Pascal is finding its way into machines of all shapes and sizes around the world." This is a good article which gives a brief history, and the current usages of Pascal, from micro's to maxi's and small applications programs to operating systems.

Scientific-Technical Book & Copy Center, Letter to Andy Mickel; "Pascal is our best seller ... We would very much like to see a copy of Pascal News".

Silicon Gulch Gazette, Volume 3, Number 3: "UCSD Pascal On An S-100 System"; "Dr. Jim Gagne of Los Angeles, CA, will ... explain the joys and sorrows of implementing UCSD Pascal on his small computer and the difficulties involved in the project." This is a report on scheduled lectures during The Third West Coast Computer Faire, which took place November 3rd and 4th.

From the preceeding: "A Portable Compiler for a Pascal-like Language"; "... will be described by Mark Green. He will treat the problem of program portability. Three solutions to the problem will be presented. As well, a particular piece of portable software developed for the Micro Pascal Compiler will be examined."

Communications of the ACM, October 1978, back cover: An advertisement for jobs with the Software Technology Company; "develop a compiler for a sophisticated, Pascal-based communications language with real-time multiprocessing features, extensive exception-handling facilites, global data modules, and other state-of-the-art characteristics." "{Softechs} compiler was produced on the UNIX system and later moved to RSX-11."

TimeShare, open letter to PUG members: "TSC has adopted Pascal as the primary implementation language for its LSI11 based products. ... It is, however, difficult to find programmers experienced with Pascal and RT11 (or RSX) and RSTS." TSC is looking for applicants with these qualifications (plus 2-4 years experience).

ATRA

ADVANCED/TECHNOLOGY
RESEARCH/ASSOCIATES
P.O.BOX 456, MINNEAPOLIS, MN 55440

(612) 374-1293 or
PUG member Eric Hand, 2633 Dupont Ave S., Minneapolis, MN 55408 (612) 377-7387, informed us that if you are tired of cumbersome COBOL, obsolete FORTRAN, or Brontosaurian PL/1, you can show your support for the best general-purpose language now available by acquiring a Pascal-POWER T-shirt for $5.95 postpaid from ATRA. Sizes are S, M, L, and XL for a lite-blue shirt with an artistic portrait of a smirking Blaise Pascal.

PASCAL IN TEACHING

This new section will report on experiences with Pascal used for teaching in computer science. The first report is a nice survey done in Australian Universities by Jan Hext from the University of Sydney. Following that is a report from Japan, and one on a CAI system developed at ETH Zurich. Judy Bishop at the University of Witwatersrand in Johannesburg, South Africa, promised to send a description of a Pascal programming contest held for undergraduates. Substantial prizes were given.

The University of Tasmania

Postal Address: Box 252C, G.P.O., Hobart, Tasmania, Australia 7001
Telephone: 23 0561. Cables 'Tasuni' Telex: 58150 UNTAS

11th October, 1978

Dear Andy,

I enclose some information which should be of interest to Pascallers. A friend of mine, Jan Hext from the University of Sydney, has been polling Australian Universities to measure the extent of Pascal's penetration into the teaching area. The sampling is very selective (ie. by membership of PUG!), but many of the institutions not polled would either not teach computer science, or would contribute insignificantly. There are exceptions, of course, notably Monash University - I am reliably informed they are switching over in 1979.

Yours sincerely,

Arthur

Arthur Sale,
Department of Information Science.

TEACHING PASCAL IN 1979

In order to survey the market for Pascal textbooks in 1979, a questionnaire was sent to the universities and colleges listed in the Pascal Users Group mailing list. Three questions were asked:

1. How many students would be learning Pascal in 1979?
2. Would they have learned any other language previously? If so, which one?
3. What textbook would be recommended?

The answers are summarized in the table below. Allowing for a few self-taught students, etc., the main conclusion is that at least 2500 people in Australia will be learning Pascal in 1979, of whom 1900 will be learning it as their first language.

Also listed below are thirteen textbooks on Pascal which are either available or else in press.

The enthusiasm for Pascal may be reflected in the fact that all of the questionnaires were returned without any extra prompting. I would like to express my appreciation to the people who so helpfully answered them.

J.B. Hext

Replies to questionnaires

University or Institute	Introductory Students	As a Second Language: students, first language
Adelaide	350	40, Fortran
A.N.U.	250	-
Melbourne	200 (?)	100, Fortran
Newcastle	-	35, Fortran
N.S.W.	320	-
Queensland	400	100, Fortran
R.M.I.T.	150	-
S.A.I.T	-	100, Cobol
Sydney	-	200, Fortran
Tasmania	120	-
W.A.	100	-
Wollongong	-	60, Basic
Total	1890	595

Textbooks

The following textbooks are either introductions to Pascal or more advanced books that make use of Pascal. Reviews of them are cited from the Pascal Newsletter (PN) and the ACM Computing Reviews (CR).

Addyman and Wilson: "A Practical Introduction to Pascal", MacMillan, 1978, 140 pp.

Alagic and Arbib" "The Design of Well-Structured and Correct Programs", Springer, 1978, 292 pp. (PN#11).

Bowles: "Microcomputer Problem Solving Using Pascal", Springer, 1977, 563 pp. (PN#11).

Conway, Gries and Zimmerman: "A Primer on Pascal", Winthrop, 1976, 448 pp. (PN#12).

Findlay and Watt: "An Introduction to Programming in Pascal", Pitman, 1978.

Grogono: "Programming in Pascal", Addison-Wesley, 1978, 350 pp. (PN#12).

Jensen and Wirth: "Pascal Users Manual and Report", Springer, 1978, 167 pp.

Kieburtz: "Structured Programming and Problem Solving with Pascal", Prentice-Hall, 1977, 320 pp. (PN#10)

Rohl and Barrett: "A First Course in Programming in Pascal", Cambridge University Press, in press.

Schneider, Weingart and Perlman: "An Introduction to Programming and Problem Solving with Pascal", Wiley, 1978, (PN#12).

Webster: "Introduction to Pascal", Heyden, 1976, 129 pp. (PN#8).

Wirth: "Systematic Programming: An Introduction", Prentice-Hall, 1973.

Wirth: "Algorithms and Data Structures = Programs", Prentice-Hall, 1976.

NIHON UNIVERSITY

COLLEGE OF INDUSTRIAL TECHNOLOGY

Izumicho Narashino Shi
Chiba 275 Japan

A Report from College of Industrial Technology 78/08/03
Nihon University, Japan

Prof.H.Shima feel strongly the fruitfull effect of utilizing the Pascal language in computer science education, and so he utilize that language in his class.
The year Prof.H.Shima started to introduce the language to the computer science course of the department of mathematical engineering was 1976' academic year and 30 students attended' to it's seminar. The first semester of 1977, he utilized Pascal for 110 students of junior enroll to the department in computer science class, and all these times they used "Systematic Programming: An Introduction"(Prentice Hall'71:-Translated to Japanese Edition) as a text.

Now, in 1978' academic year, on both former term and later term he use mainly Pascal in his class for computer science education, referring "Algorithm + Data Structures = Programs"(Prentice Hall'76) and using a text note which Prof.H.Shima himself edited for his junior level students and they belong to the department of mathematical engineering.

Students are served to use concurrent Pascal compiler for their practice and it is implemented by Assistant Prof.J.Ohshima on his laboratory minicomputer(Facom U-mate).

XS-0

In the Apr/May/Jun 1978 issue of the AEDS Monitor, an article appeared entitled XS-0 "XS-0: A Self-Explanatory School Computer" by J. Nievergelt. The paper was presented at the NAUCAL 1977 Fall Computer Conference in Dearborn, MI. Nievergelt is with the Institut fuer Informatik, ETH Zurich and also with the Department of Computer Science at the University of Illinois. Other people involved in the project are H. P. Frei, H. Burkhart, Chris Jacobi, B. Pattner, H. Sugaya, B. Weibel, and J. Weydert also of ETH. The project, begun in Fall, 1975, was intended to develop an interactive system that should serve as a self-explanatory school computer so that a user should be able to learn programming without further help. An extended version of Pascal-S was used both as an author language and as the programming language for teaching purposes. The hardware consisted of a PDP 11/03 with 28K words and dual floppys, 2 graphics terminals with TV monitors and 8080 micros with 8K bytes of RAM. The system software was written in MODULA. The 8080 was programmed in assembler.

Latest News About DOD-1 (ADA or DODO) - Andy Mickel

As we've told you in previous issues of Pascal News, the U. S. Department of Defense (DOD) has endeavored to procure a common programming language based on Pascal for all "embedded" computer applications--computer systems attached to weaponry. Reliable software should kill people reliably! A series of proposals were drawn up under the names Strawman, Woodenman, Tinman, Ironman, and now Steelman (June, 1978) which are alternatively titled "Department of Defense Requirements for High Order Computer Programming Languages." The DOD awarded 4 contracts to 4 software houses from those who had responded to the Ironman specifications in July, 1977. They formulated actual language designs in documents which are known by colors: BLUE-SofTech; GREEN-Honeywell Bull; RED-Intermetrics; and YELLOW-SRI International.

Basically, the designs consist of Pascal extended for concurrent prgcessed and time-dependent ("real-time") programming. Because a projected $\$3.0 \times 10^9$ will be spent each year by the DOD on software written in this language, the stakes are high. This fact alone has stimulated much manufacturer interest in Pascal over the last two years. We were always worried that this new language (formerly referred to as "DOD-1" and which has now been dubbed "ADA"--see Pascal In the News--or DODO) would swamp Pascal if it were too similar in form. Manufacturers then simply would not support Pascal but instead supply the new, extended language.

In February, 1978 the DOD narrowed the field to 2 by selecting GREEN and RED for actual implementation efforts. More than 50 groups of academic, military, and industrial people were hired to review and comment on the proposals. Niklaus Wirth and Tony Hoare consulted for YELLOW (the least ambitious of the proposals) and Henry Ledgard for GREEN. It is reassuring that none other than Edsgar Dijkstra wrote caustic comments which appeared in SIGPLAN Notices: EWD663 in July and EWD659-662 in October. ADA is safely going off the rails, and the threat to the integrity of Pascal is over, I believe. To quote Dijkstra:

> BLUE - "unacceptably complex"; GREEN - "the mixture between sense and nonsense remains baffling"; RED - "both advanced and backward in such an incongruous manner that I am baffled"; YELLOW - "an unsalvagable mess."

He stated in EWD663:

> "...It makes also quite clear why the new programming language cannot be expected to be an improvement over Pascal, on which the new language should be 'based'. (I am pretty sure that the new language--if it ever gets designed at all--will be much, much worse than Pascal if they proceed in this fashion.) You cannot improve a design like Pascal significantly by only shifting the 'centre of gravity' of the compromises embodied in it: such shifts never result in a significant improvement, in the particular case of Pascal it will be extra hard to achieve any improvement at all, as most of its compromises have been chosen very wisely..."

Please see Ed Reid's letter in the Open Forum section.

BOOKS AND ARTICLES

Please submit all notices of Pascal books, articles, abstracts, etc. to Rich Stevens at the address below:

KITT PEAK NATIONAL OBSERVATORY
Operated by The
ASSOCIATION OF UNIVERSITIES FOR RESEARCH IN ASTRONOMY, INC.
Under Contract With The
NATIONAL SCIENCE FOUNDATION

MEMBER INSTITUTIONS:
UNIVERSITY OF ARIZONA
CALIFORNIA INSTITUTE OF TECHNOLOGY
UNIVERSITY OF CALIFORNIA
UNIVERSITY OF CHICAGO
HARVARD UNIVERSITY
INDIANA UNIVERSITY
UNIVERSITY OF MICHIGAN
OHIO STATE UNIVERSITY
PRINCETON UNIVERSITY
UNIVERSITY OF TEXAS AT AUSTIN
UNIVERSITY OF WISCONSIN
YALE UNIVERSITY
UNIVERSITY OF HAWAII

950 North Cherry Avenue
P. O. Box 26732
Tucson, Arizona 85726
AC 602 327-5511
Cable Address:
AURACORP, Tucson

Tuesday evening, Nov. 21, 1978

Andy,

Here is the Books and Articles section for #13. Thank the world for self correcting typewriters. I promise to have things better organized so that my secretary can do the typing for #14.

After going through the previous Newsletters I decided to break the Books and Articles section into:

- Articles
- Books
- Book Reviews.

I did not include any abstracts with each article reference and only included a comment when I felt one was needed for clarification as to the papers relevance to PUG. This should cut down on the size of the section a little. I expanded the book section and gave as much information on the book as possible (table of contents when available) as this is the kind of stuff that I look at when initially inspecting a book.

I just received your UCC Computer User's Manual today and am initially very impressed (especially with the introduction to computing).. I'll send more detailed comments shortly.

ARTICLES

Amman, Urs, "Error Recovery in Recursive Descent Parsers", ETH Zurich, Berichte des Instituts fur Informatik, No. 25, May 1978.

Berry, R. E., "Experience with the Pascal P-Compiler", Software - Practice and Experience, Vol. 8, 617-627 (1978).

Burger, Wilhelm F., "Parser Generation for Micro-Computers", Dept. of Computer Sciences, U. of Texas at Austin, TR-77, March 1978.
(* A parser for the language Pascal can be accomodated in less than 4K of 8-bit bytes *)

Erkio, Hannu and Sajanienu, Jorma and Salava, Autti, "An Implementation of Pascal on the Burroughs B6700", Dept. of Computer Science, U. of Helsinki, Finland, Report A-1977-1.

Krouse, Tim, Electronic Design, Vols. 19 thru 23, 1978.
(* A continuing series of tutorials on Pascal *)

Lawrence, A. R. and Schofield, D., "SFS - A File System Supporting Pascal Files, Design and Implementation", National Physics Laboratory, NPL Report NAC 88, Feb. 1978.

LeBlanc, Richard J., "Extensions to Pascal for Separate Compilation", SIGPLAN Notices, Vol. 13, No. 9, Sept. 1978.

Lecarme, Olivier and Peyrolle-Thomas, Marie-Claude, "Self-compiling Compilers: An Appraisal of their Implementation and Portability", Software - Practice and Experience, Vol. 8, 149-170 (1978).
(* The study is centered around a specific case, the programming language Pascal and its many compilers *)

Marlin, Chris D., "A Model for Data Control in the Programming Language Pascal" Proceedings of the Australian Colleges of Advanced Education Computing Conference, Aug. 1977, A. K. Duncan (Ed.), pp. 293-306. Available from author at Dept. of Computing Science, U. of Adelaide, Adelaide, South Australia 5001.

Marlin, Chris D., "A Heap-based Implementation of the Programming Language Pascal,"Software - Practice and Experience, to appear. Also available from the author, see above.

Mohilner, Patricia J., "Prettyprinting Pascal Programs", SIGPLAN Notices, Vol. 13, No. 7, July 1978.

Neal, David and Wallentine, Virgil, "Experiences with the Portability of Concurrent Pascal", Software - Practice and Experience, Vol. 8, 341-353 (1978).

Posa, John G.,"Pascal Becomes Software Superstar", Electronics, Oct. 12, 1978.

Posa, John G., "Microcomputer Made for Pascal", Electronics, Oct. 12, 1978.

Pratt, Terrence W., "Control Computations and the Design of Loop Control Structures", IEEE Transactions on Software Engineering, Vol. SE-4, No. 2, Mar. 1978.
(* Examples drawn from a Pascal Compiler *)

Sale. A. H. J., "Strings and the Sequence Abstraction in Pascal", Dept. of Information Science, U. of Tasmania.

Sale, A. H. J., "Stylistics in Languages with Compound Statements", Australian Computer Journal, Vol. 10, No. 2, May 1978.

Shrivastava, S. K., "Sequential Pascal with Recovery Blocks", Software - Practice and Experience, Vol. 8, 177-185 (1978).

Tennent, R. D., "Another Look at Type Compatability in Pascal", Software - Practice and Experience, Vol. 8, 429-437 (1978).

BOOKS

PASCAL: An Introduction to Methodical Programming by Bill Findlay and David Watt (U. of Glascow, Computing Science Dept.). Computer Science Press, 306 pp.; UK Edition by Pitman International Text, 1978 (£4.95).

The book does not assume previous knowledge of computing, nor of advanced mathematics. Emphasis is placed on programming principles, good style and a methodical approach to program development. The technique of stepwise refinement is taught by consistent example throughout. In addition, two major chapters are exclusively devoted to programming methodology. The first is placed early enough to encourage good practice from the start. It includes sections on choosing refinements, testing and correcting and documentation. The second, at the end of the book, draws all the material together in two realistic case studies. Since the whole language is covered, the book may be of value to those who wish to learn something of the modern concepts of program structure and data structure, even if they must use a language other than Pascal. Contents:

Part 1: First Steps in Programming
Computers and programming; data and data types; the INTEGER type; the BOOLEAN type; Boolean algebra; input/output; control structures, WHILE and IF; methodical programming, Case Study 1.
Part 2: More Data Types
CHAR, enumerated types, subranges; REAL; arrays.
Part 3: More Control Structures
CASE, FOR, REPEAT, GOTO.
Part 4: Subprograms
Functions; procedures, parameter passing, procedures and program structure; advanced uses of procedures.
Part 5: More data structures
Records; strings; files; sets; pointers.
Part 6: Programming Methodology
Case studies 2 and 3, general principles.
Appendices
Collected syntax diagrams; reserved words and special symbols; predeclared entities; legible input and output; character sets.
Answers to selected exercises.
(* Author's information *)

Programming via Pascal by J. S. Rohl and Barrett (U. of Western Australia), Cambridge University Press, in press.
(* Anybody have any more information on this text ? *)

A Practical Introduction to Pascal by I. R. Wilson and A. M. Addyman, Springer-Verlag New York, 1978, 145 pp. ($7.90); MacMillan, London, 1978, (£3.50).

Suitable for beginners and experienced programmers who wish to learn the complete Pascal language, this concise introduction includes
- Syntax diagrams and complete examples illustrating each feature of the language;
- Simple problems introducing control constructs, expressions and the use of procedures;
- A discussion of the concept of data type, followed by a complete description of the data structure facilities of Pascal;
- An analysis of more advanced procedures and dynamic data structures;
- Over sixty programs.

Contents:
Introduction. The form of a program and basic calculations. Basic control constructs. Variables, constants and expressions. An introduction to input and output. An introduction to procedures and functions. Data types. An advanced data type - the sequential file. Elementary structured types 1, 2, 3 and 4: Set, array, record and variant. Advanced uses of procedures and functions. Dynamic data structures.
(* From publishers information *)
(* See below for review *)

The Design of Well-Structured and Correct Programs by S. Alagic and M. A. Arbib, Springer-Verlag New York, 1978, 292pp. ($12.80).
(* We are awaiting a review of this book from Duke Haiduk for next issue. *)

BOOK REVIEW

Programming in PASCAL by Peter Grogono
Addison-Wesley, Reading, Mass., 1978, 357 pp., $9.95.

Finally, an easy to read, lucid description of Pascal. This book is described in its preface as being suitable for an introductory programming course and in addition it should be an excellent self-study text for the experienced programmer who wants to learn Pascal.

The author made a point to cover the entire language and this is one of the book's strongest points. (One of the other texts on Pascal, A Primer on Pascal by Conway, Gries and Zimmerman does not cover the entire language, omitting sets, functions, pointers, records and files). Grogono also includes a good description of a specific implementation (the Zürich CDC system) and this will help one appreciate the implementation of the abstract language on a specific computer.

Another strong point of the book is that it is not just a text on writing programs in Pascal, rather it is a text on the Pascal language, intermediate data structures and structured programming. The inclusion of a chapter on program design and an appendix on program standards are a welcome addition to any language description, especially if the book is to be used for an introductory text. The data structures covered include linked lists and trees.

The examples used in the text are excellent and well thought out. Wirth's technique of stepwise refinement is used extensively. An interesting table processing program is provided to show that "Pascal, with a relatively small number of basic constructs, can nevertheless be used effectively to solve problems outside the domain of academic programming."

There are very few complaints that I have with this book. Each chapter is followed by a group of exercises (solutions are not provided) and some indication as to the relative difficulty of each exercise would be helpful. There are relatively few typographical errors.

All in all the book is excellent and a long awaited addition to the Pascal literature.

W. Richard Stevens

BOOK REVIEW

A Practical Introduction to Pascal by I. R. Wilson and A. M. Addyman
Springer-Verlag, New York, 1978, 145 pp., $7.90. ISBN 0-387-91136-7.

This book admirably fulfills the promise of its title - it gives a concise, well-organized tutorial on how to write programs in Pascal. The complete language is presented in fourteen short chapters. Particularly notable is the attention paid to the data structuring facilities of Pascal: fully six of the chapters deal directly with data structures.

After an introduction in Chapter 1, the basic structure of a complete Pascal program is shown in Chapter 2. Chapter 3 describes the control structures available in the language and gives advice on their use (including obligatory warnings about GOTO's). Chapters 4 and 5 discuss variables, constants, expressions, and input/output. Chapter 6, "An Introduction to Procedures and Functions", is especially good: the appearance at this point in the course of the presentation of these concepts is well-motivated and natural. Also, Pascal's parameter mechanism is explained nicely. Chapters 7 thru 12 discuss data types including files and record variants. Procedures and functions are revisited in Chapter 13 to show recursion and in Chapter 14 pointers are introduced in the context of "dynamic data structures". Each chapter is followed by suitable sets of exercises(easy) and problems (hard). There are four appendices: the completesyntax, delimiter words, answers to exercises and suggestions for solutions to the problems, and a note about the Pascal User's Group.

The pace of the presentation is even and well-motivated. New syntactic forms are introduced with simplified syntax charts andexamples and their semantics are conveyed by incisive programs or program fragments. Particularly useful for the new Pascal user is the printing of programs as they might actually be listed alongwith those ugly digraphs "(*" and "*)". Keywords are, however, printed in boldface. Example programs are developed in good style - stepwise refinement and top-down design are advocated and used.

The book is not above some minor criticism: some references are too broad "... readers are referred to Coleman (1978), Dahl et al. (1972) and Aho et al. (1974)." appears on page 69; refinement of program steps proceeds from comments expressed in Pascal comments later in the book (page 60, ff) but by lowercase fragments earlier (page 19, ff); there are a few misprints. Also, some of the exercises and example programs would be easier to understand if samples of their input and output were presented.

In summary, the book is a welcome addition to the Pascal literature. It is physically attractive and provides an excellent introduction to the language for beginning and experienced programmers alike.

R. Warren Johnson
Department of Mathematics and
Computer Science
St. Cloud State University
St. Cloud, Minnesota

CONFERENCES

We received recently, the latest Bulletin de Liaison du Sous-Groupe Pascal no.4 from Olivier Lecarme in France. He of course heads the French AFCET Pascal Group. This issue of the Bulletin was 125 pages long and is beginning to look like an issue of Pascal News! It contained an editorial, bibliography, list of Pascal implementations, and seven articles. Most interesting was the detailed commentary about the International Working Group on Pascal Extensions supplied by Olivier, and if we only had the time, it would be the quickest thing to do to translate and print in the Open Forum section. The contributions in the articles section are:

- Pointers: False Problems and Real Insufficiencies by M. Gauthier.
- A Graphic Extension for Pascal by N. & D. Thalmann.
- The "Mentor" System: A Pascal Programming Environment by P. Maurice.
- An Aspect of TSIMONE: A Version for Pascal Program Profiles by D. Renault.
- Where is the Standardization of Pascal? by O. Lecarme.
- A Comparison and Contrast between Concurrent Pascal and Modula by R. Rousseau.
- An Efficient Method of Controlling Type Unions by Nguyen Van Lu.

An ACM/SIGPLAN - Pascal User's Group sessions is being held at ACM '78 in Washington DC. See below.

The Australian Computer Science Conference will hold a workshop on Pascal. The conference is scheduled for February 1 and 2 in Hobart, Tasmania. Arthur Sale, of course is the host and is currently serving as vice-president of the Australian Computer Society. This is the second year for this conference. It was successfully launched under the name Australian Universities Computer Science Conference which was enthusiastically received last year.

Finally the University of California at San Diego (UCSD) Summer Workshop on Extensions was held this last July and has been reported on by Richard Cichelli below. I was promised, but did not receive, reports by Jeff Tobias, Arthur Sale and Ken Bowles. The major results of the Workshop were to get together a variety of computer manufacturers with some dyed-in-the-wool Pascalers. The Workshop rebuffed nearly all proposed extensions except those referred to the International Working Group on Pascal Extensions (such as otherwise for a case statement--see Open Forum under Standards). The members of the Workshop including the more than 15 manufacturers unanimously endorsed a motion to support the speedy adoption of the BSI/ISO Pascal Standard under development by Tony Addyman and his team...see Open Forum.

acm **SIGPLAN** SPECIAL INTEREST GROUP ON PROGRAMMING LANGUAGES

Association for Computing Machinery
1133 Avenue of the Americas
New York, NY 10036
(212) 265-6300

REPLY TO: Mail Stop 125A
NASA Langley Research Center
Hampton, VA 23665

July 24, 1978

Dear Andy,

An informal evening session devoted to PASCAL will be held at the 1978 ACM conference which will take place December 4-6, 1978, in Washington, D.C. The purpose of this session is to allow all conference attendees who are interested in PASCAL to get together and interact.

This is not a technical session in the usual sense. However, in order to convey the most information, it will consist, at least in part, of a series of short presentations (i.e., approximately 10 minutes) on PASCAL related topics. A presentation can address just about anything related to the language and its software; e.g., experience with PASCAL, tools for PASCAL programing, implementations, etc. Anybody who is planning to attend ACM '78 and who is interested in making a presentation should send a short description of what they will discuss by October 1 to:

John C. Knight
Mail Stop 125A
NASA Langley Research Center
Hampton, Virginia 23665

Presenters will be informed of their selection by November 1.

The purpose of requesting descriptions is not to perform any refereeing or technical judgment, but merely to allow a balanced program to be prepared for the limited time available.

Sincerely,

John C Knight

John C. Knight
SIGPLAN Representative
1978 ACM Conference Program Committee

THE UCSD PASCAL WORKSHOP

by

Richard J. Cichelli
ANPA/RI
Lehigh University

This is a personal report of my experiences with the UCSD Pascal Workshop held by Dr. Kenneth Bowles at the University of California at San Diego during July of 1978. I will discuss my own role at the workshop, and in no way should this report be considered a report from the workshop participants as a whole.

In May of this year, I received a letter from Ken Bowles inviting me to attend his planned workshop, the purpose of which was to "standardize extensions to Pascal". Ken and I had spoken about his efforts in putting Pascal on small machines previously at the ACM 1977 Conference. After seeing the UCSD Pascal system in action, I was convinced that it was excellent technology and held great promise for both educational and commercial applications. I reviewed Ken's book, Microcomputer Problem-Solving in Pascal, for PN #11 and sent a pre-publications copy of that review to Byte Magazine. Upon receiving this review, Carl Helmers began his own interaction with Ken concerning the UCSD system.

At ACM'77 Ken talked to me about the language changes that he felt small systems required. He spoke of the lack of viable Standards activities within the Pascal Users Group and his willingness to organize a Standards workshop. I suggested to Ken at that time that Standards were something that the Users Group would soon be more involved with and that his help on a PUG Standards Committee would be welcome. I was truly surprised and chagrined to hear of Ken's organizing his own Pascal Workshop. I have never felt that the precision of expression and depth of understanding necessary for Standard-related activity was the type of thing done well by implementation-oriented individuals. Even more important was the fact that effective international Standards activities had already been initiated in PUG under the direction of Tony Addyman and an Extensions Working Group has been formed chaired by Steengaard-Madsen. Dr. Wirth was helping this group with their activities.

On June 1, I sent a letter to Ken expressing my concern about the UCSD project. The text of the letter is as follows:

> "I have given careful thought to your invitation to participate in the UCSD workshop. ANPA was a member of 27 associations that participated in the acceptance of the 1966 ANSI-FORTRAN Standard. We consider our endorsement of programming language standards of great importance to our 1200 newspaper members and are sure that Pascal will have a major impact on future newspaper computer systems. Unfortunately, no matter how well meaning your efforts towards standardized extensions are, we believe the appropriate review and evaluation activities should lie wholly within the Pascal Users Group. We would welcome your initiative in being part of a PUG Standards Committee but neither ANPA nor I will support or endorse any self-proclaimed UCSD Pascal modification adventure."

Copies of the letter were forwarded to the Standards Committee, the Working Group, and Andy Mickel. My primary concern with the UCSD effort was that any extensions agreed to by the UCSD Group would become a defacto Standard and "enhanced Pascal" would go into competition with Standard Pascal. I very much felt that most of the UCSD deviations from the Standard were simply inappropriate. I was sure that most, if not all, of the UCSD language modifications would be rejected both by the Working Group and the Standards Committee. I firmly believe that the UCSD interactive systems feature good engineering. However, like most new implementations derived from the Zurich produced P4 system, the UCSD Pascal fails to implement important parts of the Standard and has extra goodies implemented in ways inconsistent with either the Standard, or worse, the recommended extension technique.

Upon returning from a business trip, I found that Ken had placed an urgent call to me. I returned his call and spent 2 hours talking with him. Ken was very concerned about Andy Mickel's reaction to the UCSD project. Andy and I shared similar reservations. During the conversation, Ken invited me to attend the workshop as a PUG representative instead of as a member of a contributing organization. I said that I would give consideration to this idea.

During the month of June I had many conversations with Andy and other potential workshop attendees. Upon receiving a document titled "Checklist of topics for the UCSD Workshop on Pascal Extensions" that consisted of more than 75 items, I was even more concerned. In my opinion, adoption of changes proposed in this checklist would effectively rape the Pascal Standard. My primary hope at that point was that no one would want to go to the UCSD workshop. Bob Dietrich of Tektronix made a number of telephone calls to me indicating first, that Tektronix was interested in participating in such a workshop and second, that he felt as I did about most of the checklist items. He assured me that many other potential workshop participants felt as I did about the Standard and about these UCSD extensions. He felt that if the issues were properly dealt with, it was likely that the consensus of the workshop would be to reject almost all of the proposed extensions. In later conversations with Ken, he himself also assured me that rejection of ill-conceived extensions would be an important activity of the workshop. By this time it was clear that there would be a number of participants in the workshop and that it was important that those workshop participants who were responsible for corporate implementations of Pascal hear arguments in favor of adhering to the Standard. Andy informed me that because of prior committments he could not attend the workshop to represent the Users Group. He asked me to do so in his stead. Also charged with a similar mission were Jim Miner, Arthur Sale, and Bob Johnson. Since Andy, Bob, and I founded the Pascal Users Group, we hoped to be able to effectively represent membership as a whole at UCSD.

At this point I accepted Ken's invitation to attend the workshop. I also agreed to attend a pre-workshop meeting of like-minded individuals that was the brainchild of Bob Dietrich. Of particular help in formulating a "pro-standard" position was the extensive work done on the checklist by Mike Ball.

The week at the conference was one of the most interesting and challenging of my computer science career. By the Sunday meeting we had all found out where the UCSD group stood as far as the extensions were concerned. It seems they had already decided to endorse most of these ill-conceived ideas by actually implementing them within the UCSD software system. Shortly after Ken's initial address to the more than 50 participants of the workshop, a number of participants suggested that the overall goals for the workshop be clearly laid out before specific consideration of the checklist items began. A number of views in addition to Ken's were presented on this topic and I was asked to speak on this "as a representative of the Pascal Users Group". Most of the workshop participants were chosen by Ken because they were members of Pascal implementation teams at various large companies. These people are used to identifying problems and developing solutions. I am sure they did not welcome hearing from me that I believed they should act only in an advisory capacity and defer final evaluation to a Standards Committee within PUG. I am sorry that I don't have a complete transcript of the extemporaneous talk I gave addressing this issue, but the most important point that I tried to make was that ad-hoc solutions to perceived problems with Pascal were to be preferred to hastily conceived and implemented changes to the language Standard. I assured the group that if they chose to take a united stand favoring an array of extensions, the changed language would be a competitor to Standard Pascal, much to the detriment of the user community.

One of the problems in giving this talk was that Ken asked me to present the issues not in the general framework that I just outlined but instead as an item by item review of "how do you do 'x'?" (where 'x' might be direct access files, overlays, complex numbers, strings, etc.). My general statements included a suggestion that the only types of extensions that should be considered at all are those which 1) are consistent with the design goals of Pascal, and 2) add a facility not implementable in Standard Pascal. For example, in talking about segments and overlays I suggested that such concepts had nothing to do with the problem solved by an algorithm but only with how a compiler translated the algorithm expressed as a program into executable code for a particular operating system. I suggested that if it was necessary for the compiler to know about overlays, then this information should be incorporated in compiler directive comments. (Pascal-6000 needs no such compiler directives for overlayed programs.) I suggested, addressing the issue of complex numbers, that they are easily created within the standard mechanisms of the language. I also noted that direct access files are being considered by the European Working Group. I also mentioned that at Lehigh University we have used direct access files extensively and do so by calling external library routines. Since, at Lehigh, more than four different systems of direct access file support are utilized by Pascal programmers, I suggested that reasonable men would differ as to what constituted a good set of primitive functions for accessing such datasets. I suggested that where adequate ad-hoc solutions exist and no consensus about them exists, no Standard should be imposed. By not creating a Standard for such an item, experimentation is encouraged. From this experimentation better solutions can be derived.

During the next three days we broke into subcommittees to consider checklist items one by one. It was Ken's idea that subcommittee sessions would be recorded and "where consensus was reached on an item a consensus position would be prepared". Each subcommittee had one or more UCSD students or faculty members on it to help in recording and transcribing the group's deliberation. A few of these individuals acted as monitors on their subcommittees.

I worked with what was called the Expressions Group. Our approach was more formalized than some of the other groups. In addition to myself, the members of the subdommittee were Terry Miller (moderator), Steven Dum of Tektronix (recording secretary), Ruth Richart of Burroughs, Skip Stritter of MOTOROLA, and Don Baccus of OMSI. We began by considering each of our 15 topics one by one. For each topic we first stated exactly what our recommendations were and then we presented our reasoning that went into the recommendation. For example: Item 3.2 on our list was -- "provide for short circuit AND and OR." The text of our recommendation is

3.2 We recommend that AND and OR should be left as defined. I.e. the implementor may choose short circuit or complete evaluation, user beware!

Short circuit AND and OR (CAND and COR) can be programmed around in existing Pascal. They are a minor extension. The majority of the group felt that the cost of implementation (size, introducing features, etc.) does not justify the benefit.

We firmly reject the concept of introducing complete evaluation operators such as LAND or LOR.

On item 3.4 -- "provide for exponentiation" -- we made the following recommendations:

3.4 We recommend rejection of exponentiation as an infix operator or standard function.

It is possible to provide a predefined function POWER or to write it as a Pascal function with the parameters defined as

```
FUNCTION POWER (A,B:REAL):REAL;
```

We felt that it was not necessary to add a function to raise an integer to an integer power as most usage of exponentiation seems to be satisfied with the real form.

I have the highest regard for the people that Ken recruited to participate in his workshop. Ruth Richart, for example, is a principal implementor of a new systems language that Burroughs is using. This language is modeled after Pascal. On item 3.2 (the short circuit AND and OR), she pointed out that on Burroughs machines the short circuit evaluation is significantly less efficient than full evaluation in most cases. Burroughs machines are exceedingly efficient on stack operations (and thus super expression evaluators) and not nearly so efficient on conditional branches. We concluded that it was important that the implementor of a Pascal compiler be given the freedom to choose the optimal evaluation technique.

As we worked on each item, we followed Wirth's suggested procedure for considering extensions. First, we introduced the extension in a tutorial fashion to the subgroup. Then we showed how the extension would be used in practical programming. Then we discussed its relationship to the language standard and its implementation consequences. It was interesting to note that in the AND/OR controversy the UCSD supplied example program was clearly not of the best design. After exploring the issue for some time, it became clear that short circuit evaluation was most often used in an attempt to sneak past undefined conditions. This led actually to a suboptimal or less clear presentation of the algorithm. It was this kind of discussion that gave us confidence in our recommendations.

At the conclusion of the day's meeting, Steve Dum took our carefully worded notes and typed them into one of the UCSD Terak systems. A little quick editing and we had line printer copy of the day's discussion. The next day we made multiple copies of our preliminary statements on the 15 items. All members of our subgroup were chagrined that the other groups did not have written statements of their recommendations. At the conclusion of the general meeting on Tuesday, all subgroups were directed to go back and produce concise, well-worded descriptions of their recommendations and deliberations. And the Expression Group was asked to consider seven more items. Meanwhile Bob, Arthur, Jim and I were meeting before the workshop sessions began and after the workshops ended each evening. We were attempting to formulate an appropriate policy statement for PUG which would guarantee that what constituted Pascal was defined by the PUG membership. In this effort we sorely missed having Andy with us to help formulate policy.

By Thursday the work of the subgroups neared completion. Also a number of individuals in the workshop indicated interest in working thru PUG on implementations and standards. The following position paper was developed by the PUG representatives and Andy was consulted and asked to have his name included on the paper:

PUG Working Position

(1) In October PUG will publish a proposed constitution. Upon acceptance of the constitution by the PUG membership, election of officers will take place. It is hoped that by January 1, 1979, a formal governing structure for PUG will be established.

(2) A draft of the ISO Pascal Standard will be published by the end of 1978 for member reaction.

(3) An implementation subgroup will be formed to coordinate the enhancement and distribution of portable compilers and to facilitate correspondence among implementors. A new section of Pascal News will inform the membership of these activities.

(4) A standards subgroup will be formed. It will distribute (for a reasonable fee) a Validation Suite. An incomplete version of the Suite constructed under the direction of Brian Wichmann (developer of the Algol 60 Validation Suite) and distributed by R. Cichelli will be available during September, 1978.

(5) Actual proposals from the International Working Group will appear in Pascal News. The first will be in October, 1978.

PUG aid to the UCSD Workshop

(1) Pascal News will publish (subject to length constraints) a report of the UCSD Workshop and will help to distribute the full Workshop report.

(2) Pascal News will publish a new section on solutions of and commentary about significant programming problems which may be outside the scope of the Pascal Standard.

Andy Mickel
Richard J. Cichelli
Robert Warren Johnson
James F. Miner
Arthur H. J. Sale
July 13, 1978

Jim Miner presented the PUG working position paper to the workshop and it was greeted with applause.

It is my opinion that the result of the first week of the UCSD workshop was to strengthen the Pascal Standard and to reaffirm the pre-emminence of PUG with regards to Pascal. One of the most important factors of that week was the acceptance by all workshop participants of the following "agreement in principle":

> At the time the workshop convened, two major activities with respect to the definition of the language Pascal were already underway. The International Standards Organization had begun working on a complete definition of the Pascal language in light of the shortcomings of the Jensen and Wirth document. A Working Group focused around Steensgaard-Madsen had begun working on extensions to the Pascal Language aimed at correcting a few well-known deficiencies in the language. In light of these activities the workshop assumed as its primary goal to address well-defined, consistent, application-oriented extension sets and agreed to pass to the other two bodies such recommendations and information deemed appropriate to their work.
>
> The workshop recognized the existence of possible modifications to the Pascal Language which, due to the impact throughout the language, would de-facto create a new language and decided not to act on these modifications at this time.
>
> In order to achieve the purposes stated above the workshop has resolved to:
>
> I. Publish and distribute the Proceedings of the workshop. In particular the Proceedings will be forwarded to ISO, the Pascal Users Group, and the Steensgaard-Madsen Working Group.
>
> II. Organize a structure which will permit the orderly continuation of the work begun at the meeting in San Diego.
>
> III. Provide a mechanism to reinforce the importance of Standard Pascal by agreeing that all compilers purporting to support the programming language Pascal should include a variant of the following statement in the source code and all documentation:
>
> "The language --(1)-- supported by this compiler contains the language Pascal, as defined in --(2)--, as a subset with the following exceptions:
>
> (a) features not implemented
>
> --(3)--   -- refer to page --
> --------
>
> (b) features implemented which deviate from the Standard format
>
> -----   --
> -----   --
>
> Notes:
>
> (1) insert the name of the dialect
>
> (2) insert "the Jensen and Wirth User Manual and Report" or "the ISO draft standard" or "ANSI Pascal standard" as appropriate
>
> (3) A brief statement plus reference to more detailed information will suffice. The list should be as complete as possible.

Review of Pascal News 9/10, 11, and 12

Backissues of Pascal News 9/10, 11, and 12 are still available, and will be for the forseeable future. Therefore I would like to urge all new members to consider obtaining them so that you will be better oriented to events in our recent past. Issues 1-8 are unfortunately out of print. 1-4 are described in detail in issue 6; 5-8 are described in detail in issue 11; 1-8 are briefly described in issue 9/10.

If you want to know generally what is important, then issue 9/10 contains the base roster for PUG, and a complete survey of Implementations. It also contains the last bibliography and list of textbooks to date. Issue 11 contains the worst collection of wild proposals to extend Pascal, and the terrific article on type compatibility by Pierre Desjardins. An errata to old printings of Pascal User Manual and Report is in #11. Issue 12 contains our first applications sections with two important software tools: COMPARE and a pair of programs for Performance Measurement.

All three issues contain important information about Pascal standards.

Pascal News #9/10 (combined issue), September, 1977, Pascal User's Group, University of Minnesota Computer Center, 220 pages (114 numbered pages), edited by Andy Mickel.

Editor's Contribution: Pascal jobs, a list of computer companies using Pascal, Pascal on personal computers, current information on the status of PUG and Pascal News: printing error in #8, Australasian distribution center, change in the name of Pascal Newsletter to Pascal News, new policies, back issues, growth in membership, and PUG finances.

Here and There: Tidbits (9 pages), reports from German and French Pascal conferences, Books and Articles classified by applications, languages, textbooks, and implementations; Bibliography of 68 entries; past issues of Pascal Newsletter (1-8); PUG finances for 1976-1977; Roster 77/09/09.

Articles:

"Pascal at Sydney University"
-Tony Gerber and Carroll Morgan
[A description of implemented (proven) extensions and changes to the CDC-6000 Pascal compiler in use at Uni of Sydney. These include operating system interface, ability to read strings, read and write user-defined scalar types, case statement extensions, and two machine-dependent extensions. The conclusion states that these changes to the compiler have not detracted from the overall efficiency of the compiler, and that 2-year's use has vindicated the inclusion of these extensions.]

"Disposing of Dispose"
-Stephen Wagstaff
[An argument for an automatic garbage collection system for dynamic variables in Pascal is made, thus obviating the need for, and the risks associated with, user-controlled de-allocation (e.g. DISPOSE). Complete protection from "dangling" pointers may be obtained.]

"What is a Textfile?"
Bill Price
[The definition of the pre-defined type Text in Pascal as File of char is in error and because of this lapse, complex special-case notions are introduced as primitive concepts. A new, more useful understanding and definition of the textfile notion is proposed.]

"Generic Routines and Variable Types in Pascal"
-B. Austermuehl and H.-J. Hoffmann
[Generic routines and variable types, as introduced in EL1 are a means to postpone the binding time of routines and data. An examination is given of what degree such features may be carried over to Pascal without severe violation of the static type checking requirement. The conclusion is made that generic routines fit into Pascal, while variable types have to subject to strong restrictions. Variable types may only be used in connection with a special syntactic form.]

Open Forum:

77/05/10 Arthur Sale to Andy Mickel: [Australasian distribution Centre, CDC-bias: files, program heading, Burroughs 6700 implementation on 7700, 6800, etc.]
77/05/24 Tony Gerber to Andy Mickel: [PUGN distribution to Australia, why haven't you printed our paper, Pascal not Utopia 84, extensions to Pascal 6000.]
77/06/01 Richard Cichelli to Andy Mickel: [Each issue of PUGN better, software tools, an applications section in PUGN.]
77/06/16 Mike Ball to Andy Mickel: [Interdata 8/32 Pascal, Univac 1100 Pascal, proposed extensions to standard Pascal, proposed standard for editing format and distribution of Pascal software tools and programs.]
77/06/16 Peter Grogono to Andy Mickel: [standardizing Pascal--preserve its simplicity, change to Read procedure for error recovery, especially for interactive programs.]
77/06/24 Wally Wedel to Andy Mickel: [CDC-6000 and DEC-10 Pascal at the Univ. of Texas, standards via X3 and experience from X3J committee.]
77/07/22 George Richmond to Andy Mickel: [Keep up the good work, support for preserving standard Pascal. Distribution at Colorado is now running smoothly.]
77/07/28 Neil Barta to Andy Mickel: [Pascal jobs available at ADP Network services, using Nagel's DEC-10 Pascal compiler.]
77/07/29 Stephen Soule to Andy Mickel: [Pascal competing with FORTRAN: variable-initialization, own variables, flexible array parameters, textfiles and variant-records in formatting.]

Special Topic: Micro/Personal Computers and Pascal
77/07/08 David Mundie to Andy Mickel: [Zilog rumor about Pascal machine, letters to personal computer journals, game programs in Pascal, like variant records.]
77/06/27 Larry Press to Andy Mickel: [Would like to publish work from PUG members in SCCS Interface to counter BASIC proliferation.]
77/09/01 Maria Lindsay to Andy Mickel: [Microcomputer library and resource center in Madison Wisconsin very interested in Pascal materials.]
77/08/24 Jim Merritt to Andy Mickel: [disagree about pressing supposed advantage on micro computers. UCSD Pascal project may hold future hope, UNIX Pascal information.]
77/09/06 Carl Helmers to Andy Mickel: [Will write editorial in the December BYTE for Pascal. Pascal an excellent choice to succeed BASIC]

Special Topic: Standards
Introduction
77/08/09 D. G. Burnett-Hall to Andy Mickel: Another Attention List.

Implementation Notes: Checklist, General Information, Software Tools, Portable Pascals: Pascal-P, Pascal Trunk, Pascal J; Pascal Variants: Pascal S, Concurrent Pascal, Modula; Feature Implementation Notes: Set of Char, the For statement, Else in case, var parameters, Interactive I/O; Machine-Dependent Implementations: Amdahl 470, B1700, B3700/4700, B5700, B6700-7700, CDC Cyber 18 & 2550, CDC3200, CDC3300, CDC3600, CDC 6000/Cyber 70,170, CDC7600/Cyber 76, CDC Omega 480, CDC Star-100, CII Iris 50, CII 10070, Iris 80, Computer Automation LSI-2, Cray-1, Data General Eclipse/Nova, DEC PDP-8, PDP-11, DEC-10/20, Dietz Mincal 621, Foxboro Fox 1, Fujitsu Facom 230, Harris/4, Heathkit H-11, Hewlett-Packard 21MX, 2100, 3000, Hitachi Hitac 8000, Honeywell H316, Level 66, IBM Series 1, 360/370, 1130, ICL 1900, ICL 2900, Intel 8080, 8080a, Interdata 7/16, Interdata 7/32,8/32, ITEL AS/4, AS/5, Kardios Duo 70, Mitsubishi Melcom 7700, MITS Altair 680b, MOS Technology 6502, Motorola 6800, Nanodata QM-1, NCR Century 200, Norsk Data Nord 10, Prime P-400, SEMS T1600, Solar 16/05/40/65, Siemens 330, Siemens 4004, 7000, Telefunken TR-440, Terak 8510, TI-ASC, TI 9900, Univac 90, Univac 1100, Univac V-70, Xerox Sigma 6, 9, Xerox Sigma 7, Zilog Z-80.

Pascal News #11, February, 1978, Pascal User's Group, University of Minnesota Computer Center, 202 pages (106 numbered pages), edited by Andy Mickel.

Editor's Contribution: Addenda on list of companies using Pascal. Itemization of costs from PUG(UK) distribution center.

Here and There: Pascal jobs, Help wanted for numerical library project, Tidbits (7 pages), Evolution of PUG dog, Pascal in the News, DOD-1 report, reports from German ACM Pascal meeting and ACM '77 Pascal session in Seattle, Books and Articles including Applications, Implementations,Languages, and Textbooks; Concurrent Pascal literature, documents obtainable from the University of Colorado Pascal distribution center. Errata to Pascal User Manual and Report Second edition. Detailed review of Pascal Newsletters 5, 6, 7, and 8. Roster Increment (77/12/31).

Articles:

"Type Compatibility Checking in Pascal Compilers"
Pierre Desjardins
[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 consequances.]

"A Novel Approach to Compiler Design"
James Q. Arnold
[A sarcastic appraisal of the Honeywell Level 66 compiler implemented by the University of Waterloo. Its poor realization is examined with respect to program portability, program correctness, and user interface.]

"Status of UCSD Pascal Project"
Kenneth L. Bowles
[A description is given of the project which developed the LSI-11 Pascal implementation at UCSD. The project was motivated by teaching interests at the university and has evolved into research and development interests centering on microprocessors. Descriptions follow of the Pascal-based software system, minimum configuration, 8080 and Z-80 versions, Pascal extensions and alterations, Introductory Pascal course and textbook, a "Tele-mail" user support facility, and forthcoming improvements.]

"Suggestions for Pascal Implementations"
Willett Kempton
[A user's point of view is presented on features encountered in 3 Pascal implementations. Conditional debugging code, a better cross-reference, flagging non-standard constructs, implementation of UNPACK, PACK, and LINELIMIT, conversation compilation, error-recovery and formatting of input, interactive I/O, padding strings with blanks, and more predefined constants like MAXINT are examined.]

"Suggested Extensions to Pascal"
Robert A. Fraley
[A number of extensions and modifications to Pascal are suggested. 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 hopefully preserve its clarity and simplicity. Some of them are optionally available in the UBC/IBM 370 Pascal compiler.]

"Adapting Pascal for the PDP 11/45"
D. D. Miller
[A description and adaptation is given 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.]

"Pascal: Standards and Extensions"
Chris Bishop
[Comments are given on the current standards/extensions argument, and to suggest some specific modifications to the standard and some useful extensions. These include: array parameters, standard type char, otherwise in case, no formatted read, repeat and case statement changes, inverse to ord, different treatment of file variables, and I/O and textfiles, addition of exponentiation operator.]

Open Forum:

77/11/09 Helmut Weber to Andy Mickel: [CDC-6000 Pascal inquiries]
77/10/28 Barbara Kidman to Andy Mickel: [Pascal teaching at the University of Adelaide.]
77/11/03 Tom Kelly to Andy Mickel: [Burroughs Pascal from UCSD now running at Burroughs.]
77/10/12 Tony Schaeffer to Andy Mickel: [Interactive I/O, language standards in the light of the natural evolution of Latin and ANS Fortran.]
77/08/25 Robert A. Fraley to Andy Mickel: [Comments on changing the definition of Pascal and his submitted paper also appearing in the issue.]
77/11/07 Robert A. Fraley to Arthur Sale: [Comments on the Feature Implementation Note concerning else in case, sets of char; support for ASCII as a Pascal standard!]
77/12/26 Barry Smith to Andy Mickel: [Oregan Minicomputer Software history with ESI and OMSI, and their PDP-11 Pascal implementation; Pascal T-shirts.]
77/12/12 Dave Thomas to Andy Mickel: [Pascal at Imperial College, London. A multi-user reentrant STARTREK program exists in Pascal for the IBM 370 implementation at IC.]
77/11/07 Mitchell R. Joelson to Andy Mickel: [Law Enforcement Assistance Administration regulations vis-a-vis programming alnguages for use in criminal justice information systems.]
77/12/30 Ken Robinson to Andy Mickel: [Australasian distribution; Pascal use in Australian Universities, sundry comments on Pascal]

Special Topic: Pascal Standards:
Introduction by Andy Mickel and Jim Miner: [ISO Standard Pascal, Conventionalized Extensions, Laundry Lists of Additional Features, Pascal Compatibility Report.]
77/12/09 Bengt Nordstrom to Andy Mickel: [The Swedish Technical Committee on Pascal; Yet Another Attention List, will be in touch with the British Standards group.]
77/12/30 Ken Bowles to Andy Mickel: [Standardized Pascal Extensions, proposal for Pascal Workshop with representation from industrial firms, governmental agencies, and "academic experts". Consideration of DOD-1, a proposed Pascal-X extended version]

Implementation Notes: General Information; Applications; Portable Pascals, Pascal-P4 Bug Reports and how Pascal-P4 relates to the standard. Pascal Variants: Pascal-S, Concurrent Pascal, Modula. Feature Implementation Notes: Unimplementable Features - Warning; Compiling Boolean Expressions -- The Case for a "Boolean Operator" Interpretation; Long Identifiers; Interim Report -- Implementation of For Statement 2, More on For Statement. Machine-Dependent Implementations: Alpha Micro AM-100, Andromeda Systems 11/B, Burroughs B5700, B6700/7700, CDC Cyber 18 and 2550, CDC3200, CDC 6000/Cyber 70,170, CDC 7600, Data General Nova/Eclipse, DEC PDP-8, DEC PDP-11, DECUS Pascal SIG; DEC LSI-11, DEC 10, HP-21MX, Honeywell 6000, level 66, H316, IBM 360/370, ICL Clearing House, ICL 1900, ICL 2900, Intel 8080, MITS Altair 8800, Motorola 6800, Prime P-300, Univac 1100, Zilog Z-80.

Pascal News #12, June, 1978, Pascal User's Group, University of Minnesota Computer Center 135 pages (70 numbered pages), edited by Andy Mickel.

Editor's Contribution: Personal Observations regarding Pascal-P, the first good critical article about Pascal, the need for a "business-oriented" Pascal procedure library, and more news needed about teaching experiences about Pascal. Status of Pascal User's Group: must raise rates for US and UK; rates lowered for Australasia.

Here and There With Pascal: Pascal Jobs, Tidbits (7 pages), French/English, English/French Pascal Identifiers, Pascal in the News, Conferences, Books and Articles: Applications, Implementations, Languages, Textbooks, Reviews, Articles wanted. Roster Increment (78/04/22).

Applications (new section): News: Empirical study of Pascal programs (Pascal program style analyzer),numerical library project. Algorithms: A-1 Random Number Generator A-2: Timelog; Software Tools: S-1 Compare (compare two textfiles for equality), S-2-1 Augment, S-2-2 Analyze (Pascal performance measurement programs); Programs: P-1 Printme (reproduce self).

Articles:

"Extensions to Pascal for Separate Compilation"
Richard J. LeBlanc
[The lack of features in Pascal to allow procedures and functions to be compiled separately can be of considerable inconvenience in the development of large programs. This weakness is particularly evident when modifications are being made only to limited parts of a program. Modificaitons of this sort are common, for example, in the maintenance or extension of a Pascal compiler.. By creating a global environment, separate compilation of routines using that environment, and additions to the environment without requiring recompilation of existing routines and declarations--all via extensions--a useful mechanism can be attained.]

"What Are Pascal's Design Goals?"
Robert Vavra
[As a long-time reader of Pascal News, the author has enjoyed the many articles in which people have discussed various features which could be added to Pascal, but they have been unable to take seriously. In arguing for or against some particular feature, writers have rarely involked Pascal's design goals in support of their arguments. Such failure to build a proper foundation for one's arguments might be acceptable in casual conversation, but not in a serious discussion.]

"Pascal Environment Interface"
Terje Noodt
[Work is presented for a Pascal implementation for the Norsk Data Nord 10, running interactively. The Pascal Report does not say too much about how to interface a compiler to a computer system and its users. To further complicate matters, what it does say about this relates to a batch system, and is worthless or unusuable in an interactive system. A language is often judged on the way a particular implementation interfaces to its environment such as what tools are available for the construction, compilation, and execution of a program, and what interfaces are like between the implementation and other systems on the computer (particularly the operating system. The conclusion is to think ecologically, and do not let the environment pollute Pascal!]

"Subranges and Conditional Loops"
Judy M. Bishop
[The subrange facility in Pascal is an aid to run-time security for fixed-boundary constructs such as counting (for) loops and array subscripts. The relevant types can be precisely and naturally defined, and the compiler can minimise the amount of run-time checking required. However, an index which increases under program control, as in a conditional (while) loop, presents a problem. This note discusses the problem and presents a solution in terms of a naming convention.]

"A Few Proposed Deletions"
John Nagle
[Since quite a number of extensions to Pascal have been proposed, I thought that it would be desirable to propose a few deletions to keep the size of the language down. With the goal in mind of keeping Pascal a simple, elegant, and useful language requiring a minimum of run-time machinery, I propose a few simple changes in the direction of simplicity.]

Open Forum:

78/01/18 Arthur Sale to Andy Mickel: [Pascal News distribution in Australasia; explanation of large size of PUG(AUS) fee. Pascal as a first language in Australian uni's.]
77/11/11 Giuseppe Selva to Andy Mickel: [Comments on the increasing use of Pascal; need for reading and writing scalars, varying length strings, formatted input, etc.]
78/02/02 Jerry Pournelle to Andy Mickel: [Acquiring Pascal for a 48K Z-80.]
78/02/24 Joe Celko to Andy Mickel: [Comments to proposed extensions by Robert Fraley in last issue: doesn't miss common or modules; compiling included files nice.]
78/02/23 Hellmut Weber to Andy Mickel: [Wish list from a user's point of view for Pascal-6000 Release 3 from Minnesota.]
78/02/24 Arthur Sale to PUG membership: [Commentary on Pascal News No. 11; David Barron's proposal for algorithms excellent, Pascal is not up for grabs, PUGN maturing.]
78/02/27 Greg Wetzel to Andy Mickel: [Shame on you for including Fraley's article--it scared us--congratulations, you were terrifyingly successful! Stand by your guns.]
78/03/06 Eric Small to Andy Mickel: [Looking for Pascal programmer for consultants in broadcasting technology.]
78/03/08 Bob Jardine to Andy Mickel: [Reply to criticism of B6700 by Arthur Sale's Feature Implementation Note on Unimplementable Features.]
78/03/10 K. S. Bhaskar to Andy Mickel: [Pascal needs standardization and perhaps an extension mechanism like ALGOL 68.]
78/03/15 Terje Noodt to Andy Mickel: [A new implementation forthe Nord 10; the system interface is an important consideration.]
78/03/16 Don Terwilliger to Andy Mickel: [Even though Tektronix is actively using Pascal it does not currently have products incorporating Pascal programming capabilities.]
78/03/16 Edward Reid to Andy Mickel: [Interested in Arthur Sale's comments about Pascal on the B6700; comments on other items in past Pascal News issues.]
78/01/02 Werner Remmele to Andy Mickel: [Pascal implementation on the Intel 8080 using the ISIS II operating system. Notes about the project.]
78/03/15 Mark Horton to Andy Mickel: [Pascal at the Univ. of Wisconsin; comments about proposed extensions to Pascal, some more proposed extensions to Pascal]
78/04/11 Jon Squire to Andy Mickel: [Pascal and DOD-1; need for a standard set of acceptance test programs for Pascal.]
78/04/07 Judy Bishop to John Strait: [A further comment on predefined types and subranges used in conditional loops.]

Pascal Standards:

Introduction by Andy Mickel and Jim Miner: International Working Group by Jørgen Steensgaard-Madsen investigating conventionalized extensions at last; News from Tony Addyman on the BSI/ISO Pascal Standard; criticism of the upcoming UCSD Workshop on Pascal Extensions.
78/04/07 Niklaus Wirth to Andy Mickel: [Definition of Pascal syntax using Extended Backus Naur Form on only 2 pages.]
78/02/06 Tony Addyman to Andy Mickel: [New phone number, urge that all PUG members comment on the BSI/ISO draft standard document.]
78/02/01 Tony Addyman to DPS/13/4, Swedish Technical Committee and all correspondents: [Update on progress by the BSI working group DPS/13/4 for a Pascal standards document]
78/03/23 Charles Fischer to Andy Mickel: [Criticism of structure and format of Ken Bowles's proposed summer Workshop at UCSD.]
78/04/10 Richard LeBlanc to Andy Mickel: [Reservations expressed about the structure and format of Ken Bowles's summer Workshop on Pascal extensions.]
78/03/30 Bob Vavra to Andy Mickel: [Comment on Pascal's Design Goals; optimistic about the future of Pascal in spite of all the moves to extend Pascal.]

Implementation Notes: Checklist (new item); Portable Pascals (more Pascal-P4 bug reports).
Feature Implementation Notes: Representation of Sets; Machine-Dependent Implementations: B6700/7700, B4700, B1700, CII 10070, IRIS 80, Commodore 6502, Computer Automation LSI-2,4, Data General Eclipse, DEC PDP-11, VAX 11/780, HP-2100,21MX, HP-3000, IBM 360/370, Intel 8080, Interdata 7/16, 8/32, Northwest 85/P, Prime P-400, Index to Implementations for issues 9-12

ROSTER INCREMENT (78/10/31)

Following is a list of PUG members who either joined or changed address since the last roster increment was printed on 78/04/22. The list actually includes some persons who renewed, but whose address didn't change. Sorry.

```
01002   WILLIAM D. TORCASO/ HAMPSHIRE COLLEGE/ BOX 548/ AMHERST MA 010 02/ (413) 549-4600
01003   TERRY E. WEYMOUTH/ DEPT OF COMP & INFO SCI/ UNIV. OF MASSACHUS ETTS/ AMHERST MA 01003
01060   BERT MENDELSON/ COMPUTER CENTER/ MCCONNELL HALL/ SMITH COLLEGE / NORTHAMPTON MA 01060
01247   S. J. BATTORY JR./ 15 MURRAY AVE./ NORTH ADAMS MA 01247
01420   KENNETH R. WADLAND/ COMPUTER SCIENCE DEPT./ FITCHBURG STATE CO LLEGE/ MAIL BOX NUMBER 6372/ FITCHBURG MA 01420/ (617) 345-2151 X181
01451   PETER CONKLIN/ BOLTON ROAD/ HARVARD MA 01451/ (617) 851-5071 X 2119
01505   JESSE HEINES/ DIGITAL EQUIPMENT CORPORATION/ 215 MAIN ST.RTE.7 0/ BOYLSTON MA 01505
01581   DAVID C. CLINE/ 1106 WINDSOR RIDGE/ WESTBORO MA 01581/ (617) 3 66-9509
01581   RICH COON/ SOFTWARE DEVELOPMENT/ DATA GENERAL CORP./ ROUTE 9/  WESTBORO MA 01581
01581   KENNETH L. WILLIAMS/ 135 E. MAIN ST. R12/ WESTBORO MA 01581/ ( 617) 366-9236
01581   NICHOLAS WYBOLT/ MS 71141/ DATA GENERAL CORP./ 15 TURNPIKE RD. / WESTBORO MA 01581/ (617) 366-8911
01701   DENIS KOMINSKY/ 1640 WORCESTER RD./ FRAMINGHAM MA 01701/ (617)  879-3654
01701   BERNIE ROSMAN/ MATH/CS DEPT./ FRAMINGHAM STATE COLLEGE/ FRAMIN GHAM MA 01701/ (617) 620-1220
01720   THEODORE R. CROWLEY/ 16 ALGONQUIN RD./ ACTON MA 01720/ (617) 3 66-8911 X 5725
01730   STEPHEN HATCH/ RAYTHEON COMPANY - MSD/ HARTWELL RD./ BEDFORD M A 01730/ (617) 274-7100
01730   EMANUEL WACHSLER/ 20 PAUL REVERE RD./ BEDFORD MA 01730/ (617)  275-0593/ (617) 890-3330
01741   DEAN BANDES/ PARKE MATHEMATICAL LABS INC./ ONE RIVER ROAD/ CAR LISLE MA 01741/ (617) 369-3818
01742   ATTN: INFORMATION CENTER/ ENVIRONMENTAL RESEARCH & TECH. INC./  696 VIRGINIA RD./ CONCORD MA 01742
01742   EDWARD E. L. MITCHELL/ MITCHELL & GAUTHIER ASSOCIATES/ P.O. BO X 685/ CONCORD MA 01742/ (617) 369-5115
01752   DONALD D. BURN/ 29-7 BRIARWOOD LN/ MARLBORO MA 01752/ (617) 48 5-6774
01752   MIKE GILBERT/ 39-1 BRIARWOOD LANE/ MARLBORO MA 01752/ (617) 48 1-4275
01752   ARON K. INSINGA/ 273B W. MAIN ST./ MARLBORO MA 01752/ (617) 48 5-4620
01752   MIKE KNUDSON/ 79A PHELPS ST./ MARLBORO MA 01752/ (617) 485-817
01754   BRUCE MACKENZIE/ 74 POWDER MILL RD./ MAYNARD MA 01754/ (617) 8 97-5429
01778   WILLIAM WOLFSON/ 188 PELHAM ISLAND RD./ WAYLAND MA 01778
01810   BRUCE ALLEN/ MODICON DIV./ GOULD INC./ P.O. BOX 83/ ANDOVER MA  01810/ (617) 475-4700
01810   ROBERT I. DEMROW/ 11 LINDA DRIVE/ ANDOVER MA 01810/ (617) 475- 1563
01821   ATTN: TECHNICAL LIBRARY/ MS 813/ HONEYWELL INFO. SYSTEMS INC./  300 CONCORD ROAD/ BILLERICA MA 01821
01824   JOHN DE ROSA JR./ 7 GLENN AVE APT 11/ CHELMSFORD MA 01824
01824   THOMAS J. STOODLEY III/ DISTRIBUTED SYSTEMS CO./ 17 WILSON ST. / CHELMSFORD MA 01824/ (617) 256-8742
01852   FRITZ EBERLE/ 578 ANDOVER ST./ LOWELL MA 01852/ (617) 454-8909
01867   GAYE MARR/ ADVERTISING/ ADDISON-WESLEY/ JACOB WAY/ READING MA  01867/ (617) 944-3700 X391
01880   STEVEN L. COOL/ ANALOGIC CORP./ AUDUBON RD./ WAKEFIELD MA 0188 0/ (617) 246-0300
01886   RICHARD KRASIN/ K SYSTEMS/ BOX 508/ WESTFORD MA 01886
01887   ALAIN J. HANOVER/ DYMOGRAPHIC SYSTEMS INC./ 355 MIDDLESEX AVE. / WILMINGTON MA 01887/ (617) 933-7000
01890   PETER STEIN/ 28 FRANKLIN RD./ WINCHESTER MA 01890
01945   ATTENTION: WILLIAM MAIN/ NEW ENGLAND MICRO TECHNOLOGY INC./ P. O. BOX 767/ MARBLEHEAD MA 01945/ (617) 631-6005
01945   JON F. HUERAS/ 34 OLD SALEM ROAD/ MARBLEHEAD MA 01945
02115   TIM KIEFFER/ 290 NEWBURY ST./ BOSTON MA 02115
02125   R. A. MORRIS/ MATH DEPT/ U OF MASSACHUSETTS - BOSTON/ BOSTON M A 02125
02134   DAN FYLSTRA/ 22 WEITZ ST. #3/ BOSTON MA 02134/ (617) 782-5932
02138   J. SCOTT DIXON/ DEPT OF CHEMISTRY/ HARVARD UNIV./ 12 OXFORD ST ./ CAMBRIDGE MA 02138
02138   CHARLES ROBERT MORGAN/ BOLT BERANEK AND NEWMAN/ 50 MOULTON STR EET/ CAMBRIDGE MA 02138/ (617) 491-1850 X502
02139   CHARLES L. BROOKS/ 16 ANTRIM STREET/ CAMBRIDGE MA 02139/ (617)  661-3671
02139   ROBERT FRANKSTON -COPY A/ P.O. BOX 70 - MIT BRANCH/ CAMBRIDGE  MA 02139
02139   DANIEL R. KILLORAN/ MAIL STOP 16/ CHARLES STARK DRAPER LAB./ 5 55 TECH SQUARE/ CAMBRIDGE MA 02139/ (617) 258-1438
02139   FRANCIS F. LEE/ 575 RESEARCH LAB. OF ELECTRONICS/ 36/ M.I.T./  CAMBRIDGE MA 02139
02139   JOHN M. STRAYHORN/ BOX 157 MIT BRANCH P.O./ CAMBRIDGE MA 02139 / (617) 923-1133
02142   ATTN: KINDLER ASSOCIATES INC./ ONE BROADWAY/ CAMBRIDGE MA 0214 2/ (617) 491-4963/ (617) 491-4415
02142   JAMES STEINBERG/ 863/ DOT/TSC/ KENDALL SQUARE/ CAMBRIDGE MA 02 142/ (617) 494-2015
02146   BEARDSLEY RUML II/ 59 HOLLAND RD/ BROOKLINE MA 02146/ (617) 27 7-9494
02149   WALTER L. PRAGNELL/ GRACE CHURCH RECTORY/ 9 WARREN STREET/ EVE RETT MA 02149
02154   RONALD V. BOSSLET/ GTE LABS INC./ 460 TOTTEN POND ROAD/ WALTHA M MA 02154/ (617) 890-4100
02154   ALAN B. FINGER/ GTE LABS/ 40 SYLVAN RD./ WALTHAM MA 02154/ (61 7) 890-4100
02154   ROBERT FRANKSTON -COPY B/ INTERACTIVE DATA CORP./ 486 TOTTENPO ND ROAD/ WALTHAM MA 02154
02154   JODY PAUL PERONI/ TEXAS INSTRUMENTS INC./ 504 TOTTEN POND RD./  WALTHAM MA 02154
02173   GEORGE S. GORDON JR./ 7 COACH RD./ LEXINGTON MA 02173
02173   DAVID GRABEL/ 125 REED ST./ LEXINGTON MA 02173/ (617) 861-9371
02173   MARGERY HARRIS/ HONEYWELL ELECTRO OPTICS CENTRAL/ 2 FORBES ROA D/ LEXINGTON MA 02173
02173   FRANK SCHWARTZ/ SOFTWARE ASSISTANCE INC./ 18 HARBELL ST./ LEXI NGTON MA 02173
02174   FRED COTTON/ 51 THORNDIKE ST./ ARLINGTON MA 02174
02174   DONALD WARREN/ 290 MASSACHUSETTS AVE./ ARLINGTON MA 02174
02176   CARLOS CHRISTENSEN/ 63 E. EMERSON ST./ MELROSE MA 02176/ (617)  665-5736
02178   ROBERT OSBORN/ DIALOG SYSTEMS/ 32 LOCUST ST./ BELMONT MA 02178 / (617) 489-2830
02181   GREGORY J. O'BRIEN/ INCOTERM CORP./ 65 WALNUT ST./ WELLESLEY H IL* MA 02181/ (617) 237-2100
02181   JAMES L. PYLES/ SOFTWARE PRODUCT PLANNING/ PRIME COMPUTER/ 40  WALNUT ST./ WELLESLEY HIL* MA 02181/ (617) 237-6990
02194   A. FREDERICK ROSENE/ GTE SYLVANIA/ 77 A STREET/ NEEDHAM HTS MA  02194/ (617) 449-2000 X2332
02195   JAMES F. HART/ CODEX CORP./ 15 RIVERSIDE AVENUE/ NEWTON MA 021 95/ (617) 969-0600
02215   TIMOTHY GRIESER/ COMP. CENTER/ BOSTON UNIV./ 111 CUMMINGTON ST ./ BOSTON MA 02215
02747   JOHN W. GRAY/ DEPT. OF ELECT. ENG./ SOUTHEASTERN MASSACHUSETTS  UNIV./ N. DARTMOUTH MA 02747
02871   DAVID J. DE FANTI/ SUBMARINE SIGNAL DIVISION/ RAYTHEON COMPANY / P.O. BOX 360/ PORTSMOUTH RI 02871/ (401) 847-8000
03055   L. DAVID BALDWIN/ RFD 2/ MILFORD NH 03055/ (603) 465-7857
03060   BILL MARSHALL/ SANDERS ASSOCIATES INC./ 95 CANAL ST./ NASHUA N H 03060/ (603) 885-2551
03102   RICHARD M. SMITH/ 77 GARDEN DR. NO. 3/ MANCHESTER NH 03102
03103   ATTN: DB/DC SOFTWARE ASSOC./ P.O. BOX 4695/ MANCHESTER NH 0310 3
03755   JAMES P. MAGNELL/ LOGIC ASSOCIATES/ BOX 568/ HANOVER NH 03755
03768   STEVEN CAMPBELL/ SOFTWARE SYSTEMS/ PINNACLE RD./ LYME NH 03768 / (603) 795-2244
04473   MAL CAREY/ CREATIVE COMPUTING/ P.O. BOX 147/ STILLWATER ME 044 73
05402   LILLIAN WILHELMSON/ 1320/ GENERAL ELECTRIC CO./ LAKESIDE AVE./  BURLINGTON VT 05402
06032   HOUSTON P. LOWRY/ 49 HIGH STREET/ FARMINGTON CT 06032
06035   TIMOTHY DENNIS/ 62 MANITOOK LAKE/ GRANBY CT 06035/ (203) 653-4 492
06430   REID SMITH-VANIZ/ PRODUCT DEVELOPMENT SERVICES/ 2000 BLACK ROA D TURNPIKE/ FAIRFIELD CT 06430
06455   C. A. ZANONI/ ZYGO CORP./ LAUREL BROOK RD./ MIDDLEFIELD CT 064 55/ (203) 347-8506
06457   FRANK W. TYRON JR./ 31 EVERGREEN AVE./ MIDDLETOWN CT 06457
06460   ARTHUR LACROIX LA CROIX/ APT. A-3/ 1060 NEW HAVEN AVE./ MILFOR D CT 06460
06468   RICHARD ROTH/ 39 WILLIAMS DR./ MONROE CT 06468
06497   BRUCE HIBBARD/ 165 COLONY _TRET/ STRATFORD CT 06497
06830   R. ETZI/ M.S. 186/ AMERICAN CAN CO./ AMERICAN LANE/ GREENWICH  CT 06830
06851   PATRICIA J. GARSON/ TURNKEY SYSTEMS INC./ 111 EAST AVE./ NORWA LK CT 06851/ (203) 853-2884
06856   ABRAHAM SAVITZKY/ M/S 284/ PERKIN-ELMER CORP./ MAIN AVENUE/ NO RWALK CT 06856/ (203) 762-1000
07009   WILLIAM MEIER/ M W ASSOCIATES/ 735 POMPTON AVE./ CEDAR GROVE N J 07009
07054   JAMES B. THOMPSON JR./ R. SHRIVER ASSOCIATES/ 120 LITTLETON RO AD/ PARSIPPANY NJ 07054/ (201) 335-7800
07060   GEORGE E. HOLZ/ VARITRONICS SYSTEMS/ 97 GRANDVIEW AVE/ N. PLAI NFIELD NJ 07060/ (201) 754-9429
07067   LARRY STEIN/ 151 KLINE BLVD./ COLONIA NJ 07067/ (201) 574-3373
07083   GENE KEENOY/ INFORMATION MANAGEMENT SERV./ KEAN COLLEGE OF NEW  JERSEY/ MORRIS AVE./ UNION NJ 07083
07207   RUSSELL J. PEPE/ CONSUMER PRODUCTS GROUP/ DEPT. 384-20/ SINGER  CORP./ 321 1ST ST./ ELIZABETH NJ 07207/ (201) 527-6000
07462   STEVEN R. RAKITIN/ STAR ROUTE / BOX 32/ VERNON NJ 07462
07632   JAMES W. O'CONNOR/ EHRHART-BABIC ASSOCIATES INC./ 120 ROUTE 9W / ENGLEWOOD CL. NJ 07632/ (201) 461-6700
07724   CHRISTOPHER J. HENRICH/ SOFTWARE DEVELOPMENT/ INTERDATA INC./  106 APPLE STREET/ TINTON FALLS NJ 07724/ (201) 747-7300 X549
07724   DAVID NEAL/ PERKIN-ELMER DATA SYSTEMS/ 106 APPLE STREET/ TINTO N FALLS NJ 07724/ (201) 747-7300/ (913) 532-6350 (WORK)
07762   WILLIAM D. BRISCOE/ 703 FIFTH AVENUE/ SPRING LAKE NJ 07762/ (2 01) 449-7681
07764   G. B. SWARTZ/ MATH / COMP. SCI. DEPT./ MONMOUTH COLLEGE/ CEDAR  AVE./ W. LONG BRANCH NJ 07764/ (201) 222-6600 X381
07932   KARL P. ADEY/ ENERGY GROUP/ VYDEC INC./ 9 VREELAND RD./ FLORHA M PARK NJ 07932/ (201) 822-2100
07960   WARREN SCHODER/ MORRIS COUNTY SAVINGS BANK/ 21 SOUTH STREET/ M ORRISTOWN NJ 07960/ (201) 539-0500
08002   LEE FRANK/ BTI COMP. SYSTEMS/ 3 EXECUTIVE CAMPUS/ CHERRY HILL  NJ 08002/ (609) 662-1122
08033   ANN S. ADAMS/ 718 GRAISBURY AVENUE/ HADDONFIELD NJ 08033
08046   LARRY E. ELLISON/ 19 HUNTINGTON LANE/ WILLINGBORO NJ 08046/ (6 09) 877-8847
08052   GEORGE P. CAMPBELL/ 3C EMERSON ROAD/ MAPLE SHADE NJ 08052/ (60 9) 779-8688
08101   R. K. PAETZOLD/ TACS-204-2/ RCA CORP./ CAMDEN NJ 08101/ (609)  338-4106
08540   ROBERT BOYLAN/ METROMATION/ 1101 STATE ROAD/ PRINCETON NJ 0854 0
08540   HERMAN EUREMA/ P.O. BOX 2204/ PRINCETON NJ 08540
08540   IRVING S. SCHECHTMAN/ NATIONAL COMPUTER ANALYSTS INC./ HIGHWAY  1 & FARBER RD./ PRINCETON NJ 08540/ (609) 452-2800
08540   HENRY WOOD/ 259 MT. LUCAS ROAD/ PRINCETON NJ 08540
08753   ROBERT C. PERLE/ 1108 RUBY DRIVE/ TOMS RIVER NJ 08753/ (201) 5 32-2831
08816   CHARLES ANDERSON/ 26 TAYLOR AVE./ EAST BRUNSWICK NJ 08816
08822   GEOFFREY F. WALKER/ RD 1 - BOX 56/ FLEMINGTON NJ 08822
08826   GEORGE B. DIAMOND/ DIAMOND AEROSOL CORP./ RD #1/ GLEN GARDNER  NJ 08826/ (201) 832-7128
08854   JOSEPH A. MEZZAROBA/ 15 BIRCHVIEW DRIVE/ PISCATAWAY NJ 08854/  (201) 469-5176/ (215) 679-9900 (HOME)
08854   JAMES R. SCHRAGE/ 255 OLD NEW BRUNSWICK RD./ PISCATAWAY NJ 088 54/ (201) 981-0190
09175   ATTN: SAM CALVIN/ COMPUTER EDUCATION/ DARMSTADT CAREER CENTER/  APO NEW YORK NY 09175/ 06151-69-2371-7203 (GERMANY)
09403   EUGENE K. GOODELL/ ODCSI SYS DIV/ HQ USAREVR & 7A/ BOX 353/ AP O NY 09403
10001   ED LEARY/ SYSTEMS SOFTWARE/ CBS DATA CENTER/ 2 PENN PLAZA/ NEW  YORK NY 10001/ (212) 975-4321
10003   BRIAN GLASSER/ B.G. SOUND/ 60 E. 9TH ST. APT 615/ NEW YORK NY  10003
10005   JAMES L. MORAN/ 700/ C/O FAHNESTOCK & CO./ 110 WALL ST./ NEW Y ORK NY 10005
10014   BARBARA BERGER/ 704 WASHINGTON ST./ NEW YORK NY 10014/ (212) 9 24-5172
10016   GLENN ENTIS/ 203 E. 27TH ST. APT.52/ NEW YORK NY 10016/ (212)  689-4926
10020   JOHN G. POSA/ ELECTRONIC MAGAZINE/ 1221 AVE. OF THE AMERICAS/  NEW YORK NY 10020
10021   MICHAEL H. LESKIN/ 218 E. 74TH ST. APT 1-R/ NEW YORK NY 10021/  (212) 679-0804
10021   ANTHONY TOOGOOD/ COMPUTER ASSOCIATES INC./ 655 MADISON AVE./ N EW YORK NY 10021/ (212) 355-3333
10022   CHARLES H. BROWNING/ PHELPS DODGE CORP./ 300 PARK AVE./ NEW YO RK NY 10022/ (212) 751-3200 X289
```

```
10022    C. H. BROWNING/ PHELPS-DODGE CORP./ 300 PARK AVE/ NEW YORK NY  10022
10023    PETER RENNICK/ 201 W 70TH ST APT 33A/ NEW YORK NY 10023
10024    IRA A. CLARK/ COMPUTER SYSTEMS DEVELOPMENT/ 275 CENTRAL PARK W EST/ NEW YORK NY 10024/ (212) 787-0767
10024    PAUL SPRECHER/ 241 WEST 77TH STREET/ NEW YORK NY 10024/ (212)  787-0176 (HOME)/ (212) 873-0677 (WORK)
10038    ANDREW VARANELLI/ COMP. AND INFO. SCI./ 721A/ PACE UNIVERSITY/  PACE PLAZA/ NEW YORK NY 10038
10530    ATTN: LJS COMPUTER SERVICES/ 6 CATERSON TERRACE/ HARTSDALE NY  10530/ (914) 946-1632
10533    FRANK PAVLIK/ 163 S. BROADWAY/ IRVINGTON NY 10533/ (914) 591-6 215
10570    JUSTINA JOHNSON/ P.O. BOX 33/ PLEASANTVILLE NY 10570
10580    LAWRENCE M. GARCIA/ SYNTAX SYSTEMS INC./ 65 REYMONT AVE./ RYE  NY 10580/ (914) 967-8421
10598    VICTOR S. MILLER/ THOS J. WATSON RESEARCH CENTER/ IBM/ P.O. BO X 218/ YORKTOWN HGTS NY 10598
11215    BOB SIEGEL/ 401 FOURTH ST./ BROOKLYN NY 11215
11374    ESTHER ROSENSTOCK/ 97-40 62ND DRIVE/ REGO PARK NY 11374
11432    DANIEL LEY/ 85-50 169TH ST./ JAMAICA NY 11432
11530    ANTHONY R. HEALY/ 41 GREENRIDGE AVE/ GARDEN CITY NY 11530/ (51 6) 437-3823 (HOME)/ (212) 363-7380 (WORK)
11552    PETER J. HARRINGTON/ 149 WILLETS AVE./ WEST HEMPSTEAD NY 11552 / (516) 293-8400
11566    MORRIS MOLIVER/ 1928 LOWELL LANE/ MERRICK NY 11566/ (516) 623- 4122
11713    LOUISE GOLDSTEIN/ 71 S. COUNTRY RD./ BELLPORT NY 11713/ (516)  286-8241
11714    N. KERMAN/ MS A31-005/ GRUMMAN AEROSPACE/ BETHPAGE NY 11714/ ( 516) 575-7403
11756    ROBERT SCHUTZ/ 93 MERIDIAN ROAD/ LEVITTOWN NY 11756/ (516) 735 -7244
11790    DAVID VANCE/ LIBRARY E-2340/ MUSEUM COMMUNITY NETWORK INC./ SU NY - STONY BROOK/ STONY BROOK NY 11790/ (516) 246-6077
11968    ROBERT TUPPER/ GRUMMAN AEROSPACE/ INDIAN ROAD/ SOUTHAMPTON NY  11968
11973    FRANK LEPERA/ APPLIED MATHEMATICS DEPT./ BROOKHAVEN NATIONAL L ABORATORY/ UPTON NY 11973/ (516) 345-4112
12202    G. O'SCHENECTADY/ 20 ELM STREET/ ALBANY NY 12202/ (518) 465-28 87
12222    PETER BLONIARZ/ COMPUTER SCIENCE DEPT./ ES 316/ SUNY AT ALBANY / ALBANY NY 12222/ (510) 457-4605
12345    MATTHEW KAZLAVSKAS/ BLDG 28-310/ GENERAL ELECTRIC CO./ 1 RIVER  ROAD/ SCHENECTADY NY 12345
12401    G. M. KREMBS/ DEPT 66A BLDG 003/ IBM CORP./ NEIGHBORHOOD RD./  KINGSTON NY 12401
12561    JEANNE FERRANTE/ 201 N. OHIOVILLE RD./ NEW PALTZ NY 12561
13203    MAURY GOLDBERG/ MINI MICRO MART/ 1618 JAMES STREET/ SYRACUSE N Y 13203/ (315) 422-4467
13210    STEVE QUALLINE/ MACHINERY HALL/ SYRACUSE UNIVERSITY/ SYRACUSE  NY 13210/ (315) 423-3812
13323    ROBERT J. ELLISON/ MATH DEPT./ HAMILTON COLLEGE/ CLINTON NY 13 323/ (315) 859-4138
13760    BARBARA K. NORTH/ 304 HILLSIDE TERRACE/ ENDWELL NY 13760
13902    MARY DIEGERT/ MATHEMATICS DEPT./ BROOME COMMUNITY COLLEGE/ BIN GHAMTON NY 13902/ (607) 722-5022
14420    NORMAN V. PLYTER/ ACADEMIC COMPUTER CENTER/ HARTWELL HALL/ SUN Y - BROCKPORT/ BROCKPORT NY 14420
14450    E. GOTTWALD/ 2 TILEGATE GLEN/ FAIRPORT NY 14450/ (716) 423-779 7/ (716) 223-5383
14502    JOHN L. DEBES/ BOX 167/ MACEDON NY 14502/ (202) 447-0547
14580    RICHARD ALRUTZ/ 241 W128/ XEROX CORP./ 800 PHILLIPS RD./ WEBST ER NY 14580
14627    JAMES R. LOW/ DEPT. OF COMP. SCI./ MATHEMATICAL SCIENCES BLDG. / UNIV. OF ROCHESTER/ ROCHESTER NY 14627
14850    CHARLES N. ARROWSMITH/ NCR CORP./ 950 DANBY RD./ ITHACA NY 148 50/ (607) 273-5310
14853    MARIANN CARPENTER/ G-24 URIS HALL/ OFFICE OF COMP. SERV./ CORN ELL UNIVERSITY/ ITHACA NY 14853/ (607) 256-7341
14853    HAL PERKINS/ DEPT. OF COMPUTER SCIENCE/ CORNELL UNIVERSITY/ IT HACA NY 14853/ (607) 256-4934
15069    C. Y. BEGANDY/ ALCOA TECHNICAL CENTER/ ALUMINUM CO. OF AMERICA / ALCOA CENTER PA 15069/ (412) 339-6651
15213    CHUCK AUGUSTINE/ COMPUTATION CENTER/ CARNEGIE MELLON UNIV./ SC HENLEY PARK/ CARNEGIE MELLO*/ PITTSBURGH PA 15213/ (412) 578-2649
15213    DAVID B. CROUSE/ GRAPHIC ARTS TECHNICAL FOUNDATION/ 4615 FORBE S AVE/ PITTSBURGH PA 15213
15213    JIM TSEVDOS/ CARNEGIE-MELLON UNIV./ P.O. BOX 132/ PITTSBURGH P A 15213/ (412) 665-1036
15217    ARON SHTULL TRAURING/ 5637 HOBART APT.33/ PITTSBURGH PA 15217/  (412) 421-4066
15221    ROBERT J. KING/ 2337 MARBURY ROAD/ PITTSBURGH PA 15221/ (412)  372-1212
15222    STEPHEN G. HUSSAR/ PPG INDUSTRIES INC./ ONE GATEWAY CENTER/ PI TTSBURGH PA 15222
15230    FREDERICK E. SHIPLEY JR./ GULF RES. & DEV. CO./ PO BOX 2038/ P ITTSBURGH PA 15230/ (412) 362-1600
15236    ELMER T. BEACHLEY/ P.O. BOX 18046/ PITTSBURGH PA 15236
15238    ROBERTA WACHTER/ INDUSTRY SYSTEMS DIVISION/ 200 BETA DRIVE/ PI TTSBURGH PA 15238/ (412) 782-1730 X544
15260    ALAN M. LESGOLD/ LRDC COMPUTER FACILITY/ UNIV. OF PITTSBURGH/  3939 O'HARA ST./ PITTSBURGH PA 15260/ (412) 624-4901
16057    PETER RICHETTA/ 287 NORMAL AVENUE/ SLIPPERY ROCK PA 16057/ (41 2) 794-3531
16701    FRANK BREWSTER/ 1 N. VISTA AVE./ BRADFORD PA 16701/ (814) 368- 6319
17331    MICHAEL D. BROWN/ R. H. SHEPPARD CO. INC./ 101 PHILADELPHIA ST REET/ HANOVER PA 17331/ (717) 637-3751
18016    CHARLES T. LEWIS/ BETHLEHEM STEEL/ 1581 MARTIN TOWER/ BETHLEHE M PA 18016/ (215) 694-6359
18017    ROBERT COLE/ 782 BARRYMORE LANE/ BETHLEHEM PA 18017/ (215) 865 -6509
18103    RICHARD J. CICHELLI/ 901 WHITTIER DRIVE/ ALLENTOWN PA 18103/ ( 215) 797-3153 (HOME)/ (215) 253-6155 (WORK)
18914    PHILIP W. ROSS/ 8 HICKORY LANE/ CHALFONT PA 18914
19004    JIM SHALLOW/ 115 BIRCH AVE./ BALA CYNWYD PA 19004
19020    ROBERT H. TODD JR/ BRIARWOOD #1167/ CORNWALLIS HGT PA 19020/ ( 215) 752-4604
19020    BOB LIDRAL/ 3806 BENSALEM BLVD. #214/ CORNWELLS HTS. PA 19020
19044    JAMES A. MCGLINCHEY/ 296 BLAIR MILL RD. APT B-7/ HORSHAM PA 19 044
19047    RODNEY MEBANE/ 600 OLD STREET ROAD #AT13/ TREVOSE PA 19047
19083    T. L.(FRANK) PAPPAS/ 338 FRANCIS DRIVE/ HAVERTOWN PA 19083/ (2 15) 789-3206
19102    RICHARD L. DAY/ TIME SHARE SUPPORT GROUP/ BELL TELEPHONE OF PE NNSYLVANIA/ ONE PARKWAY/ PHILADELPHIA PA 19102
19111    ALAN M. KANISS/ 1327 MCKINLEY ST./ PHILADELPHIA PA 19111/ (215 ) 441-2051 (WORK)
19117    DAN MORTON/ 359 NORTHWOOD AVE./ PHILADELPHIA PA 19117
19122    BILL CHESWICK/ COMPUTER ACTIVITY - SYSTEMS GROUP/ TEMPLE UNIV. / BROAD & MONTGOMERY STREETS/ PHILADELPHIA PA 19122/ (215) 787-8527 (WORK)
                      (215) 862-2153 (HOME)
19128    JOHN F. RATTI/ 300 HERMITAGE ST./ PHILADELPHIA PA 19128
19145    PAUL J. PANTANO/ 2323 S. 17 ST./ PHILADELPHIA PA 19145
19172    NICK CVETKOVIC/ VIM 6E/ PENN MUTUAL LIFE/ 510 WALNUT ST./ PHIL ADELPHIA PA 19172
19317    BOB KELLER/ CONCORD WAY/ CHADDS FORD PA 19317
19342    JAMES I. WILLIAMS/ RD 4 BOX 18/ GLEN MILLS PA 19342/ (215) 648 -3554
19380    THOMAS J. AHLBORN/ DEPT. MATH/ WEST CHESTER STATE COLLEGE/ WES T CHESTER PA 19380/ (215) 436-2181
19380    GARY L. WEIGEL/ 202 WESTBROOK DRIVE/ WEST CHESTER PA 19380/ (2 15) 328-9100 (WORK)/ (215) 696-8739
19401    BILL BRENNAN/ 39 JODY DRIVE/ NORRISTOWN PA 19401/ (215) 277-24 66
19422    PETER A. NAYLOR/ MS B/220M/ SPERRY UNIVAC/ P.O. BOX 500/ BLUE  BELL PA 19422/ (215) 542-3732
19422    J. P. M. STOFBERG/ MS B/220M/ SPERRY UNIVAC/ P.O. BOX 500/ BLU E BELL PA 19422/ (215) 542-4011
19446    MICHAEL ROSIAK/ 122 ARDWICK TERRACE/ LANSDALE PA 19446
19446    RICHARD WHIFFEN/ ENERTEC/ 19 JENKINS AVE/ LANSDALE PA 19446/ ( 215) 362-0966
19518    RICHARD A. JOKIEL/ P.O. BOX 136/ DOUGLASVILLE PA 19518
19711    WILLIAM Q. GRAHAM/ COMPUTING CENTER/ U. OF DELAWARE/ 192 S. CH APEL ST./ NEWARK DE 19711/ (302) 453-6032
19711    FRED A. MASTERSON/ DEPT. OF PSYCHOLOGY/ 220 WOLF HALL/ UNIV. O F DELAWARE/ NEWARK DE 19711
19898    SAMUEL C. KAHN/ INFO SYSTEMS DEPT/ N-1450 PLANNING DIV./ DU PO NT & CO./ WILMINGTON DE 19898
20005    JOHN B. HOLMBLAD/ TELENET COMMUNICATIONS CORP./ 1012 14TH ST.  NW/ WASHINGTON DC 20005/ (202) 637-7900
20016    RICHARD B. FITZ/ 4215 38TH STREET NW/ WASHINGTON DC 20016
20022    R. CARLYLE NEELY JR./ 10114 KATHLEEN DRIVE/ FRIENDLY MD 20022/  (301) 248-6244
20037    VINCENT STANFORD/ DEPT. OF MEDICINE COMPUTER RESEARCH C*/ 517  ROSS HALL/ GEORGE WASHINGTON UNIV./ 2300 EYE ST. NW/ WASHINGTON DC 20037/
20052    MICHAEL B. FELDMAN/ DEPT. OF EE & CS/ GEORGE WASHINGTON UNIV./  WASHINGTON DC 20052/ (202) 676-7593
                                                                                                                     (202) 676-3673
20052    E. MICHAEL HAMILTON/ C.A.A.C./ GEORGE WASHINGTON UNIV./ 2013 G  STREET NW/ WASHINGTON DC 20052/ (202) 676-6140
20250    T. Q. STEVENSON/ O & F DATA SERVICES/ RM 4646-S/ USDA/ WASHING TON DC 20250/ (202) 447-6275
20375    NIELS K. WINSOR/ CODE 6752/ NAVAL RESEARCH LABORATORY/ WASHING TON DC 20375/ (202) 767-3134
20755    JOHN NOLAN/ NATIONAL SECURITY AGENCY/ R51/ DEPARTMENT OF DEFEN SE/ 9800 SAVAGE ROAD/ FT. MEADE MD 20755/ (301) 796-6461
20770    CAROL B. HOWELL/ P.O. BOX 326/ GREENBELT MD 20770/ (301) 982-2 281 (GODDARD)
20810    RANDY BARTH/ 9206 CANTERBURY RIDING/ LAUREL MD 20810
20822    TOM ENTERLINE/ 13311 CHAUNCEY PL. #203/ MT. RAINIER MD 20822
20850    TOM LOVE/ SOFTWARE METHODOLOGY/ GENERAL ELECTRIC/ 401 N. WASHI NGTON ST./ ROCKVILLE MD 20850/ (301) 340-4000
20852    ATTN:INFORMATICS INC. BOOKSTORE/ 6011 EXECUTIVE BLVD./ ROCKVIL LE MD 20852
20852    PATRICIA SHELLY/ INFORMATICS INC. BOOK STORE/ 6011 EXECUTIVE B LVD./ ROCKVILLE MD 20852
20853    THOMAS A. MARCINIAK/ 13311 ARCTIC AVENUE/ ROCKVILLE MD 20853/  (301) 942-0538
20855    BOB ROGERS/ 18625 AZALEA DRIVE/ DERWOOD MD 20855/ (301) 869-20 89
20903    H. A. COOK/ 1223 CRESTHAVEN DR./ SILVER SPRING MD 20903
20904    JOHN G. GUTHRIE/ COMPUTER ENTRY SYSTEMS INC./ 2141 INDUSTRIAL  PARKWAY/ SILVER SPRINGS MD 20904/ (301) 622-3500
21030    SPEC BOWERS/ 9H BREEZY HILL CT./ COCKEYSVILLE MD 21030
21043    WALLACE KENDALL/ 9002 DUNLOGGIN ROAD/ ELLICOTT CITY MD 21043/  (301) 465-4253
21045    BARTON F. NORTON/ CHROMA/ P.O. BOX 126/ COLUMBIA MD 21045/ (30 1) 992-7404
21203    ROB BIDDLECOMB/ MS 451/ WESTINGHOUSE ELECTRIC CORP./ SDD EAST  BOX 746/ BALTIMORE MD 21203/ (301) 765-6322
21204    EDWARD W. KNUDSEN/ AAI CORP./ P.O. BOX 6767/ BALTIMORE MD 2120 4/ (301) 666-1400
21235    LESTER SACHS/ MS 3-0-25 OPER. BLDG/ SOCIAL SECURITY ADMINISTRA TION/ 6401 SECURITY BOULEVARD/ BALTIMORE MD 21235/ (301) 594-5482
21401    DAVID V. SOMMER/ RT. 5 BOX 220/ ANNAPOLIS MD 21401
21701    J. BOGAR/ FREDERICK ELECTRONICS CORP./ P.O. BOX 502/ FREDERICK  MD 21701
22030    ATTN: J. M. P. ASSOCIATES/ 3219 PRINCE WILLIAM DR./ FAIRFAX VA  22030/ (703) 591-8525
22101    DAVID AULT/ COMPUTER SCIENCE/ WP 615/ THE MITRE CORP./ 1820 DO LLY MADISON BLVD./ MCLEAN VA 22101/ (703) 437-7898 (HOME)
22101    H. F. HESSION/ ADVANCED RECORD SYSTEMS ENGINEERING/ WESTERN UN ION/ 7916 WEST PARK DRIVE/ MCLEAN VA 22101/ (703) 790-2241
22110    ROBERT A. GIBSON/ 8902 NICOL LANE #207/ MANASSAS VA 22110/ (70 3) 367-4792 (WORK)/ (703) 369-5640 (HOME)
22151    HAROLD D. JENKINS JR./ SPRINGFIELD SUPPORT CENTER/ FAIRFAX COU NTY PUBLIC SCHOOLS/ 6707 ELECTRONIC DR./ SPRINGFIELD VA 22151
22151    PAUL T. DYKE/ RESOURCE SYSTEM & PROGRAM ANALYSIS/ 428 GHI BLDG ./ U.S.D.A./ 500 12TH ST S.W./ WASHINGTON DC 22151
22180    ROBERT G. FITZGERALD/ 133 EAST STREET N.E./ VIENNA VA 22180/ ( 301) 868-5229
22205    WALTER A. WHITE/ 6048 N 9TH ST./ ARLINGTON VA 22205
22209    LARRY DUBY/ 1500 ARLINGTON BLVD. #910/ ARLINGTON VA 22209
22309    GERALD P. SHABE/ 3206 NORWICH TERRACE/ ALEXANDRIA VA 22309/ (7 03) 360-5587
22310    RONALD OTTO/ 5800 LANE DRIVE/ ALEXANDRIA VA 22310
22312    ART BARRETT/ THE MITRE CORP./ 4112 CENTURY CT./ ALEXANDRIA VA  22312
22401    RONALD HARTUNG/ 1114 THOMAS JEFFERSON PL./ FREDRICKSBURG VA 22 401/ (703) 373-6573
22801    MICHAEL STAUFFER/ EASTERN MENNONITE COLLEGE/ HARRISONBURG VA 2 2801
22901    AVERY CATLIN/ THIMBLE FARM/ ROUTE 5 - BOX 363/ CHARLOTTESVIL*  VA 22901
22923    LINWOOD FERGUSON/ RT 1 BOX 3C - LAKE SAPONI/ BARBOURSVILLE VA  22923/ (804) 973-5166
```

```
23185          DOUGLAS DUNLOP/ 1502 CONWAY DRIVE - APT. 103/ WILLIAMSBURG VA 23185/ (804) 826-1725
23284          AGNES H. ELMORE/ COMPUTING ACTIVITIES/ VIRGINIA COMMONWEALTH U NIV./ 1015 FLOYD AVE./ RICHMOND VA 23284
23502          DAVID E. HAMILTON/ SUITE 106/ #18 KOGER EXECUTIVE CENTER/ NORF OLK VA 23502/ (804) 461-0268
23505          R. E. CRITTSINGER JR./ 136 BLAKE ROAD/ NORFOLK VA 23505
23505          LLOYD D. FINK/ AIR CARGO INC./ P.O. BOX 9793/ NORFOLK VA 23505 / (804) 480-2660
23669          JOHN C. CLARSON/ 303 TENDERFOOT COURT/ HAMPTON VA 23669
27702          WILLIAM H. DIUGIUD/ PLANNING DIV./ CITY OF DURHAM/ 101 CITY HA LL PLAZA/ DURHAM NC 27702
28214          WARREN C. FORDHAM/ MCCLURE LUMBER CO./ 6000 MT. HOLLY RD/ CHAR LOTTE NC 28214
28704          CARROLL B. ROBBINS JR./ APT 32/ ARDEN ARMS APTS./ ARDEN NC 287 04/ (919) 684-0168
29206          HOWARD EISENSTEIN/ 6616 DARE CIRCLE/ COLUMBIA SC 29206/ (803)  782-5041
29210          BILL RAEUBER/ 149 LEEWARD RD./ COLUMBIA SC 29210/ (803) 777-60 01
30021          CRAIG M. INGLIS/ 1420-C POST OAK DR./ CLARKSTON GA 30021
30033          JOHN P. CUCHES/ THE HYDE COMPANY/ 2169 CLAIRMONT RD NE/ DECATU R GA 30033
30067          HENRY D. KERR III/ 4820 HAMPTON LAKE DRIVE/ MARIETTA GA 30067/  (404) 971-2197
30303          DARRELL PREBLE/ COMPUTER CENTER USER SERVICES/ GEORGIA STATE U NIVERSITY/ ATLANTA GA 30303/ (404) 658-2683
30305          JEFFREY H. BIGGERS/ SUITE 411/ DTW INC./ 3100 MAPLE DRIVE NE/  ATLANTA GA 30305
30305          WILLIAM G. CHRISTIAN/ SUITE 450/ CLS INC./ 3100 MAPLE DRIVE NE / ATLANTA GA 30305
30305          FRANK S. SPARKMAN/ SUITE 411/ DTW INC./ 3100 MAPLE DRIVE NE/ A TLANTA GA 30305
30305          DAVID T. WILSON/ SUITE 411/ DTW INC./ 3100 MAPLE DRIVE NE/ ATL ANTA GA 30305
30327          JOHN WEST/ DIGITAL SYSTEMS DESIGN GROUP/ 4559 DUDLEY LANE NW/  ATLANTA GA 30327/ (404) 894-2264
30342          K. M. ALBRIGHT/ SYSTEMS ANALYSIS/ SUITE 600/ SPERRY UNIVAC/ 57 75C PEACHTREE DUNWOODY RD./ ATLANTA GA 30342/ (404) 256-5690
30354          RICHARD P. DE ROBERTS/ FEDERAL AVIATION ADMINISTRATION/ P.O. B OX 82822/ ATLANTA GA 30354/ (404) 763-7478 (OFF.)/ (404) 876-5370 (RES.)
32204          ATTENTION: ROY W. FILEGER/ SUITE 110 EAST/ COMPUTER POWER/ 661  RIVERSIDE AVE/ JACKSONVILLE FL 32204
32304          PEGGY ROBLYEN/ EDUCATIONAL COMPUTING PROJECT/ FLORIDA STATE DE PT. OF EDUCATION/ TALLAHASSEE FL 32304
32407          ANNA WATSON/ 3705 DELWOOD DRIVE/ PANAMA CITY FL 32407/ (904) 2 34-4423
32670          RICHARD J. NAST/ 1721 SW 55TH LANE/ OCALA FL 32670
33065          HOWARD S. MARSHALL JR./ 2648 NW 86TH AVE./ CORAL SPRINGS FL 33 065
33068          DEAN JAMES/ 7440 S.W. 10TH ST. - #102/ N. LAUDERDALE FL 33068
33142          MONTE ELLIS/ RYDACOM INC./ 3401 NW 36TH ST./ MIAMI FL 33142
33181          JAMES GROSSMAN/ 2365 MAGNOLIA DR./ N. MIAMI FL 33181/ (305) 89 1-3440
33528          CLARA L. JOHNSON/ MEDIA RESEARCH DIV. - ENGINEERING/ A. C. NIE LSON CO./ 375 PATRICIA AVE/ DUNEDIN FL 33528/ (813) 734-5473
33549          HERBERT M. BRYANT JR./ 14410 HELLENIC DR. F19/ LUTZ FL 33549
33601          R. D. EMRICK/ FIRST FLORIDA TOWER/ GTE DATA SERVICES/ P.O. BOX  1548/ TAMPA FL 33601/ (813) 224-3131
33803          ALLEN F. DOWNARD/ 3008 REDWOOD AVE./ LAKELAND FL 33803
35801          MARVIN E. KURTTI/ 1327 MONTE SANO BLVD. S.E./ HUNTSVILLE AL 35 801/ (205) 837-7610
35803          DAVID MCQUEEN/ 2410 ARROWWOOD DR./ HUNTSVILLE AL 35803/ (205)  881-3628
37076          LARRY D. BOLES/ 649 DENVER DRIVE/ HERMITAGE TN 37076
37660          J. W. DISSELKAMP/ 202 BUILDING 54/ TENNESSEE EASTMAN COMPANY/  KINGSPORT TN 37660/ (615) 246-2111
40206          TOM EUBANK/ PRAGMATECH/ 2310 MELLWOOD AVE./ LOUISVILLE KY 4020 6/ (502) 895-1230
40583          BEVERLY SWISSHELM/ KENTUCKY CNTR FOR ENERGY RES. LABORAT*/ UNI V. OF KENTUCKY/ IRON WORKS PIKE BOX 13015/ LEXINGTON KY 40583/ (606) 252 5535
43147          RICHARD L. MAHN/ 245 W. COLUMBUS ST./ PICKERINGTON OH 43147
43229          RICHARD E. ADAMS/ 967 ATLANTIC AVE #634/ COLUMBUS OH 43229/ (6 17) 436-3206
43230          RICHARD GREENLAW/ 251 COLONY COURT/ GAHANNA OH 43230/ (614) 47 5-0172
43762          RALPH G. HOLLINGSWORTH JR./ 186 MONTGOMERY BLVD./ NEW CONCORD  OH 43762
43778          TOM LEGRAZIE/ RURAL ROUTE 1/ SALESVILLE OH 43778
44092          LYNN C. HUTCHINSON/ BAILEY CONTROL COMPANY/ 29801 EUCLID AVE/  WICKLIFFE OH 44092/ (216) 943-5500
44106          JACK D. ALANEN/ JENNINGS COMPUTING CENTER/ CASE WESTERN RESERV E UNIV./ CLEVELAND OH 44106/ (216) 368-2800
44106          M. MARVINNEY/ DEPT. OF BIOMETRY/ 150 WEARN BLDG./ CASE WESTERN  RESERVE UNIV/ CLEVELAND OH 44106
44106          PAUL MEILAND/ DENTAL SCHOOL CLINICS/ CASE WESTERN RESERVE UNIV ./ 2123 ABINGTON ROAD/ CLEVELAND OH 44106
44107          BILL SHANNON/ 2038 ARTHUR/ LAKEWOOD OH 44107
44512          ATTN:WESTERN RESERVE COMMUNICATIONS/ 424 INDIANOLA ROAD/ YOUNG STOWN OH 44512
45201          WILLIAM R. METZ/ MSD - DEVELOPMENT/ THE PROCTER & GAMBLE COMPA NY/ P.O. BOX 599/ CINCINNATI OH 45201/ (513) 562-2747
45215          G. D. MONTILLON/ 351 FLEMMNGRD./ CINCINNATI OH 45215
45241          FRANCIS H. BEARDEN/ DATA SYSTEMS/ CINCINNATI ELECTRONICS CORP. / 2630 GLENDALE-MILFORD ROAD/ CINCINNATI OH 45241/ (513) 563-6000 X140
45244          CLINTON HERLEY/ MEDIATOR INC./ 2812 SADDLEBACK DRIVE/ CINCINNA TI OH 45244
45342          D. R. HILL/ MONSANTO RESEARCH CENTER/ P.O. BOX 32/ MIAMISBURG  OH 45342
45409          DAN C. WATSON/ WRIGHT BROS./ BOX 541/ DAYTON OH 45409/ (513) 2 23-2348
45414          LAWRENCE A. SHIVELY/ 6014 FREDERICK ROAD/ DAYTON OH 45414
45424          M. B. CLAUSING/ 5603 FISHER DRIVE/ DAYTON OH 45424/ (614) 236- 3475
45424          W. A. SHULL/ 4063 BUTTERWOOD COURT/ DAYTON OH 45424/ (614) 233 -6487
45432          JOE CLMA/ SUITE 200/ SIMULATION TECHNOLOGY INC./ 4124 LINDEN  AVE./ DAYTON OH 45432/ (513) 252-5623
46201          C. W. SAWYER/ MS 1-210/ RCA - CE/ 501 N. LASALLE ST./ INDIANAP OLIS IN 46201/ (317) 267-6802
46202          ATTN: REGENSTRIEF INSTITUTE/ REGENSTRIEF HEALTH CENTER/ 1001 W . TENTH - 5TH FLOOR/ INDIANAPOLIS IN 46202/ (317) 630-6221
46205          RICHARD A. BYERS/ 3690 GLENCAIRN LANE/ INDIANAPOLIS IN 46205
46312          VINCENT ELIAS/ SECURITY FEDERAL S & L ASSN./ 4518 INDIANAPOLIS  BLVD./ EAST CHICAGO IN 46312
46322          PHILIP T. HODGE/ 3102 99TH ST. EAST/ HIGHLAND IN 46322/ (219) 924-5581
46526          IAN SCHMIDT/ 1301 S. MAIN STREET/ GOSHEN IN 46526/ (219) 534-1 794
46805          R. GARY LEE/ DEPT. OF COMPUTER TECHNOLOGY/ PURDUE UNIV./ 2101  COLISEUM BLVD./ FORT WAYNE IN 46805
46808          DALE GAUMER/ GOVT. & INDUSTRIAL DIV./ MAGNAVOX/ 1313 PRODUCTIO N ROAD/ FORT WAYNE IN 46808/ (219) 482-4411
47272          DONALD L. CLAPP/ R. #1/ ST. PAUL IN 47272
47401          ANNA BUCKLEY/ WRUBEL COMPUTING CENTER/ 75K HPER/ INDIANA UNIV. / BLOOMINGTON IN 47401/ (812) 337-1911
47907          KENNETH LEROY ADAMS/ COMPUTING CENTER/ G-148 MATH SCIENCES/ PU RDUE UNIV./ W. LAFAYETTE IN 47907/ (317) 493-9407 OR 494-8232 (WORK)
47907          JOSEPH H. FASEL III/ COMPUTER SCIENCES/ 442 MATH SCIENCES BUIL DING/ PURDUE UNIVERSITY/ W. LAFAYETTE IN 47907/ (317) 493-3832
47907          EDWARD F. GEHRINGER/ DEPT. OF COMPUTER SCIENCE/ MATH SCIENCES  BUILDING/ PURDUE UNIVERSITY/ W. LAFAYETTE IN 47907/ (317) 743-3429
47907          SAUL ROSEN/ COMPUTING CENTER/ G175 MATH SCIENCES BLDG/ PURDUE  UNIV./ W. LAFAYETTE IN 47907/ (317) 494-8235
47907          MICHAEL DEISEMROTH/ SCHOOL OF IND ENGR./ PURDUE UNIV./ W.LAFAY ETTE IN 47907/ (317) 493-3157
48010          SHAUN DEVLIN/ 6854 CEDARBROOK/ BIRMINGHAM MI 48010/ (313) 322- 6856
48033          H. DICK BREIDENBACH/ 4955 PATRICK/ W. BLOOMFIELD MI 48033
48043          ROBERT J. MATHIAS JR/ APT. 2/ 235 CASS AVE./ MT. CLEMENS MI 48 043/ (313) 465-0068
48093          CHRISTOPHER A. PHILLIPS/ 29205 LUND / SOUTH BLDG. APT 14/ WARR EN MI 48093
48098          WESLEY E. MANGUS/ 5786 NORTHFIELD PKWY./ TROY MI 48098
48103          ALAN A. KORTESOJA/ 701 W. DAVIS/ ANN ARBOR MI 48103/ (313) 995 -7063
48103          WILLIAM G. LEDERER/ W. G. LEDERER & ASSOCIATES INC./ 701 S 7TH / ANN ARBOR MI 48103
48103          WILLIAM LUITJE/ 2509 WEST LIBERTY ROAD/ ANN ARBOR MI 48103/ (3 13) 769-7820
48103          LES WARNER/ 1804 LINWOOD/ ANN ARBOR MI 48103
48105          JOHN D. EISENBERG/ 1510 PLYMOUTH RD. #59/ ANN ARBOR MI 48105/  (313) 665-6410 (HOME)/ (313) 453-1400 X 3752 (WORK)
48105          KURT METZGER/ 478 CLOVERDALE/ ANN ARBOR MI 48105/ (313) 662-47 57
48106          DAVID J. WILSON/ ADP NETWORK SERVICES/ 175 JACKSON PLAZA/ ANN ARBOR MI 48106/ (313) 769-6800
48109          PAUL PICKELMANN/ COMPUTING CENTRE/ UNIV. OF MICHIGAN/ 1075 BEA L AVE/ ANN ARBOR MI 48109/ (313) 764-2121
48130          GREG WINTERHALTER/ WINTERHALTER & ASSOC. INC./ 3825 N. ZEEB RO AD/ DEXTER MI 48130/ (313) 426-3029
48169          JOHN S. GOURLAY/ 8645 TOMA ROAD/ PINCKNEY MI 48169/ (313) 994- 6645
48184          DAVID R. POSH/ DEPT 3741/ BURROUGHS CORP./ 3737 S. VENOY RD./  WAYNE MI 48184/ (313) 722-8460 X267
48228          R. NEIL FAIMAN JR./ 8235 APPOLINE/ DETROIT MI 48228/ (313) 834 -3065
48640          BOB METZGER/ COMPUTER TECHNOLOGY DEV./ DOW CHEMICAL CO./ 2040  DOW CENTER/ MIDLAND MI 48640
48824          J. F. P. MARCHAND/ CYCLOTRON LABORATORY/ MICHIGAN STATE UNIV./  EAST LANSING MI 48824
49001          PHILLIP I. GOOD/ 7293 32-2/ THE UPJOHN CO./ KALAMAZOO MI 49001
49003          JAMES H. WALTERS/ AMERICAN NATIONAL HOLDING CO./ P.O. BOX 2769 / KALAMAZOO MI 49003/ (616) 383-6700
49006          LOREN L. HEUN/ THE UPJOHN CO./ 301 HENRIETTA ST./ KALAMAZOO MI  49006/ (616) 385-7886
49008          ATTN: SERIAL RECORDS/ DWIGHT WALDO LIBRARY/ WESTERN MICHIGAN U NIV./ KALAMAZOO MI 49008
49008          MARK C. KERSTETTER/ DEPT. OF MATHEMATICS/ WESTERN MICHIGAN UNI VERSITY/ KALAMAZOO MI 49008/ (616) 383-0959
49085          ANTHONY J. SCHAEFFER/ 1023 VINELAND RD./ ST. JOSEPH MI 49085/  (616) 429-8517
49464          MIKE HAMMAN/ HERMAN MILLER INC./ ZEEMAN MI 49464/ (616) 772-33 00
49503          JOHN DE LONGPRE/ 16 UNION ST. APT. 3/ GRAND RAPIDS MI 49503
49684          TOM LEE/ NORTHWESTERERN MICH. COLLEGE/ 1701 E.FRONT ST/ TRAVER SE CITY MI 49684
49931          JAMES H. HOWARD/ 1113 RUBY/ HOUGHTON MI 49931/ (906) 487-2110
49931          KENNETH M. MCMILLIN/ SIMULATION LAB/ MICHIGAN TECH UNIV./ HOUG HTON MI 49931/ (906) 487-2111
50011          ATTN: ADP CENTER/ 117 PEARSON HALL/ IOWA STATE UNIV./ AMES IA  50011
50158          DAVID HICKOK/ R.A. ENGEL TECH. CENTER/ FISHER CONTROLS CO./ P. O. BOX 11/ MARSHALLTOWN IA 50158/ (515) 754-3011
50307          ATTN: D. M. MOFFETT/ ANNEX/ BANKERS LIFE/ 575 SEVENTH ST./ DES  MOINES IA 50307
51106          ALBERT SHPUNTOFF/ DEPT. OF MATH AND COMPUTER SCIENCE/ MORNINGS IDE COLLEGE/ SIOUX CITY IA 51106/ (712) 274-1184 (HOME)/ (712) 277-5197 (WORK)
52240          LAURA L DICKINSON/ 2107 F ST./ IOWA CITY IA 52240/ (319) 338-9 976
52302          DON STOVER/ 2270 26TH STREET/ MARION IA 52302/ (319) 377-8529
52302          DENNIS SUTHERLAND/ 2835 25TH AVE./ MARION IA 52302/ (319) 377- 4200
52302          T. R. THURMAN/ 1410 7TH ST./ MARION IA 52302/ (319) 395-2280
52402          JAMES C. COZZIE/ 1957B AVE NE/ CEDAR RAPIDS IA 52402
53012          JACK P. SHAW/ W73 N726 LOCUST AVE./ CEDARBURG WI 53012/ (414)  377-4128
53092          S. R. BUCHANAN/ 12613 JONQUIL CT./ MEQUAN WI 53092
53201          THOMAS W. HUEBNER/ 507 E. MICHIGAN ST./ MILWAUKEE WI 53201/ (4 14) 276-9200 X653
53201          GEORGE T. JACOBI/ JOHNSON CONTROLS/ P.O. BOX 423/ MILWAUKEE WI  53201
53202          ATTN: TECHNICAL LIBRARY 47-687/ JOHNSON CONTROLS INC./ 507 E M ICHIGAN ST./ MILWAUKEE WI 53202/ (414) 276-9200 X687
53202          WAYNE CATLETT/ SUITE 335N/ APPLIED COMPUTER DESIGNS INC./ 811  EAST WISCONSIN AVE./ MILWAUKEE WI 53202/ (414) 277-0400
53207          GREGORY JENNINGS/ 3174 S. LOGAN AVE./ MILWAUKEE WI 53207/ (414 ) 483-4972
53217          G. THOMAS SLUSSER/ 5417 N. KENT AVE./ WHITEFISH BAY WI 53217
53218          A. OLDENBURG/ AJAX CORP./ P.O. BOX 18442/ MILWAUKEE WI 53218/  (414) 463-3600
53511          STEPHEN LOCKE/ RESEARCH/ BELOIT CORP./ 1 ST. LAWRENCE AVE./ BE LOIT WI 53511/ (608) 365-3311
53705          EDWARD K. REAM/ 508 FARLEY AVE. - APT. 5/ MADISON WI 53705/ (6 08) 231-2952
53715          O. ARTHUR STIENNON/ PARK-REGENT MEDICAL BLDG./ 1 SOUTH PARK ST ./ MADISON WI 53715/ (608) 255-6262
53927          JAMES E. TARVID/ BOX 20/ DELLWOOD WI 53927/ (608) 339-7259
54601          THOMAS C. HICKS/ 1108 S 5TH. ST./ LA CROSSE WI 54601/ (608) 78 4-4345
54901          AND°EW L. PERRIE/ 1208 BAY SHORE DR./ OSHKOSH WI 54901/ (414)  233-4661 (HOME)/ (414) 424-2068 (WORK)
```

```
55042    CHRIS BOYLAN/ 10711 50TH ST NORTH/ LAKE ELMO MN 55042/ (612) 4 39-0707
55066    TERRY MYHRER/ 1324 EAST AVENUE/ RED WING MN 55066/ (612) 388-4 974
55102    WALT PERKO/ 341 RAMSEY #11/ ST. PAUL MN 55102/ (612) 291-1268
55110    BRIAN HANSON/ 1904 LAKEAIRES BLVD/ WHITE BEAR LA* MN 55110/ (6 12) 429-3400
55113    STEVEN SIEGFRIED/ M.S. 4752/ SPERRY-UNIVAC/ 2276 HIGHCREST DRI VE/ ROSEVILLE MN 55113/ (612) 633-6170 X3667
55113    CHARLES J. PURCELL/ 1260 W. SHRYER AVE./ ST. PAUL MN 55113/ (6 12) 482-2250
55116    MIKE TILLER/ 1239 JUNO AVE/ ST. PAUL MN 55116/ (612) 835-2330
55165    RAYMOND YOUNG/ M.S. U2T18/ SPERRY UNIVAC/ P.O. BOX 3525/ ST. P AUL MN 55165/ (612) 456-5517
55337    BOB ARNOLD/ 3315 BROOKVIEW DR/ BURNSVILLE MN 55337/ (612) 378- 5043 (WORK)/ (612) 894-4307 (HOME)
55343    WILLIAM T. WOOD/ 938 WESTBROOK WAY #7/ HOPKINS MN 55343/ (612)  887-4447
55364    CLARENCE LEHMAN/ 6755 WOODLEDGE RD./ MOUND MN 55364/ (612) 472 -1405
55402    WILLIAM HEILAND/ INQUIRY/ 305 FOSHAY TOWER/ MINNEAPOLIS MN 554 02/ (612) 335-2546
55403    JOEL M. HALPERN/ 1935 GIRARD AVE S./ MINNEAPOLIS MN 55403/ (61 2) 822-5936
55404    EDWARD DEPPE/ DPS INC./ 2550 PILLSBURY AVE. S./ MINNEAPOLIS MN  55404
55404    GLENN FISHBINE/ SUITE 300/ MCPC/ 2344 NICOLLET AVE. S./ MINNEA POLIS MN 55404/ (612) 870-0729
55411    CHARLES WONG/ 2005 HILLSIDE AVE./ MINNEAPOLIS MN 55411/ (612)  378-2441
55414    KIM ADELMAN/ LEE CON INC./ 828 KASOTA AVE./ MINNEAPOLIS MN 554 14/ (612) 378-2500 (BUS)/ (612) 475-3513 (HOME)
55414    THOM HOARD/ P.O. BOX 14113/ MINNEAPOLIS MN 55414/ (612) 376-62 90
55414    JOHN E. LIND/ 515 SE HURON ST./ MINNEAPOLIS MN 55414/ (612) 37 9-9276
55414    JOHN NAUMAN/ 1015 12TH AVE SE #202/ MINNEAPOLIS MN 55414
55414    JON L. SPEAR/ 1007 13TH AVE S.E./ MINNEAPOLIS MN 55414/ (612)  331-1965
55417    GEORGE D. JELATIS/ 3015 E. MINNEHAHA PKWY./ MINNEAPOLIS MN 554 17/ (612) 722-7258
55419    JOHN SCOBEY/ 204 VALLEY VIEW PLACE/ MINNEAPOLIS MN 55419
55435    RON THOMAS/ 7625 BUSH LAKE ROAD/ EDINA MN 55435/ (612) 835-736 1
55435    HAROLD L. BOERLIN II/ SUITE 196/ INFO-DYNE/ 4600 W 77TH ST./ M INNEAPOLIS MN 55435/ (612) 831-5906
55441    DAVID L. SEARLE/ CONTROL DATA CORP./ 2200 BERKSHIRE LANE/ MINN EAPOLIS MN 55441/ (612) 545-2851
55443    WARD SLY/ 8200 TOLEDO AVE. N./ BROOKLYN PARK MN 55443/ (612) 5 66-3928
55454    TIM BONHAM/ D605/1630 S. 6TH ST./ MINNEAPOLIS MN 55454/ (612)  339-4405/ (612) 348-5142 (WORK)
55454    PAUL BRAINERD/ D1605/1630 S. 6TH ST./ MINNEAPOLIS MN 55454/ (6 12) 341-3789
55455    RON ANDERSON/ DEPT. OF SOCIOLOGY/ 1010 SOC SCI TOWER/ UNIV. OF  MINNESOTA/ WEST BANK/ MINNEAPOLIS MN 55455/ (612) 373-0177 (WORK)/ (612) 835-1170 (HOME)
55455    SCOTT S. BERTILSON/ GEOSCIENCE DEPT./ 422 SPACE SCIENCE/ UNIV.  OF MINNESOTA/ MINNEAPOLIS MN 55455/ (612) 373-1994 (WORK)/ (612) 331-2464 (HOME)
55455    DAVE BIANCHI/ UNIVERSITY COMPUTER CENTER/ 227 EXP.ENGR./ U. OF  MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-4181
55455    BRAD BLASING/ UNIVERSITY COMPUTER CENTER/ 227 EXP. ENGR./ U OF  MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 376-5262
55455    JEFFREY J. DRUMMOND/ UNIVERSITY COMPUTER CENTER/ LAUDERDALE/ U  OF MINNESOTA/ MINNEAPOLIS MN 55455/ (612) 376-5603
55455    DANIEL E. GERMANN/ UNIVERSITY COMPUTER CENTER/ 227 EXP ENGR/ U NIV OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 941-1082/ (612) 376-5262 (WORK)
55455    RICK L. MARCUS/ UNIVERSITY COMPUTER CENTER/ 227 EXP ENGR/ UNIV  OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 339-1638/ (612) 373-4181
55455    JAMES F. MINER/ SSRFC/ 25 BLEGEN HALL/ U OF MINNESOTA/ WEST BA NK/ MINNEAPOLIS MN 55455/ (612) 373-5599
55455    STEVEN A. REISMAN/ UNIVERSITY COMPUTER CENTER/ 227 EXP. ENGR./  UNIV OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 376-1755
55812    DAN M. LALIBERTE/ 2015 E 2ND STREET/ DULUTH MN 55812/ (218) 72 8-6177
55901    J. W. MCINTOSH/ IBM/ HIGHWAY 52 & NW 37TH ST./ ROCHESTER MN 55 901/ (507) 286-1011
55987    GERALD W. CICHANOWSKI/ DEPT. COMPUTER SCIENCE/ ST. MARY'S COLL EGE/ P.O. BOX 17/ WINONA MN 55987/ (507) 452-4430 X265
56301    PAUL HELVIG/ 314 4TH AVE. S./ ST. CLOUD MN 56301/ (612) 253-53 52
56301    DAVID L. PETERSON/ 4610 SWEETAIRE STREET/ ST. CLOUD MN 56301/  (612) 253-7548
56320    MIKE STEIN/ 704 4TH STREET NORTH/ COLD SPRING MN 56320/ (612)  685-3710
56381    MIKE CHALENBURG/ BOX 733/ STARBUCK MN 56381
57401    THOMAS L. GERBER/ 419 E. RAILROAD AVE./ ABERDEEN SD 57401/ (60 5) 226-0200
59801    JEFFREY C. JENNINGS/ ROUTE 4/ MISSOULA MT 59801/ (406) 258-537 8
60005    DAVE NUTTING/ 511 W. GOLF RD./ ARLINGTON HTS IL 60005/ (312) 7 56-0710
60010    WILLIAM A. BRIGGS/ 25399 N. BARSUMIAN DR./ TOWER LAKES IL 6001 0/ (312) 948-5500
60025    HUGO HSIUNG/ 1718 ROBIN LANE/ GLENVIEW IL 60025
60053    LOU WARSHAWSKY/ 9220 LINDER/ MORTON GROVE IL 60053/ (312) 965- 1547
60077    JOSEPH LACHMAN/ LACHMAN ASSOCIATES/ 8931 BRONX AVENUE/ SKOKIE  IL 60077/ (312) 674-5685 (WORK)
60106    TIM STEVENS/ 120 E GEORGE ST. NO. 622/ BENSENVILLE IL 60106
60115    LYLE B. SMITH/ COMPUTING SERVICES/ 205A ALTGELD/ NORTHERN ILLI NOIS UNIV./ DEKALB IL 60115/ (815) 753-1059
60120    MARTIN RUNYAN/ ENGINEERING SECTION/ LAKESIDE PRESS/ 920 DAVIS  ROAD/ ELGIN IL 60120/ (312) 697-8310
60174    KEITH GORLAND/ ARTHUR ANDERSEN & CO./ 1405 N. FIFTH AV/ ST. CH ARLES IL 60174
60202    DAVID B. CHRISTIE/ 820 FOREST AVE./ EVANSTON IL 60202/ (312) 3 28-6579 (WORK)
60204    GARY R. GUTH/ MANAGEMENT INFORMATION SYSTEMS/ RUST-OLEUM CORP. / 2301 OAKTON ST./ EVANSTON IL 60204/ (312) 869-1100
60419    STEPHEN PIKE/ 14512 SANDERSON/ DOLTON IL 60419/ (312) 841-6690
60510    ROBERT GOODWIN/ MS 7/ FERMI LAB/ P.O. BOX 500/ BATAVIA IL 6051 0/ (312) 840-4416
60514    HARRIS M. KOEHN/ 6515 S CLARENDON HILLS RD./ CLARENDON HLLS IL  60514
60544    A. L. WOLBERT/ MARINE RESEARCH & DEVELOPMENT/ CHICAGO BRIDGE &  IRON/ ROUTE 59/ PLAINFIELD IL 60544/ (815) 436-2912
60559    STEPHEN HOLLATZ/ 770 OAKWOOD DRIVE/ WESTMONT IL 60559
60603    JAMES E. BECKLEY/ ROAN & GROSSMAN LAW OFFICES/ 120 S. LA SALLE  ST./ CHICAGO IL 60603/ (312) 263-3600
60604    RICHARD A. STACK/ SUITE 1339/ THOMAS L. JACOBS & ASSOCIATES/ 5 3 W. JACKSON BLVD./ CHICAGO IL 60604/ (312) 786-0233
60606    WILLIAM J. NYBACK/ 29 N. WACKER DRIVE/ CHICAGO IL 60606
60613    HENRIETTE KLAWANS/ 3900 N LAKE SHORE DR. NO. 23K/ CHICAGO IL 6  0613
60614    JACK LIEBSCHUTZ/ 626 W BELDEN #2/ CHICAGO IL 60614/ (312) 281-  5827
60618    THOMAS P. HOVEKE/ 3223 W. BERTEAU AVE./ CHICAGO IL 60618
60626    FRANK ALVIANI/ 1327 W. LUNT/ CHICAGO IL 60626
60626    FRANK NUSSBAUM/ DEPT. OF MATH. SCIENCES/ LOYOLA UNIV./ 6525 N.  SHERIDAN/ CHICAGO IL 60626
60626    DAVID J. ZOOK/ 1100 W. PRATT/ CHICAGO IL 60626/ (312) 262-7744
60630    RICHARD E. PRICE/ 5812 W. GIDDINGS/ CHICAGO IL 60630/ (312) 73 6-8618
60657    JAMES KLAJA/ 902 BELMONT/ CHICAGO IL 60657
60658    DAVID C. MADSEN/ 12716 LACROSSE/ ALSIP IL 60658/ (312) 388-679 6
60660    ALAN J. ILIFF/ 1082 W. THORNDALE ST/ CHICAGO IL 60660/ (312) 5 61-4492
60684    THOMAS CORRIGAN/ DEPT. 704-5/ BSC 47-9/ SEARS ROEBUCK AND CO./  SEARS TOWER/ CHICAGO IL 60684
61101    J. MICHAEL SULLIVAN/ J. L. CLARK MFG CO./ 2300 6TH ST./ ROCKFO RD IL 61101
61701    KENNETH L. CHRISTENSEN/ STATE FARM INSURANCE/ 1 STATE FARM PLA ZA A-4/ BLOOMINGTON IL 61701/ (309) 662-2926
61701    SID SMART/ STATE FARM INSURANCE CO./ 1 STATE FARM PLAZA/ BLOOM INGTON IL 61701
61801    RICHARD BALOCCA/ 207 ADVANCED COMPUTATION BLDG/ U OF ILLINOIS/  URBANA IL 61801/ (217) 333-2362
61820    AVRUM ITZKOWITZ/ 205 E. HEALEY #36/ CHAMPAIGN IL 61820/ (217)  359-9644 (HOME)/ (217) 352-6511 (WORK)
62702    MIKE HARRIS/ 407 W. CALHOUN #20/ SPRINGFIELD IL 62702/ (217) 7 89-7669 (HOME)/ (217) 782-0014 (WORK)
62901    F. G. PAGAN/ DEPT. OF COMP. SCI./ SOUTHERN ILLINOIS UNIV./ CAR BONDALE IL 62901
63042    JOHN R. GRINDON/ 853 COACHLIGHT LANE/ ST. LOUIS MO 63042
63105    BOBBY OTIS NASH/ P.O. BOX 16205/ ST. LOUIS MO 63105
63110    GERALD C. JOHNS/ COMPUTER SYSTEMS LAB/ WASHINGTON UNIVERSITY/  724 S. EUCLID AVENUE/ ST. LOUIS MO 63110/ (314) 454-3395
63132    M. W. VANNIER/ 917 FLORADALE/ OLIVETTE MO 63132
63132    MICHAEL A. WHITE III/ LUMBERMATE CO./ 10443 BAUR BLVD./ ST.LOU IS MO 63132/ (314) 991-2004
63166    PETER R. ATHERTON/ DEPT. 112A/ 132 BLDG 2 - LEVEL 1/ MCDONNELL  AIRCRAFT CO./ P.O. BOX 516/ ST. LOUIS MO 63166/ (314) 232-0232
63188    JOHN K. MCCANDLISS/ ALMSA/ ATTN: DRXAL-TL/ P.O. BOX 1578/ ST.  LOUIS MO 63188/ (314) 268-2786
63367    JOHN W. MCCAIN/ NATIONAL SOFTWARE EXCHANGE/ 1000 LAKE ST. LOUI S BLVD. #27/ LAKE ST. LOUIS MO 63367
64108    LARRY D. LANDIS/ UNITED COMPUTING SYSTEMS/ 2525 WASHINGTON/ KA NSAS CITY MO 64108/ (816) 221-9700
64110    JOHN P. CHAPMAN/ 4115 KENWOOD/ KANSAS CITY MO 64110
64110    JOSEPH M. JOYCE/ 5925 ROCKHILL RD./ KANSAS CITY MO 64110/ (816 ) 523-7656
64118    JEFF PALMER/ 2001 NE 52ND TERR./ KANSAS CITY MO 64118/ (816) 4 52-8335
64134    RON TIPTON/ P.O. BOX 9674/ KANSAS CITY MO 64134
65201    FRED DITTRICH/ 705 LYONS/ COLUMBIA MO 65201/ (314) 443-0082
65201    STUART J. KRETCH/ 8 KEENE ST. APT.D-30/ COLUMBIA MO 65201/ (31 4) 449-1952 (HOME)/ (314) 882-2284 (WORK)
66030    A. D. MOORE/ P.O. BOX 364/ GARDNER KS 66030/ (913) 884-8125
66102    DAVID M. ALLAN/ 1317 CENTRAL AVE./ KANSAS CITY KS 66102/ (913)  371-6136 (WORK)/ (913) 381-5588 (HOME)
67460    LYNN MACEY/ ACCK/ 105 E. KANSAS AVE./ MCPHERSON KS 67460
68106    RICHARD WALCH/ SAHLER BUSINESS FORMS CO./ P.O. BOX 6308/ OMAHA  NE 68106
68131    LINDA LEA RAY/ RESEARCH DIVISION/ BOYS TOWN INSTITUTE/ 555 NOR TH 30TH STREET/ OMAHA NE 68131/ (402) 449-6500
68154   DARYL E. MALENA/ SUITE 8/ 10838 OLD MILL ROAD/ OMAHA NE 68154/   (402) 330-4100
68182    LYNNE J. BALDWIN/ DEPT. OF MATH/COMP. SCI./ U OF NEBRASKA/ 60T H AND DODGE/ OMAHA NE 68182/ (402) 554-2836
68588    ATTN: INFORMATION/RESOURCE CENTER/ 225 NEBR. HALL/ UNIVERSITY  OF NEBRASKA/ LINCOLN NE 68588/ (402) 472-3701
69341    GARY J. BOOS/ 2350 CHATEAU WAY/ GERING NE 69341
72205    JOHN L. CHANEY/ 615 N COOLIDGE ST./ LITTLE ROCK AR 72205
72554    DAN REED/ BOX 22/ MAMMOTH SPRING AR 72554/ (501) 625-3653
72701    JOSEPH N. HILTON/ RT. 5 BOX 385/ FAYETTEVILLE AR 72701/ (501)  443-2045
73070    MIKE O'DELL/ COMP. SCI. DEPT./ UNIV. OF OKLAHOMA/ P.O. BOX 289 1/ NORMAN OK 73070/ (405) 325-4721
73106    DAVID HUSNIAN/ 1731 N.W. 29TH/ OKLAHOMA CITY OK 73106/ (405) 5 25-7042
73501    GARY HUCKABAY/ DEPT. OF MATHEMATICS/ CAMERON UNIV./ LAWTON OK  73501/ (405) 248-2200 X49
74102    KENNETH R. DRIESSEL/ AMOCO RESEARCH/ P.O. BOX 591/ TULSA OK 74 102/ (418) 644-3551
74145    BOB DUPREE/ 4849 S. 69TH EAST AVE./ TULSA OK 74145/ (918) 663- 4646
74145    CONRAD SUECHTING/ DATA GENERAL CORP./ 9726 E. 42ND ST. SUITE 2 00/ TULSA OK 74145/ (918) 664-8530
75006    JOHN JENKINSON/ BOX 169 MS 32/ MOSTEK/ 1215 W. CROSBY/ CARROLL TON TX 75006/ (214) 242-0444 X2401
75042    ERIC PEABODY/ 2126 HOMESTEAD PLACE/ GARLAND TX 75042/ (214) 49 5-6416/ (214) 238-5936
75042    JOE C. ROBERTS/ 1529 MEADOWCREST/ GARLAND TX 75042/ (214) 238- 3711 (WORK)/ (214) 276-8157 (HOME)
75043    LARRY WEISS/ 524 GRANADA DR./ GARLAND TX 75043
75080    E. J. SAMMONS/ M/S 410-260/ ROCKWELL INTERNATIONAL/ 1200 N. AL MA/ RICHARDSON TX 75080/ (214) 996-3593
75081    JOHN P. HARVELL/ ADVANCED TECHNOLOGY 420-150/ ROCKWELL INTERNA TIONAL/ 1200 N. ALMA ROAD/ RICHARDSON TX 75081/ (214) 996-2280
75088    CLEMENT MORITZ/ NORTHEAST SCIENTIFIC CORP./ 7518 MERRITT RD/ R OWLETT TX 75088/ (214) 475-1164
75116    STANLEY E. BAMMEL/ BAMMEL SOFTWARE ENGINEERING/ 1307 W RIDGE/  DUNCANVILLE TX 75116/ (214) 298-6870
75205    CARL J. TOSETTO/ P.O. BOX 8445/ DALLAS TX 75205/ (214) 824-237 8
75221    WENDEL WHEELER/ TAYLOR PUBLISHING CO./ P.O. BOX 597/ DALLAS TX  75221
75229    CHARLIE SCOGIN/ UNISYSTEMS SERVICES/ 2840 WALNUT HILL LANE/ DA LLAS TX 75229/ (214) 350-6658
75231    KIM L. SHIVELEY/ 7777 MANDERVILLE LANE APT. 221/ DALLAS TX 752 31/ (214) 363-5249
75234    DAVID E. BREEDING/ HARRIS DATA COMM DIV/ P.O. BOX 400010/ DALL AS TX 75234/ (214) 386-2296
```

```
75236  ATTENTION: FRED BEVENSEE/ SUMMER INSTITUTE OF LINGUISTICS/ 750 0 W. CAMP WISDOM RD./ DALLAS TX 75236/ (214) 298-3331
75961  JESSE D. MIXON/ DEPT. OF ACCOUNTING/ STEPHEN F. AUSTIN STATE U NIV/ P.O. BOX 3005 SFA STATION/ NACOGDOCHES TX 75961/ (713) 569-3105/ (713) 564-4909
76019  JOHN M. HEMPHILL/ COMP. SCI. & ENGR./ UNIV OF TEXAS AT ARLINGT ON/ BOX 19015/ ARLINGTON TX 76019/ (817) 273-3785
76059  STEVE CAVENDER/ COMPUTER SERVICES/ SOUTHWESTERN ADVENTIST COLL ./ KEENE TX 76059/ (817) 645-3921
76101  ROBERT L. TURPIN/ TEXAS ELECTRIC SERVICE CO./ P.O. BOX 970/ FO RT WORTH TX 76101
76107  DAVID P. MCDONNELL/ 4912 GEDDES/ FT. WORTH TX 76107/ (817) 738 -1884
76133  CHARLES F. SHELON/ 3629 FENTON AVE./ FORT WORTH TX 76133
77001  ATTENTION: COLIN G. CAMPBELL/ MS / 781/ TEXAS INSTRUMENTS/ P.O . BOX 1444/ HOUSTON TX 77001/ (713) 491-5115 X3338
77001  S. BALASUBRAMANIAN/ SHELL DEVELOPMENT COMPANY/ PO BOX 481/ HOU STON TX 77001/ (713) 633-2335
77001  DAVID DYCHE/ MS 781/ TEXAS INSTRUMENTS/ P.O. BOX 1444/ HOUSTON  TX 77001/ (713) 491-5115 X3335
77001  GINGER KELLY/ INSTITUTE FOR COMP. SERV. AND APP./ RICE UNIVERS ITY/ P.O. BOX 1892/ HOUSTON TX 77001/ (713) 527-4965
77023  HOWARD LEVERENZ/ 4723 B MCKINNEY/ HOUSTON TX 77023/ (713) 926- 5922
77024  RICHARD A. KOEBBING/ B. J. KAHN & ASSOCIATES/ P.O. BOX 19437/  HOUSTON TX 77024
77024  JOHN B. WARDLAW/ 13935 BARRYKNOLL LANE/ HOUSTON TX 77024/ (713 ) 497-4811
77043  JOHN EARL CRIDER/ 2918 KEVIN LANE/ HOUSTON TX 77043/ (713) 462 -0299
77055  FELIX S. H. LI/ PENSION MATHEMATICS/ 14237 MISTY MEADOW LANE/  HOUSTON TX 77055/ (713) 466-7521
77074  WAYNE FLOURNOY/ 8500 NAIRN NO. 318/ HOUSTON TX 77074/ (713) 77 6-1937
77079  DAVID HOLLAND/ SUITE 200/ INTERCOMP/ 1201 DAIRY ASHFORD/ HOUST ON TX 77079
77081  JAMES S. HUGGINS/ 5920 BISSONET #45/ HOUSTON TX 77081
77302  THOMAS K. BURGESS/ 8825 SOUTH I-45/ CONROE TX 77302/ (713) 756 -5160
78148  JOHN E. NEWTON/ AFMPC/MPCDDP5/ RANDOLPH AFB TX 78148
78231  GENE HUGHES/ 2907 SKY CLIFF/ SAN ANTONIO TX 78231/ (512) 492-9 661
78363  ATTN: COMPUTATION CENTER/ TEXAS A & I UNIVERSITY/ CAMPUS BOX 1 85/ KINGSVILLE TX 78363
78664  KATIE NOONING/ 1105 DEEPWOOD/ ROUND ROCK TX 78664/ (512) 255-6 052
78712  WILHELM BURGER/ DEPT. OF COMPUTER SCIENCES/ 328 PAINTER HALL/  UNIV. OF TEXAS - AUSTIN/ AUSTIN TX 78712/ (512) 471-4353
78712  ALAN ZARING/ COMPUTER SCI. DEPT./ UNIV. OF TEXAS - AUSTIN/ AUS TIN TX 78712/ (512) 471-7316
78721  DONALD G. WEISS/ H2565/ 3501 ED BLUESTEIN BLVD./ AUSTIN TX 787 21/ (512) 928-6034
78753  ROGER W. FRECH/ 10033 CHILDRESS DR./ AUSTIN TX 78753/ (512) 83 7-6078
78758  WALT FEESER/ 8900 SHOAL CREEK SUITE 109/ AUSTIN TX 78758
78758  FREDERICK JOHN TYDEMAN/ 8405 BOWLING GREEN/ AUSTIN TX 78758/ ( 512) 454-9292
78769  DAVID N. GRAY/ MS 2201/ TEXAS INSTRUMENTS/ P.O. BOX 2909/ AUST IN TX 78769/ (512) 451-8441 X269
78873  JEFFREY W. GRAHAM/ GRAHAM COMPUTER ENTERPRISES INC./ P.O. BOX  634/ LEAKEY TX 78873
79109  HARRY P. HAIDUK/ 6202 MCCOY/ AMARILLO TX 79109/ (806) 376-5111  X356 (WORK)/ (806) 352-1811 (HOME)
79601  D. A. CAUGHFIELD/ 609 E. N. 21ST/ ABILENE TX 79601/ (915) 677- 1911
80027  THOMAS L. LIGHT/ STORAGE TECHNOLOGY CORP./ 2270 S 88TH ST./ LO UISVILLE CO 80027/ (303) 666-6581
80027  GEORGE H. RICHMOND/ MAIL DROP FF/ STORAGE TECHNOLOGY CORP./ LO UISVILLE CO 80027/ (303) 497-5151 X7416
80201  JOHN CARNAL/ MS 0424/ MARTIN MARIETTA/ P.O. BOX 179/ DENVER CO  80201
80201  BILL EHLERT/ P.O. BOX 3154/ DENVER CO 80201
80202  TERRY R. ROBERTS/ SECURITY LIFE OF DENVER/ 1616 GLENARM PLACE/  DENVER CO 80202/ (303) 534-1861
80203  DAVID HORNBAKER/ 1351 WASHINGTON/ DENVER CO 80203
80203  FRED KATZMAN/ INFORMATION SYSTEMS/ MATHEMATICA POLICY RESEARCH  INC./ 1410 GRANT STREET/ DENVER CO 80203/ (303) 837-1500
80210  R. L. ESHELMAN/ 2525 BUCHTEL BLVD./ DENVER CO 80210/ (303) 322 -0494
80231  JEAN TROUDT/ 900 S. QUINCE ST. #921/ DENVER CO 80231/ (303) 32 0-1959
80234  RON OLSEN/ ROOM 1J28/ BELL LABORATORIES/ 11900 N. PECOS ST./ D ENVER CO 80234/ (303) 451-4224
80301  DAVID ANDRUS/ KRYPTONICS INC/ 5660 CENTRAL AVENUE/ BOULDER CO  80301/ (303) 442-9173
80301  CHARLES R. PRICE/ KNUTSON ASSOCIATES INC./ 1700 N. 55TH ST./ B OULDER CO 80301/ (303) 449-0574
80302  DAVID PICKENS/ 50R / 023/ IBM CORP./ P.O. BOX 1900/ BOULDER CO  80302/ (303) 447-5844
80302  JAY SCHUMACHER/ MONOLITHIC SYSTEMS/ 1466 13TH ST./ BOULDER CO  80302
80302  JOE WATKINS/ 2895 18TH STREET/ BOULDER CO 80302/ (303) 443-859 8/ (303) 234-3115 (WORK)
80303  DENNIS R. ELLIS/ C/O CRAY RESEARCH/ 75 MANHATTAN DR. - SUITE # 3/ BOULDER CO 80303/ (303) 494-5151 X585
80306  RICHARD L. ANDERSON/ COMPUTER SERVICES CENTER/ BOULDER VALLEY  PUBLIC SCHOOLS/ P.O. BOX 9011/ BOULDER CO 80306/ (303) 447-8153
80401  THERON D. CARLSON/ 472 GLADIOLA/ GOLDEN CO 80401/ (303) 278-24 40
80537  JEFF EASTMAN/ DESKTOP COMPUTER DIV./ HEWLETT PACKARD/ P.O. BOX  301/ LOVELAND CO 80537/ (303) 667-5000
80639  ATTENTION: D. L. MYERS/ UNC COMP. CENTER/ UNIV. OF NORTHERN CO LORADO/ GREELEY CO 80639
80917  THOMAS W. LAWHORN/ SUITE 202/ CIBAR INC./ 2850 SERENDIPITY CIR ./ COLORADO SPRGS CO 80917/ (303) 574-4050
83702  JOHN E. VAN DEUSEN III/ 61 HORIZON DR./ BOISE ID 83702/ (208)  342-1464
83720  DAN HOMER/ INDUSTRIAL COMMISSION/ STATE OF IDAHO/ 317 MAIN ST. / BOISE ID 83720
84010  GORDON W. ROMNEY/ 1141 OAKRIDGE LANE/ BOUNTIFUL UT 84010/ (801 ) 292-9813
84014  MYRON R. SYPHUS/ 79 W 625 N/ CENTERVILLE UT 84014/ (801) 292-2 043
84105  JOE B. BRAME JR./ 1403 REDONDO AV/ SALT LAKE CITY UT 84105/ (8 01) 539-5603
84108  J. D. CALLAHAN/ KELON CORP./ P.O. BOX 8275/ SALT LAKE CITY UT  84108/ (801) 582-6313
84109  T. M. MALIN/ 3445 S. MILLCREEK RD./ SALT LAKE CITY UT 84109
84116  RICHARD G. LYMAN/ SPERRY UNIVAC/ 322 NORTH 2200 WEST/ SALT LAK E CITY UT 84116/ (801) 328-8066
84121  W. F. HAYGOOD/ COMPUTER SERVICES CO./ 7822 OAKLEDGE ROAD/ SALT  LAKE CITY UT 84121/ (801) 942-2300
84602  PARLEY P. ROBINSON/ COMPUTER SERVICES/ 403 CB/ BRIGHAM YOUNG U NIV./ PROVO UT 84602/ (801) 374-1211 X3681
84737  L. PAINTER/ ELGENCO INC./ BOX 460/ HURRICANE UT 84737
85002  DAVID CALCATELLI/ MAIL STATION D4/ SPERRY FLIGHT SYSTEMS/ P.O.  BOX 21111/ PHOENIX AZ 85002/ (602) 942-2311 X1222
85008  GENE A. SUMNER/ 4739 E. LEWIS AVE/ PHOENIX AZ 85008
85028  AUTHOR R. JETER/ 3946 EAST ALTADENA/ PHOENIX AZ 85028
85061  FRANK ANDERSON/ COMPUTER SERVICES/ GRAND CANYON COLLEGE/ 3300  WEST CAMELBACK RD./ PHOENIX AZ 85061/ (602) 249-3300
85260  DENNIS KODIMER/ TERAK CORPORATION/ 14405 N. SCOTTSDALE RD./ SC OTTSDALE AZ 85260/ (602) 991-1580
85352  ARDEN WOOTTON/ P.O. BOX 329/ TACNA AZ 85352/ (602) 785-4128
85613  E. W. ERRICKSON/ U.S. ARMY INTELLIGENCE CENTER/ P.O. BOX 596/  FT HUACHUCA AZ 85613/ (602) 246-6910
85721  DAVID R. HANSON/ DEPT. OF COMP. SCIENCE/ UNIV. OF ARIZONA/ TUC SON AZ 85721/ (602) 626-3617
85721  GREGG TOWNSEND/ COMP. CENTER/ UNIV. OF ARIZONA/ TUCSON AZ 8572 1/ (602) 626-2441
87107  ROBERT W. BERRY/ 601 SANDIA RD NW/ ALBUQUERQUE NM 87107/ (505)  344-7219
87107  JOHN J CORCORAN 3RD. III/ 557 MISSION NE/ ALBUQUERQUE NM 87107 / (505) 345-1309
87108  STEPHEN C. WOOD/ MICROSOFT/ 300 SAN MATEO NE SUITE 819/ ALBUQU ERQUE NM 87108/ (505) 262-1486
87112  SERGIO BERNSTEIN/ DATA DIV./ BERNE ELECTRONICS INC./ 1300 MURI EL NE/ ALBUQUERQUE NM 87112/ (505) 293-3611
87117  ATTN: AIR FORCE WEAPONS LABORATORY/ DYV (HARRY M. MURPHY JR.)/  KIRTLAND AFB NM 87117/ (505) 264-9317
87131  KARL LIEBERHERR/ 147 COMP. AND INFO. SCI. LIBRARY/ UNIVERSITY  OF NEW MEXICO/ ALBUQUERQUE NM 87131
87544  T. J. COOK/ P.O. BOX 248/ LOS ALAMOS NM 87544
87544  ORVAL F. HART JR/ 406 GRAND CANYON DR./ LOS ALAMOS NM 87544/ ( 505) 667-7847 (WORK)/ (505) 672-1353 (HOME)
87545  SUE JOHNSON/ MS 532 L-10/ LOS ALAMOS SCIENTIFIC LAB/ LOS ALAMO S NM 87545/ (505) 667-5065
87545  JOHN MONTAGUE/ GROUP C11/ MAIL STOP 296/ LOS ALAMOS SCIENTIFIC  LABORATORY/ LOS ALAMOS NM 87545/ (505) 667-7158
87545  BOB SCARLETT/ GROUP L 10 - MS 532/ LOS ALAMOS SCIENTIFIC LABOR ATORY/ P.O. BOX 1663/ LOS ALAMOS NM 87545/ (505) 667-5827
88130  JOSEPH R. FALKNER/ DEPT. OF COMP. SCI. & STAT./ EASTERN NEW ME XICO UNIV./ PORTALES NM 88130/ (505) 356-4685
89119  RONALD L. YOUNG/ 7456 S. SPENCER ST./ LAS VEGAS NV 89119/ (702 ) 361-6631
89502  LOUIS N. BELEOS/ 920 CORDONE AVE./ RENO NV 89502
89503  DARRYL KUHNS/ 1590 HILLSIDE DR./ RENO NV 89503/ (702) 786-1759
89511  WILLIAM R. BONHAM/ SIERRA DIGITAL SYSTEMS/ 13905 RANCHEROS DRI VE/ RENO NV 89511/ (702) 329-9548
90010  STEVEN J. GREENFIELD/ UNICORN SYSTEMS CO./ 3807 WILSHIRE BLVD. / LOS ANGELES CA 90010/ (213) 380-6974
90019  RAYMOND QUIRING/ B. H. INDUSTRIES/ 5784 VENICE BLVD./ LOS ANGE LES CA 90019/ (213) 937-4763
90024  ERIC PUGH/ 412 LANDFAIR AVE. #2/ LOS ANGELES CA 90024/ (213) 4 79-1352
90045  ATTENTION: LARRY LEWIS/ FUTUREDATA COMPUTER CORP./ 11205 SO. L A CIENEGA BLVD./ LOS ANGELES CA 90045/ (213) 641-7700
90045  R. L. MERCER/ SCAN-TRON CORP./ 8820 S.SEPULVEDA BLVD/P.O. BOX  45706/ LOS ANGELES CA 90045
90045  PHILIP H. SAYRE/ DELPHI COMMUNICATIONS CORP./ 11220 HINDRY AVE ./ LOS ANGELES CA 90045/ (213) 670-2040
90045  PHILIP A. WASSON/ 9513 HINDRY PLACE/ LOS ANGELES CA 90045/ (21 3) 649-1428
90046  DAVID YOST/ 8464 1/2 KIRKWOOD DR./ HOLLYWOOD CA 90046/ (213) 6 56-9820
90049  LEE A. BENBROOKS/ P.O. BOX 49208/ LOS ANGELES CA 90049/ (213)  472-1165
90064  DAVID G. CLEMANS/ 2830 SEPALVEDA #20/ LOS ANGELES CA 90064/ (2 13) 473-7961
90065  LYNN BLICKENSTAFF/ SELF-REALIZATION FELLOWSHIP/ 3880 SAN RAFAE L AVE./ LOS ANGELES CA 90065
90066  GILL LYTTON/ 4315 KENYON AVE/ LOS ANGELES CA 90066
90069  TERRENCE M. CASELLA/ 9009 ELEVADO AVE./ LOS ANGELES CA 90069/  (213) 272-1387
90230  JOHN MCMANUS JR./ SYSTEMS DEPT./ METRO-GOLDWYN-MAYER INC./ 102 02 W. WASHINGTON BLVD./ CULVER CITY CA 90230
90240  JOHN TROTTER/ ARTISAN SOFTWARE/ 6429 DOS RIOS RD./ DOWNEY CA 9 0240/ (213) 928-3742
90249  JUDITH MINAMIJI/ 15419 CIMARRON STREET/ GARDENA CA 90249/ (213 ) 324-4634
90254  MARC HANSON/ 621 25TH ST/ HERMOSA BEACH CA 90254
90260  ROBERT ALKIRE/ 4450 W. 165TH ST./ LAWNDALE CA 90260/ (213) 370 -9392
90266  HERBERT E. MORRISON/ 1257 2ND STREET/ MANHATTAN BCH CA 90266/  (213) 887-2571 (DAYS)
90266  STUART C. NIMS/ 3605 PINE AVENUE/ MANHATTAN BCH CA 90266
90274  W. A. KELLEY/ 46 ROLLINGWOOD/ R. HILLS EST. CA 90274
90277  PHYLLIS A. REILLY/ P.O. BOX 3613/ REDONDO BEACH CA 90277/ (213 ) 535-2450 (WORK)/ (213) 637-7989 (HOME)
90278  JAMES L. AGIN/ 2178 BLD. 90/ TRW-DSSG/ ONE SPACE PARK/ REDONDO  BEACH CA 90278/ (213) 535-0312
90278  JERRY F. BRUMBLE/ MI1409/ TRW DSSG/ ONE SPACE PARK/ REDONDO BE ACH CA 90278/ (213) 536-3546
90278  JAY SAX/ 90-2178/ TRW - SYSTEMS GROUP/ ONE SPACE PARK/ REDONDO  BEACH CA 90278/ (213) 392-6372 (HOME)/ (213) 535-0312 (WORK)
90291  STEVE DALSIMER/ 13044 MINDANAO WAY NO. 6/ MARINA DEL REY CA 90 291
90291  PATRICK D. GARVEY/ M-1077/ 7740 REDLANDS ST./ PLAYA DEL RAY CA  90291
90401  DICK HEISEN/ THE COMPUTER STORE/ 820 BROADWAY/ SANTA MONICA CA  90401/ (213) 451-0713
90403  GUS BACOYANIS/ 1007 20TH ST. - APT 5/ SANTA MONICA CA 90403/ ( 213) 395-1742
90403  P. U. GEORGE/ 817 1/2 16TH ST./ SANTA MONICA CA 90403
90404  ATTN: LARRY MARKWORTH - LIBRARIAN/ PACIFIC SIERRA RESEARCH COR P./ 1456 CLOVERFIELD BLVD./ SANTA MONICA CA 90404/ (213) 828-7461
90405  CHARLES SISKA JR./ 2021 10TH ST./ SANTA MONICA CA 90405/ (213)  396-2111
90406  RICHARD J. KWAN/ MAIL DROP 41-25/ SYSTEM DEVELOPMENT CORP./ 25 00 COLORADO AVE./ SANTA MONICA CA 90406/ (213) 829-7511 X3223
90406  D. LLOYD RICE/ COMPUTALKER CONSULTANTS/ P.O. BOX 1951/ SANTA M ONICA CA 90406/ (213) 392-5230
90630  JOHN L. PRUN/ 4475 CASA GRANDE CIRCLE NO. 145/ CYPRESS CA 9063 0/ (714) 821-3744
90631  R. C. THORNTON/ CHEVRON OIL FIELD RESEARCH CO./ P.O. BOX 446/  LA HABRA CA 90631
90712  CHARLES R. BEAUREGARD/ 4728 MAYBANK AVE./ LAKEWOOD CA 90712
90732  MYRON C. LONG/ 2145 AVENIDA APRENDA/ SAN PEDRO CA 90732/ (213)  548-3746
90801  ATTN: AMERICAN COMPUTER SERVICES/ P.O. BOX 2651/ LONG BEACH CA  90801
```

```
90802   DONALD H. MCCLELLAND/ UNIV. INVESTMENT MANAGEMENT CO./ 666 E. OCEAN BLVD. #3101/ LONG BEACH CA 90802/ (213) 435-6344
90815   KENNETH K. IWASHIKA/ 6934 MANTOVA ST./ LONG BEACH CA 90815/ (213) 596-7336
91011   JOHN J. WEDEL/ P.O. BOX 146/ LA CANADA CA 91011/ (213) 354-4059
91020   WERNER G. MATTSON/ P.O. BOX 621/ MONTROSE CA 91020
91030   H. LASHLEE/ P.O. BOX 987/ S. PASADENA CA 91030
91030   R. S. SCHLAIFER/ 1500 ROLLIN/ S. PASADENA CA 91030/ (213) 354-5115
91101   GURUPREM SINGH KHALSA/ KHALSA COMPUTER SYSTEMS INC./ 500 SOUTH LAKE AVENUE/ PASADENA CA 91101/ (213) 684-3311
91103   NICK COPPING/ JET PROPULSION LABS/ MS 169/332/ CALIF. INST. OF TECHNOLOGY/ 4800 OAK GROVE DR./ PASADENA CA 91103/ (213) 354-4321
91103   LARRY HAWLEY/ MS 238-218/ JET PROPULSION LABORATORY/ 4800 OAK GROVE DR./ PASADENA CA 91103/ (213) 354-2551
91103   W. O. PAINE/ MS 83-205/ JET PROPULSION LAB./ 4800 OAK GROVE DR./ PASADENA CA 91103/ (213) 354-4284
91103   STEPHEN SKEDZELESKI/ 198-136/ JET PROPULSION LAB/ 4800 OAK GROVE DR./ PASADENA CA 91103
91105   JAMES T. HERINGER/ 440 GLENULLEN DR./ PASADENA CA 91105/ (213) 257-3853
91105   HOWARD RUMSEY JR./ 151 LINDA VISTA AVE./ PASADENA CA 91105/ (213) 795-1260
91125   KARL FRYXELL/ DIVISION OF BIOLOGY/ 216-76/ CALIFORNIA INST. OF TECH./ PASADENA CA 91125/ (213) 795-6811 X2827
91126   SHAL FARLEY/ PAGE HOUSE/ CALTECH/ PASADENA CA 91126/ (213) 796-5974
91301   HARRY S. ADAMS/ P.O. BOX 70/ AGOURA CA 91301/ (213) 889-1094
91311   EDDIE CARRIE/ PERTEC COMPUTER CORP./ 20630 NORDHOFF ST./ CHATSWORTH CA 91311/ (213) 988-1800
91320   CARROLL HENNICK/ 127 DEVIA DR./ NEWBURY PARK CA 91320
91326   CHARLES RIDER/ 19100 KILLOCH WAY/ NORTHRIDGE CA 91326/ (213) 360-3254
91335   CATHERINE C. TOBEY/ 8020-3 CANBY AVE./ RESEDA CA 91335
91342   EUGENE P. MONTGOMERY/ 15721 EL CAJON ST./ SYLMAR CA 91342/ (213) 367-8101
91343   BRUCE S. SEELY/ 8545-K BURNET AVE./ SEPULVEDA CA 91343/ (213) 894-0029
91359   P. & C.F.BLOMKE CHANG/ ELECTRONIC SYSTEMS DIV./ BUNKER-RAMO/ P.O. BOX 5009/ WESTLAKE VILL* CA 91359
91360   ELIZABETH IBARRA/ 605 RIO GRANDE/ THOUSAND OAKS CA 91360/ (805) 488-4425
91364   JIM FOLEY/ MARKETRON/ 21031 VENTURA BLVD. SUITE 1020/ WOODLAND HILLS CA 91364/ (213) 347-6400
91364   CAM WATSON/ ZETA SYSTEMS/ 6430 VARIEL AVE./ WOODLAND HILLS CA 91364
91367   GEORGE MASSAR SR/ 6225-102 SHOUP AVE./ WOODLAND HILLS CA 91367/ (213) 346-1883/ (213) 377-4811 (WORK)
91409   HERMAN FISCHER/ LITTON DATA SYSTEMS/ 8000 WOODLEY AVE./ VAN NUYS CA 91409/ (213) 781-8211 X 4213
91711   LEE D. AURICH/ 473 BLAISDELL DR./ CLAREMONT CA 91711
92008   JOEL MCCORMACK/ 1731 CATALPA RD./ CARLSBAD CA 92008
92024   CHARLES O. GIMBER/ 817 CREST DR./ ENCINITAS CA 92024/ (714) 942-0754
92027   JAMES A. DARLING/ 1920 E. GRAND AVE #39/ ESCONDIDO CA 92027/ (714) 741-4921
92041   RAJ MALHOTRA/ RJ SOFTWARE SYSTEMS/ 7471 UNIV. AVE./ LA MESA CA 92041/ (800) 854-2751/ (800) 552-8820
92067   KEN BOWLES/ P.O. BOX 1123/ RANCHO SANTAFE CA 92067/ (714) 755-7288/ 452-4526
92093   BOB HOFKIN/ APIS DEPT. C-014/ UNIV. OF CALIFORNIA-SAN DIEGO/ LA JOLLA CA 92093/ (714) 452-4526
92093   JIM MADDEN/ C-010 COMPUTER CENTER/ UNIV. OF CALIFORNIA - SAN DIEGO/ LA JOLLA CA 92093/ (714) 452-4050
92103   DAN RICHMOND/ 1670 LINWOOD ST./ SAN DIEGO CA 92103/ (714) 295-5949
92110   MARY K. LANDAVER/ 2677 COWLEY WAY/ SAN DIEGO CA 92110/ (714) 275-3029
92110   JOHN LOWRY/ DEFENSE COMMUNICATIONS DIVISION/ ITT/ 4250 PACIFIC HWY #224/ SAN DIEGO CA 92110/ (714) 226-8735
92111   WEBB SIMMONS/ HORIZON TECHNOLOGY/ 7830 CLAIREMONT MESA BLVD./ SAN DIEGO CA 92111/ (724) 292-8331
92121   F. ANTONIO/ UNIVERSITY INDUSTRIAL PARK/ LINKABIT CORP./ 10453 ROSELLE ST./ SAN DIEGO CA 92121/ (714) 453-7007
92121   D. L. KNITTEL/ DIGITAL SCIENTIFIC CO./ 11425 SORRENTO VALLEY RD./ SAN DIEGO CA 92121/ (714) 453-6050
92123   DAVID M. BULMAN/ PRAGMATICS INC./ 3032 MASTERS PL./ SAN DIEGO CA 92123/ (714) 565-0565
92123   ROGER H. EVANS/ INTEROCEAN SYSTEMS INC/ 3540 AERO CT./ SAN DIEGO CA 92123/ (714) 565-8400
92127   JOSEPH W. SMITH/ MS 8401/ NCR/ 16550 WEST BERNARDO DR./ SAN DIEGO CA 92127/ (714) 485-2864
92138   W. E. CLARK/ DEPT. 244/ P.O. BOX 80158/ SAN DIEGO CA 92138/ (714) 455-1330 X302
92138   ROBERT J. REYNOLDS/ MAIL ZONE 32-6040/ GENERAL DYNAMICS/CONVAIR DIV./ P.O. BOX 80847/ SAN DIEGO CA 92138/ (714) 277-8900 X2435
92138   CRAIG A. SNOW/ TRW COMMUNICATION SYSTEMS/ BOX 80157/ SAN DIEGO CA 92138/ (714) 453-5303
92138   CLARK F. WAITE/ DATA SYSTEMS SERVICES/ MZ 43-5310/ GENERAL DYNAMICS/ P.O. BOX 8084/ SAN DIEGO CA 92138/ (714) 277-8900
92152   MICHAEL S. BALL/ CODE 632/ NAVAL OCEAN SYSTEMS CENTER/ SAN DIEGO CA 92152/ (714) 225-2366
92182   V. VINGE/ DEPT. OF MATHEMATICAL SCI./ SAN DIEGE STATE UNIV./ SAN DIEGO CA 92182/ (714) 286-6697/ (714) 286-6191
92408   TED C. PARK/ SYSTEMS DEVELOPMENT/ SUITE 105/ MEDICAL DATA CONSULTANTS/ 114 AIRPORT DRIVE/ SAN BERNARDINO CA 92408/ (714) 825-2683
92521   ALICE HUNT/ COMPUTING CENTER/ UNIV. OF CALIFORNIA - RIVERSIDE/ RIVERSIDE CA 92521
92627   DENNIS F. KIBLER/ 160 21ST ST. APT. A/ COSTA MESA CA 92627/ (714) 548-4098
92627   TIM LOWERY/ 2653 SANTA ANA AVE./ COSTA MESA CA 92627/ (714) 631-0771
92630   THOMAS J. PAULSON/ 23251 LOS ALISOS #70/ EL TORO CA 92630/ (714) 586-2802
92630   JAMES P. URONE/ 22705 MALAGA WAY/ EL TORO CA 92630/ (714) 768-4743
92631   GARY B. SHELLY/ ANAHEIM PUBLISHING CO./ 120 E ASH/ FULLERTON CA 92631
92634   THOMAS M. NEAL/ BECKMAN INSTRUMENTS/ 2500 HARBOR BLVD./ FULLERTON CA 92634/ (714) 871-4848 X 3259
92635   FRANK F. CRANDELL/ 3008 MAPLE AVE./ FULLERTON CA 92635
92646   BARCLAY R. KNERR/ 9061 CHRISTINE DRIVE/ HUNTINGTON BCH CA 92646/ (714) 633-4013
92651   GENE FISHER/ 346 CANYON ACRES DR./ LAGUNA BEACH CA 92651/ (714) 497-1241
92660   JIM SQUIRES/ 457 BAYWOOD DR./ NEWPORT BEACH CA 92660
92675   R. L. WALLACE/ 26501 CAMPESINO/ MISSION VIEJO CA 92675/ (714) 831-3127
92691   ROBERT L. JARDINE/ BURROUGHS CORP./ 25725 JERONIMO ROAD/ MISSION VIEJO CA 92691/ (714) 768-2370
92704   WILLIAM E. CROSBY/ 2701 S. FAIRVIEW ST. #R1/ SANTA ANA CA 92704/ (714) 549-7640
92704   JOHN URBANSKI/ CONTROL DATA CORP./ 3519 WEST WARNER/ SANTA ANA CA 92704/ (714) 754-4060/ (612) 373-3608 (WORK)
92713   DONALD D. PECKHAM/ PERTEC COMPUTER CORP./ P.O. BOX 19602/ IRVINE CA 92713/ (714) 540-8340 X306
92714   ALEX BRADLEY/ STANDARD SOFTWARE SYSTEMS/ 17931 'J' SKY PARK/ IRVINE CA 92714/ (714) 540-8445
92714   RUDY L. FOLDEN/ OPERATING SYSTEMS DEVELOPMENT/ P.O. BOX C-19504/ SPERRY UNIVAC/ 16901 ARMSTRONG AVE./ IRVINE CA 92714/ (714) 833-2400
92714   GREGORY L. HOPWOOD/ MINICOMPUTER OPERATIONS/ SPERRY UNIVAC/ 16901 ARMSTRONG AVE./ IRVINE CA 92714/ (714) 833-2400
92714   ERIC OLSEN/ MINICOMPUTER OPERATIONS/ SPERRY UNIVAC/ 16901 ARMSTRONG AVE./ IRVINE CA 92714/ (714) 833-2400
92714   KENNETH A. PRESCOTT JR./ MCO / PUB/ SPERRY UNIVAC/ 16901 ARMSTRONG AVE./ IRVINE CA 92714/ (714) 833-2400 X503
92714   RICHARD P. SPRAGUE/ MINICOMPUTER OPERATIONS/ SPERRY UNIVAC/ 16901 ARMSTRONG AVE./ IRVINE CA 92714/ (714) 833-2400 X119
92717   RICHARD A. EVERMAN/ REGISTRAR'S OFF./ 215 ADMINISTRATION BLDG./ UNIV. OF CALIF. - IRVINE/ IRVINE CA 92717
92717   STEPHEN D. FRANKLIN/ COMPUTING FACILITY/ UNIV. OF CALIFORNIA - IRVINE/ IRVINE CA 92717/ (714) 833-5154
92802   RICHARD BEELER/ 1640 W. BALL RD. - APT. 105/ ANAHEIM CA 92802
92803   C. L. HORNEY/ MICROELECTRONIC DEVICE DIV./ D/832-RC27/ ROCKWELL INTERNATIONAL/ P.O. BOX 3669/ ANAHEIM CA 92803/ (714) 632-3860
92805   ARVIND AGRAWAL/ 1142 W. FAY LANE #8/ ANAHEIM CA 92805/ (714) 778-4800 X495
92805   BRUCE A. BROWN/ DEPT. PG-678/ 1316 ROSEWOOD PLACE/ ANAHEIM CA 92805/ (714) 778-4800 (WORK)/ (714) 991-0929 (HOME)
93003   WIBERTA STONE/ 228 BRENTWOOD AVE./ VENTURA CA 93003/ (805) 642-8466
93010   MARK JUNGWIRTH/ 5408 E. HOLLY RIDGE DR./ CAMARILLO CA 93010/ (805) 484-9574
93017   JIM MCCORD/ 330 VEREDA LEYENDA/ GOLETA CA 93017/ (805) 968-6681
93106   LAURIAN M. CHIRICA/ DEPT. OF EECS/ UNIV. OF CALIFORNIA - SANTA BARBARA/ SANTA BARBARA CA 93106/ (805) 967-5135
93106   HUGH M. KAWABATA/ COMPUTER CENTER/ UNIV. OF CALIF. - SANTA BARBARA/ SANTA BARBARA CA 93106/ (714) 968-7837
93111   ATTENTION: NANCY BROOKS/ SCIENCE AND TECHNOLOGY DIVISION/ GENERAL RESEARCH CORPORATION/ P.O. BOX 6770/ SANTA BARBARA CA 93111/ (805) 964-7724
93120   FRED BELLOMY/ THE INFO-MART/ P.O. BOX 2400/ SANTA BARBARA CA 93120/ (805) 965-5555/ 965-0265
93407   JAMES L. BEUG/ DEPT. OF COMP. SCI./ CALIFORNIA POLYTECHNIC STATE U/ S.LUIS OBISPO CA 93407/ (805) 546-2861
93501   JOHN T. GARDNER/ 16425 KOCH ST./ MOJAVE CA 93501/ (805) 824-2578
94022   DENNIS PAULL/ PAULL ASSOCIATES/ 814 ECHO DR./ LOS ALTOS CA 94022/ (415) 948-9275
94022   JOHN H. WENSLEY/ 22451 HOLT AVE./ LOS ALTOS CA 94022/ (415) 964-9456
94025   JOHN BORGELT/ 1016 MIDDLE AVE./ MENLO PARK CA 94025
94025   HOWARD M. ZEIDLER/ STANFORD RESEARCH INSTITUTE/ MENLO PARK CA 94025/ (415) 326-6200
94025   DEAN MILLER/ 146 SANTA MARIA AVE./ PORTOLA VALLEY CA 94025/ (415) 851-2781/ (415) 961-4380 (OFFICE)
94040   GREGORY L. NELSON/ APARTMENT 31/ 2280 CALIFORNIA ST./ MOUNTAIN VIEW CA 94040
94040   ASHOK SURI/ 106 EUNICE AVE./ MOUNTAIN VIEW CA 94040
94042   DAVID MILLER/ P.O. BOX 205/ MOUNTAIN VIEW CA 94042/ (415) 966-2266
94043   R. A. STILLMAN/ ODELL INDUSTRIES CORP./ 1940 COLONY ST./ MOUNTAIN VIEW CA 94043/ (415) 961-1090
94043   PETER ECCLESINE/ FORD AEROSPACE/ 2361 LAURA LANE/ MT. VIEW CA 94043/ (415) 968-8044
94061   DAN ZURAS/ 1928 MADDUX DR./ REDWOOD CITY CA 94061/ (415) 368-5005
94066   BRUCE A. BARRETT/ 777 3RD AVE/ SAN BRUNO CA 94066/ (415) 873-3199
94086   RICH ALTMAIER/ 655 S. FAIROAKS AVE. APT. G101/ SUNNYVALE CA 94086/ (408) 732-7485
94086   DENNIS S. ANDREWS/ $/ AMDAHL CORP./ 1250 E. ARQUES AVE/ SUNNYVALE CA 94086/ (408) 746-6301
94086   GLENN T. EDENS/ BLDG 7B MS-235/ NATIONAL SEMICONDUCTOR/ 165 SAN GABRIEL/ SUNNYVALE CA 94086/ (408) 737-6046
94086   DAVE GRAHAM/ 581 KIRK AVE./ SUNNYVALE CA 94086/ (408) 257-7000
94086   THOMAS E. GRANVOLD/ 1119C REED AVE./ SUNNYVALE CA 94086/ (408) 247-7568
94086   DAVID TERRY JONES/ CONTROL DATA CORP./ P.O. BOX 7090/ SUNNYVALE CA 94086/ (408) 734-7466
94086   KEN RENWORTH/ MEGATEST CORP./ 486 MERCURY DR./ SUNNYVALE CA 94086/ (408) 736-1700
94086   JERRY W. SUBLETT/ 1249 BIRCHWOOD DR./ SUNNYVALE CA 94086/ (415) 744-0190
94086   ARTHUR C. WILLIS/ AMDAHL CORP./ 1250 EAST ARQUES AVE./ SUNNYVALE CA 94086/ (408) 746-6000
94087   KEITH G. TAFT/ T. E. E. CO./ 823 MANGO AVE./ SUNNYVALE CA 94087/ (408) 735-8423
94087   P. TORGRIMSON/ 528 CASTLEROCK/ SUNNYVALE CA 94087/ (408) 245-4578
94088   RICHARD CORE/ PO BOX 61628/ SUNNYVALE CA 94088/ (408) 735-8400 X285
94088   GARY W. WINIGER/ P.O. BOX 60835/ SUNNYVALE CA 94088/ (415) 964-6982/ (408) 742-5647 (WORK)
94105   J. GILMER/ 16TH FLOOR/ IBM CORP./ 425 MARKET ST./ SAN FRANCISCO CA 94105
94105   CHRISTOPHER OHLAND/ SUITE 201/ DATA 100/ ONE ECKER BLDG./ SAN FRANCISCO CA 94105/ (415) 546-6000
94105   T. R. SIMONSON/ SIMONSON CONSULTING ENGINEERS/ 612 HOWARD ST./ SAN FRANCISCO CA 94105/ (415) 392-5388
94109   PAUL MILLER/ PAUL MILLER & ASSOCIATES/ 1221 JONES ST./ SAN FRANCISCO CA 94109/ (415) 397-4116
94114   RICHARD C. LUND/ 703 NOE ST./ SAN FRANCISCO CA 94114/ (415) 824-5074
94127   VICTOR LEDIN/ 445 MONTICELLO STREET/ SAN FRANCISCO CA 94127
94133   WILLIAM F. ANDERSON/ MICRO INFORMATION SYSTEMS/ 158 VALPARAISO/ SAN FRANCISCO CA 94133/ (415) 441-4597
94137   ANN PORCH/ INTERACTIVE CORP. SERVICES - #3433/ BANK OF AMERICA/ P.O. BOX 37000/ SAN FRANCISCO CA 94137/ (415) 522-5222
94304   JOHN P. MCGINITIE/ SYSTEMS DEVELOPMENT DEPT./ ITEL CORP./ 3145 PORTER DRIVE/ PALO ALTO CA 94304/ (415) 494-9191
94305   ATTN: SERIAL RECORDS DIV./ STANFORD UNIV. LIBRARIES/ STANFORD CA 94305
94305   JON F. CLAERBOUT/ DEPT. OF GEOPHYSICS/ STANFORD UNIVERSITY/ STANFORD CA 94305
94305   DAVID JON FYLSTRA/ P.O. BOX 10051/ STANFORD CA 94305
94305   SCOTT WAKEFIELD/ DIGITAL SYSTEMS LABORATORY/ STANFORD UNIV./ STANFORD CA 94305/ (415) 497-0377
94402   ROSS ALLARDYCE/ 725 MELISSA CT./ SAN MATEO CA 94402
94402   I. D. SOUTHWELL/ 250 GRAMERCY DR./ SAN MATEO CA 94402
94501   GARY E. LAWRENCE/ 1417 CENTRAL AVE./ ALAMEDA CA 94501
```

94545 PING K. LIAO/ 2499 CONSTELLATION DR./ HAYWARD CA 94545/ (408) 988-7777 X261
94546 ELIZABETH CROCKER/ 4322 SEVEN HILLS/ CASTRO VALLEY CA 94546
94596 JOHN GULBENK/ P.O. BOX 4509/ WALNUT CREEK CA 94596/ (415) 932-4250
94606 AL FRANCIS/ GREAT AMERICAN WIDGIT CO./ 1010 22ND AVENUE/ OAKLAND CA 94606/ (415) 968-2752/ (415) 532-5686
94611 ROBERT C. NICKERSON/ 6966 COLTON BLVD/ OAKLAND CA 94611/ (415) 339-0436
94618 CHARLES F. MURPHY/ 5201 MASONIC AVENUE/ OAKLAND CA 94618
94619 WILSON T. PRICE/ MERRITT COLLEGE/ 12500 CAMPUS DRIVE/ OAKLAND CA 94619/ (415) 531-4911
94702 RANDY NIELSEN/ 1780 FRANKLIN ST./ BERKELEY CA 94702
94702 RICHARD W. HAMILTON/ 1074 WEST 3RD/ EUGENE OR 94702
94703 JAMES A. WOODS/ 2014A WOOLSEY ST./ BERKELEY CA 94703/ (415) 849-4346
94708 PAUL TEICHOLZ/ 1322 BAY VIEW PL./ BERKELEY CA 94708/ (415) 843-4232
94709 MAX HINCHMAN/ 780 CRESTON RD./ BERKELEY CA 94709
94941 AYERS LOCKSMITHING/ 227 SHORELINE HWY./ MILL VALLEY CA 94941/ (415) 383-1415
94941 ROBERT M. BAER/ 379 COUNTRYVIEW DRIVE/ MILL VALLEY CA 94941/ (415) 383-1655
95014 JOHN AHLSTROM/ OLIVETTI CORP. OF AMERICA/ 20370 TOWN CENTER LANE/ CUPERTINO CA 95014
95014 ATTN: RUTH SUGARMAN/ TYMSHARE TECHNICAL LIBRARY/ 20705 VALLEY GREEN DR./ CUPERTINO CA 95014
95014 DONALD E. GRIMES/ TYMSHARE INC./ 20705 VALLEY GREEN DRIVE/ CUPERTINO CA 95014/ (408) 446-6586
95014 JAMES W. HUFFMAN/ 8052 PARK VILLA CIRCLE/ CUPERTINO CA 95014
95014 SCOTT JAMESON/ HEWLETT PACKARD/ 11000 WOLFE ROAD/ CUPERTINO CA 95014/ (408) 257-7000 X2530
95014 JOE KEEFE/ 10730 WUNDERLICH/ CUPERTINO CA 95014/ (408) 257-2140
95014 DON MOXON/ 10410 STOKES AVE./ CUPERTINO CA 95014
95014 DAVID F. OHL/ P.O. BOX 257/ CUPERTINO CA 95014/ (408) 926-9803
95014 RICHARD PALCHIK/ TYMNET/ 20665 VALLEY GREEN RD./ CUPERTINO CA 95014/ (408) 446-6652
95014 BOB SHEPARDSON/ BLDG C4-H/ SHEPARDSON MICROSYSTEMS/ 20823 STEVENS CREEK BLVD./ CUPERTINO CA 95014/ (408) 257-9900
95014 LES VOGEL/ 7960 MCCLELLAN #3/ CUPERTINO CA 95014
95030 VINCENT BUSAM/ AZ-TECH ASSOCIATES INC/ 25754 ADAMS ROAD/ LOS GATOS CA 95030/ (408) 353-3277
95035 RICHARD M. LADDEN/ 1404 ACADIA AVE./ MILPITAS CA 95035
95050 ARTIE GREEN/ HEWLETT PACKARD/ 3003 SCOTT BLVD./ SANTA CLARA CA 95050/ (408) 249-7000
95050 E. HAROLD WILLIAMS/ SYSCOM/ 2996 SCOTT BLVD./ SANTA CLARA CA 95050/ (408) 246-2437
95051 JOHN W. BURNETT/ M/S 690/ NATIONAL SEMICONDUCTOR CORP./ 2900 SEMICONDUCTOR DR./ SANTA CLARA CA 95051/ (408) 737-5228
95051 BRUCE J. EDMUNDSON/ M/S 690/ NATIONAL SEMICONDUCTOR/ 2900 SEMICONDUCTOR DR./ SANTA CLARA CA 95051/ (408) 737-5244
95051 JULIANA M. KNOX/ 3655 PRUNERIDGE NO. 186/ SANTA CLARA CA 95051/ (408) 241-5028
95051 RAYMOND M. LEONG/ 3301 HOMESTEAD ROAD APT 301/ SANTA CLARA CA 95051/ (408) 733-2600
95051 HENRY MCGILTON/ 3480 GRANADA AVE #234/ SANTA CLARA CA 95051/ (408) 984-2493
95051 DAVID W. SALLUME/ 3480 GRANADA AVE. APT. 161/ SANTA CLARA CA 95051/ (805) 937-4541
95051 ED SCHOELL/ DEPT. NSAV/ NATIONAL SEMICONDUCTOR/ 2900 SEMICONDUCTOR DRIVE/ SANTA CLARA CA 95051
95051 A. I. STOCKS/ 3500 GRANADA #421/ SANTA CLARA CA 95051/ (408) 243-6985
95051 MIKE TRAVIS/ 3255 - 3A SCOTT BLVD./ SANTA CLARA CA 95051/ (408) 249-5540
95051 RICHARD M. WOODWARD/ AMERICAN MICROSYSTEMS INC./ 3800 HOMESTEAD RD./ SANTA CLARA CA 95051/ (408) 246-0330
95051 PETER YOUTZ/ C/O DMC/ 2300 OWEN ST./ SANTA CLARA CA 95051/ (408) 249-1111
95053 MIKES SISIOS/ INFORMATION PROC. CENTER/ SANTA CLARA UNIV./ SANTA CLARA CA 95053/ (408) 984-4582
95125 ALLEN L. AMBLER/ AMDAHL CORP./ 1250 EAST ARQUES AVE./ SUNNYVALE CA 95125/ (408) 746-6567
95126 JOHN H. KILFOIL/ 1777 TOPEKA AVE./ SAN JOSE CA 95126/ (408) 286-3166 (HOME)/ (408) 299-4251 (WORK)
95127 JEAN-CLAUDE ROY/ 11300 ENCHANTO VISTA/ SAN JOSE CA 95127/ (415) 257-7000 X3581
95128 ATTN: TIMESHARING BUSINESS SYSTEMS/ 3031 TISCH WAY/ SAN JOSE CA 95128
95129 ARNIE EGEL/ 7200 BOLLINGER NO. 307/ SAN JOSE CA 95129
95129 JOHN MURRAY/ 11760 SHARON DRIVE/ SAN JOSE CA 95129
95132 DANIEL F. CONWAY/ 2551 PANTALIS DR./ SAN JOSE CA 95132/ (408) 739-7700 X3492
95132 ROBERT R. VAN TUYL/ 2572 OHLONE DRIVE/ SAN JOSE CA 95132/ (408) 258-8961
95132 ANDREW HARRIS ZIMMERMAN/ 3422 DUTCHESS COURT/ SAN JOSE CA 95132
95376 TOM HORSLEY/ 1750 MELLO COURT/ TRACY CA 95376/ (209) 836-0764
95401 DEAN BILLING/ SOLVE CONSULTING/ 1815 PETERSON LANE/ SANTA ROSA CA 95401/ (707) 545-7778
95610 RICHARD HERBERT/ 5818 PRIMROSE DR./ CITRUS HEIGHTS CA 95610
95610 ROBERT H. MIX JR./ 6941 LE HAVRE WAY/ CITRUS HEIGHTS CA 95610
95650 CLINTON PACE/ 7130 MORNINGSIDE DRIVE/ LOOMIS CA 95650/ (916) 791-1504
95660 JOHN BLACKWOOD/ 3829 A STREET/ N. HIGHLANDS CA 95660
95926 DAN +ROBIN BARNES/ 279 RIO LINDO AVE. NO. 7/ CHICO CA 95926/ (916) 891-1232
95927 GEORGE N. ARNOVICK/ CALIFORNIA MICROCOMPUTER COMPANY/ P.O. BOX 3199/ CHICO CA 95927/ (916) 891-1420
95955 ALAN H. SWANN/ MTS COMP. SERVICE/ P.O. BOX 487/ MAXWELL CA 95955
96224 MARCUS C. CORNELL/ 572-40-5425 2D/ HHC DISCOM (DDC)/ APO/ SAN FRANCISCO CA 96224
96786 RICHARD FOULK/ 95-269 WAIKALAMI DR. - C604/ WAHIAWA HI 96786
96822 WESLEY PETERSON/ DEPT OF ICS/ U OF HAWAII/ 2565 THE MALL/ HONOLULU HI 96822/ (808) 948-7420
97005 PETER H. MACKIE/ PHM AND ASSOCIATES/ P.O. BOX 427/ BEAVERTON OR 97005/ (503) 644-0111
97068 WILLIAM C. PRICE/ 28282 SW MOUNTAIN ROAD/ WEST LINN OR 97068/ (503) 644-0161 X5847
97077 HOWARD CUNNINGHAM/ MS 50-362/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 644-0161
97077 BOB DIETRICH/ MS 61-272/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 682-3411 X2398
97077 SID FERMI/ MS 50-435/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077
97077 TERRY HAMM/ M.S. 60-456/ MS 61-272/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 682-3411
97077 JON MARSHALL/ M.S. 60-456/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 682-3411 X2586
97077 PAUL L. MCCULLOUGH/ MS 50/362/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 644-0161 X6157
97077 LYNN SAUNDERS/ MS 50-454/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 644-0161 X 5616
97077 ROD STEEL/ MS 61-272/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 638-3411 X2516
97123 GLEN FULLMER/ GENERAL DATA SERVICES/ 2400 SE BROOKWOOD AVE. #1B/ HILLSBORO OR 97123/ (503) 640-4040/ (503) 644-0161 X5976
97201 ATTN: OREGON MINI-COMPUTER SOFTWARE I*/ 2340 S.W. CANYON ROAD/ PORTLAND OR 97201/ (503) 226-7760
97201 JOHN WONG/ 3502 SW GALE/ PORTLAND OR 97201/ (503) 645-6464
97202 DORSEY DRANE/ COMPUTER CENTER/ REED COLLEGE/ PORTLAND OR 97202
97207 RICHARD T. BROWN/ WOOD MARKETS INC./ P.O. BOX 669/ PORTLAND OR 97207/ (503) 645-5687
97210 DAVE BAASCH/ 2683 NW RALEIGH/ PORTLAND OR 97210/ (503) 223-6570
97223 HANS JONGE VOS/ 14130 S.W. FERN ST./ TIGARD OR 97223/ (503) 644-1283
97225 LORIN RICKER/ 9450 S.W. BARNES RD./ PORTLAND OR 97225/ (503) 297-5671
97301 SHELLEY GILES/ ATKINSON GRAD SCHOOL/ WILLAMETTE UNIV./ 900 STATE STREET/ SALEM OR 97301
97330 OLE ANDERSON/ 4210 NW CRESCENT VALLEY/ CORVALLIS OR 97330/ (503) 753-6995
97330 LARRY BILODEAU/ DIGITAL ELECTRONIC SYSTEMS/ 205 NW 31ST/ CORVALLIS OR 97330/ (503) 754-1694
97330 DAVID F. CAUTLEY/ DEPT. OF COMPUTER SCIENCE/ GENERAL INFORMATION SYSTEMS INC./ 155 S.W. MADISON/ CORVALLIS OR 97330/ (503) 754-1711
97330 KURT KOHLER/ 2854 N W JOHNSON/ CORVALLIS OR 97330/ (503) 753-1770
97330 RUSSELL RUBY/ 627 SW 16TH/ CORVALLIS OR 97330/ (503) 753-2091
97331 KAMRAN MALIK/ DEPT. OF COMPUTER SCIENCE/ OREGON STATE UNIV./ CORVALLIS OR 97331/ (503) 754-3273
97401 KENT LOOBEY/ 2110 CARMEL AVE/ EUGENE OR 97401/ (503) 686-8110
97401 DAVID MEYER/ DUNHILL PERSONEL INC./ 1551 OAK ST./ EUGENE OR 97401/ (503) 484-9242
97402 TERRY LIITTSCHWAGER/ MCKENZIE FLYING SERVICE INC/ 90600 GREENHILL ROAD/ EUGENE OR 97402/ (503) 688-0971
97459 R. BUSH/ P.O. BOX F/ NORTH BEND OR 97459
97850 JOHN BUCZEK/ CYBERMEDIC/ P.O. BOX 893/ LA GRANDE OR 97850
98006 STEPHEN J. WEINBERGER/ 14032 SE NEWPORT WAY/ BELLEVUE WA 98006
98006 JOHN D. WOOLLEY/ 6722 128TH AVE. SE/ BELLEVUE WA 98006/ (206) 237-2753
98008 KEITH MITCHELL/ 16213 SE 28 PL/ BELLEVUE WA 98008/ (206) 237-2753
98020 KASI SESHADRI BHASKAR/ 22828 76TH AVE. W. APT. #33/ EDMONDS WA 98020/ (206) 778-6731 (HOME)/ (206) 774-2381 (WORK)
98040 ROBERT EMERSON/ HONEYWELL INFORMATION SYSTEMS/ 9555 SE 36TH STREET/ MERCER ISLAND WA 98040/ (206) 233-2077
98043 GARY S. ANDERSON/ JOHN FLUKE MFG. CO. INC./ P.O. BOX 43210 - M.S. 29/ MOUNTLAKE TERR WA 98043/ (206) 774-2296
98043 NORM SEETHOFF/ MAIL STOP 25/ JOHN FLUKE MFG. CO. INC./ P.O. BOX 43210/ MOUNTLAKE TER* WA 98043/ (206) 774-2381
98055 R. A. LOVESTEDT/ 20427 SE 192/ RENTON WA 98055/ (206) 432-0769
98105 JAMES G. BARON/ 5012 11TH NE NO. F/ SEATTLE WA 98105
98112 ROBERT W. RIEMANN/ NW & ALASKA FISHERIES/ FISHERIES DATA & MANAGEMENT SYSTEMS/ NOAA/ 2725 MONTLAKE BLVD. EAST/ SEATTLE WA 98112
98115 PETER A. ARMSTRONG/ 444 NE RAVENNA BLVD #309/ SEATTLE WA 98115
98115 H. M. KUHLMANN/ 2281 N.E. 60TH/ SEATTLE WA 98115/ (206) 525-3991
98115 ROBERT C. SLATE/ THERMOTEK ASSOCIATES/ 8225 17TH AVE. NE/ SEATTLE WA 98115/ (206) 523-1559
98124 DAVID F. WEIL/ MS 73-03/ BOEING COMPUTER SERVICES INC./ P.O. BOX 24346/ SEATTLE WA 98124/ (206) 237-5632
98161 FRED BALLANTINE/ C/O ARTHUR YOUNG & CO./ 2100 FINANCIAL CENTER/ SEATTLE WA 98161/ (206) 623-9000
98178 MICHAEL R. MCGUIRE/ 12022 71ST S. #308/ SEATTLE WA 98178
98188 RICHARD R. DYMANT/ GENERAL MANAGER/ DIGITAL BUSINESS SYSTEMS INC./ 774 INDUSTRY DR./ TUKWILA WA 98188/ (206) 575-3740
98195 ATTN: UNIV. OF WASHINGTON/ TECHNICAL SUPPORT SERVICES/ LOWER LEVEL JE-15/ 4545 15TH N.E./ SEATTLE WA 98195
98407 EDRICE REYNOLDS/ EDRICE ENTERPRISES/ P.O. BOX 166/ TACOMA WA 98407
98507 PHIL HUGHES/ P.O. BOX 2847/ OLYMPIA WA 98507/ (206) 352-9637
98907 JAY WOODS/ P.O. BOX 1016/ YAKIMA WA 98907/ (509) 452-9133
99163 JOHN MILLER/ P.O. BOX 2118 C.S./ PULLMAN WA 99163/ (509) 355-6636 (C.S. DEPT.)/ (509) 335-6147 (OFFICE)
99352 R. C. LUCKEY/ 1110 GILMORE/ RICHLAND WA 99352/ (509) 943-3107
99352 TOM MATHIEU/ BATTELLE PACIFIC N.W. LABS/ BATTELLE BOULEVARD/ RICHLAND WA 99352/ (509) 946-3711
99507 TOM SWANSON/ 7505 BERN STREET/ ANCHORAGE AK 99507
RA-8000 ARGENTINA MARCELO SANSEAU/ OHIGGINS 295/ BAHIA BLANCA RA-8000/ ARGENTINA
2600 AUSTRALIA NICK HAMMOND/ DFM (UNDERWATER WEAPONS)/ NAVY OFFICE/ CANBERRA A.C.T. 2600/ AUSTRALIA/ (062) 482858
3046 AUSTRALIA W. DAVIS/ 4 GRANDVIEW ST./ GLENROY VICTORIA 3046/ AUSTRALIA
3072 AUSTRALIA M. RAHILLY/ 2 RITA STREET/ EAST PRESTON VICTORIA 3072/ AUSTRALIA/ 478 6451
3130 AUSTRALIA P. S. EDWARDS/ 101 MAIN ST/ BLACKBURN VICTORIA 3130/ AUSTRALIA/ 341-6842
3168 AUSTRALIA ATTN: DEPT. OF COMP. SCI./ MONASH UNIV./ CLAYTON VICTORIA 3168/ AUSTRALIA
4067 AUSTRALIA DAN B. JOHNSTON/ DEPT. OF COMPUTER SCIENCE/ UNIV. OF QUEENSLAND/ ST. LUCIA QUEENSLAND 4067/ AUSTRALIA/ (07) 377 6930
5001 AUSTRALIA ATTN: PROGRAM LIBRARIAN/ COMPUTING CENTRE/ UNIVERSITY OF ADELAIDE/ BOX 498 G.P.O./ ADELAIDE S.A. 5001/ AUSTRALIA/ 61 223 4333 X2720
5001 AUSTRALIA CHRIS D. MARLIN/ DEPT OF COMPUTING SCIENCE/ UNIVERSITY OF ADELAIDE/ G.P.O. BOX 498/ ADELAIDE S.A. 5001/ AUSTRALIA/ (08) 223 4333 X2762
BELGIUM ATTN: BIBLIOTHEQUE CENTRALE/ FACULTES UNIVERSITAIRES/ N-D. DE LA PAIX/ RUE DE BRUXELLES 61/ NAMUR/ BELGIUM
B-1170 BELGIUM MARTINE DE GERLACHE/ 177 BTE 1/ SPERRY UNIVAC/ CHAUSEE DE LA THULFE/ BRUXELLES B-1170/ BELGIUM
BRAZIL ROBERIO DIAS/ P.O. BOX 30028/ SAO PAULO/ BRAZIL/ 444-3701
A1C 5M3 CANADA H. J. AU/ P.O. BOX 1025/ ST. JOHN'S NEWFNDLAND A1C 5M3/ CANADA

A1C 5S7	CANADA	RANDY DODGE/ COMPUTING SERVICES/ MEMORIAL UNIVERSITY/ ST. JOHN'S NEWFNDLAND A1C 5S7/ CANADA/ (709) 753-1200 X2746
B0P 1X0	CANADA	BILL WILDER/ SCHOOL OF COMPUTER SCIENCE/ ACADIA UNIV./ WOLFVILLE N. SCOTIA B0P 1X0/ CANADA
B3L 4L5	CANADA	DENNIS MISENER/ DYMAXION RESEARCH LTD./ BOX 1053 ARMDALE STN./ HALIFAX N.SCOTIA B3L 4L5/ CANADA/ (902) 429-3175
C1A 4P3	CANADA	J. W. HANCOCK/ COMPUTER CENTER/ UNIV. OF PRINCE EDWARD ISLAND/ CHARLOTTETOWN P.E.I. C1A 4P3/ CANADA
HC3 3J7	CANADA	LUC LAVOIE/ DEPT. I. R. O./ UNIVERSITE DE MONTREAL/ C.P. 6128 SUCCURSALE A/ MONTREAL QUEBEC HC3 3J7/ CANADA/ (514) 737-3700
H3C 3A9	CANADA	LES SATENSTEIN/ PERF. OPTIMIZATION / PROCESSING OPERA*/ ROYAL BANK OF CANADA/ BOX 6001/ MONTREAL QUEBEC H3C 3A9/ CANADA
H3N 2T6	CANADA	IAN MACMILLAN/ 7939 BIRNAM/ MONTREAL QUEBEC H3N 2T6/ CANADA
H3S 2L7	CANADA	MARTIN MILLER/ 6650 WILDERTON AVENUE/ MONTREAL QUEBEC H3S 2L7/ CANADA/ (514) 384-1030
H4T 1N1	CANADA	MARY SUTTON/ A.E.S. DATA LTD./ 570 RUE MCCAFFREY/ MONTREAL QUEBEC H4T 1N1/ CANADA/ (514) 341-5430 X307
H9P 1J3	CANADA	BARRIE D. MACLEOD/ POINTE-CLAIRE FIRST FLOOR/ PECHES ET ENVIRONNEMENT CANADA/ 2121 TRANS-CANADA HIGHWAY/ DORVAL QUEBEC H9P 1J3/ CANADA/ (514) 683-8152
J0B 2C0	CANADA	JOHN SEITZ/ C.P. 525/ NORTH HATLEY QUEBEC J0B 2C0/ CANADA/ (819) 842-2375
K0J 1P0	CANADA	P. D. MCMORRAN/ P.O. BOX 904/ DEEP RIVER ONTARIO K0J 1P0/ CANADA/ (613) 584-3311
K1S 5B6	CANADA	RICHARD F. DILLON/ DEPT. OF PSYCHOLOGY/ CARLETON UNIV./ OTTAWA ONTARIO K1S 5B6/ CANADA/ (613) 231-3636
K2H 5S3	CANADA	ROGER F. BURROWS/ 33 HOBART CRESCENT/ OTTAWA ONTARIO K2H 5S3/ CANADA
K2K 1X4	CANADA	W. MITCHELL/ OTTAWA COMPUTER GROUP/ P.O. BOX 13218/ KANATA ONTARIO K2K 1X4/ CANADA
L1S 3B4	CANADA	GORDON C. BOWRON/ 120 GREGORY RD./ AJAX ONTARIO L1S 3B4/ CANADA/ (416) 683-8655
L5C 1C8	CANADA	JIM FINN/ SONOTEK/ 2410-5 DUNWIN DR./ MISSISSAUGA ONTARIO L5C 1C8/ CANADA
L5N 1W2	CANADA	S. B. MATTHEWS/ R & D CENTRE/ AES DATA LTD./ 2332 MILLRACE COURT/ MISSISSAUGA ONTARIO L5N 1W2/ CANADA
L7M 1K4	CANADA	G. CHALIFOUR/ DIAMOND CANAPOWER LTD./ 1122 PIONERR RD./ BURLINGTON ONTARIO L7M 1K4/ CANADA/ (416) 335-0321
M2J 2W6	CANADA	DOUG MARSHALL/ BXLE SYSTEMS LTD./ 617 SENECA HILL DRIVE/ WILLOWDALE ONTARIO M2J 2W6/ CANADA
M3C 1H7	CANADA	BRUCE DAVIDSON/ DEPT. 806/ IBM CANADA LABORATORY/ 1150 EGLINTON AVE. EAST/ DON MILLS ONTARIO M3C 1H7/ CANADA/ (416) 443-3162
M3C 1Z3	CANADA	ATTENTION: DIANNE CAMERON/ SOFTWARE DEVELOPMENT/ CONSOLIDATED COMPUTER INC./ 50 GERVAIS DRIVE/ DON MILLS ONTARIO M3C 1Z3/ CANADA/ (416) 449-8401
M3J 1P3	CANADA	JOHN C. MCCALLUM/ COMPUTER SCIENCE DEPT./ YORK UNIV./ DOWNSVIEW ONTARIO M3J 1P3/ CANADA
M4R 1V2	CANADA	TOM A. TROTTIER/ 411 DUPLEX AVE. - APT. 612/ TORONTO ONTARIO M4R 1V2/ CANADA/ (416) 488-8802
M4R 1Z2	CANADA	M. DIANNE CAMERON/ 66 EDITH DRIVE/ TORONTO ONTARIO M4R 1Z2/ CANADA/ (416) 488-5738 (HOME)/ (416) 449-8401 (WORK)
M5N 2Z6	CANADA	DAVID ROSENBOOM/ P.O. BOX 543 - STATION Z/ TORONTO ONTARIO M5N 2Z6/ CANADA
M9A 3V3	CANADA	DONALD R. BAIN/ 319 THE KINGSWAY APT #10/ ISLINGTON ONTARIO M9A 3V3/ CANADA
N2C 3E0	CANADA	T. A. CARGILL/ DEPT. OF COMP. SCI./ UNIV. OF WATERLOO/ WATERLOO ONTARIO N2C 3E0/ CANADA
N2J 4G5	CANADA	F. A. CELLINI/ NCR CANADA LTD./ 580 WEBSTER ST. N/ WATERLOO ONTARIO N2J 4G5/ CANADA/ (519) 884-1710 X196
N2L 3G1	CANADA	W. MORVEN GENTLEMAN/ COMPUTER SCIENCE DEPT./ UNIVERSITY OF WATERLOO/ WATERLOO ONTARIO N2L 3G1/ CANADA/ (519) 885-1211 X2187/ (519) 885-1211
N6A 4K1	CANADA	PAUL DENNISON/ LONDON LIFE INSURANCE CO./ 255 DUFFERIN AVE./ LONDON ONTARIO N6A 4K1/ CANADA/ (519) 432-5281 X164
N6A 5B7	CANADA	ATTN: PROGRAM LIBRARY/ COMPUTING CENTER/ 223 NATURAL SCIENCE CENTER/ U OF WESTERN ONTARIO/ LONDON ONTARIO N6A 5B7/ CANADA/ (519) 679-2151 X45
N6A 5B7	CANADA	GORDON BARKER/ NATURAL SCIENCE CENTRE/ 223 COMPUTING CENTER/ UNIVERSITY OF WESTERN ONTARIO/ LONDON ONTARIO N6A 5B7/ CANADA/ (519) 679-2151
N6A 5B9	CANADA	L. MCHARDY/ ENG. & MATH. SCI. BLDG./ COMP. SCI. DEPT./ UNIV. OF WESTERN ONTARIO/ LONDON ONTARIO N6A 5B9/ CANADA/ (519) 679-2636
P0J 1K0	CANADA	ROSS ALEXANDER/ P.O. BOX 1175/ HAILEYBURY ONTARIO P0J 1K0/ CANADA/ (705) 672-5193
P7B 5E1	CANADA	ATTN: DEPT. OF MATHEMATICAL SCI./ LAKEHEAD UNIV./ THUNDER BAY ONTARIO P7B 5E1/ CANADA/ (807) 345-2121 X469
R3H 0R9	CANADA	D. A. MOIR/ C/O CYBERSHARE LTD/ 550 BERRY ST./ WINNIPEG MANITOBA R3H 0R9/ CANADA/ (204) 786-5831
S7H 1B5	CANADA	DEREK F. ANDREW/ 223 6TH ST. EAST/ SASKATOON SASK. S7H 1B5/ CANADA/ (306) 665-3226
T6G 2C2	CANADA	MARK R. JOHNSON/ ATMOSPHERIC SCIENCES DIV./ ALBERTA RESEARCH COUNCIL/ 11315-87 AVENUE/ EDMONTON ALBERTA T6G 2C2/ CANADA/ (403) 432-8125
V3N 4N8	CANADA	KIM WILLIAMS/ SUITE 602/ 7818 6TH ST./ BURNABY B.C. V3N 4N8/ CANADA/ (604) 524-9741
V5A 1S6	CANADA	I. GANAPATHY/ COMPUTING CENTRE/ NOOTKA BUILDING/ SIMON FRASER UNIV./ BURNABY B.C. V5A 1S6/ CANADA/ (604) 291-4712
V6K 2C1	CANADA	NORMAN J. JAFFE/ GNS SYSTEMS LTD/ 3054 W. 8TH AVENUE/ VANCOUVER B.C. V6K 2C1/ CANADA/ (604) 731-9028
V6T 1W5	CANADA	C. A. MILLER/ TRIUMF/ UNIV. OF BRITISH COLUMBIA/ VANCOUVER B.C. V6T 1W5/ CANADA/ (604) 228-4711
V6X 2L4	CANADA	PETER NEEDHAM/ 2771 NUMBER FOUR RD./ RICHMOND B.C. V6X 2L4/ CANADA
DK-2100	DENMARK	JORGEN OXENBOLL/ COMPUTER DEPT./ H.C.ORSTED INSTITUTET/ UNIV. OF KOBENHAVN/ UNIVERSITETSPARKEN 5/ KOBENHAVN O DK-2100/ DENMARK
DK-2730	DENMARK	ROLF MOLICH/ DANSK DATA ELEKTRONIK/ HERLEV HOVEDGADE 207/ HERLEV DK-2730/ DENMARK/ 45 2 84 50 11
DK-2770	DENMARK	FLEMMING PEDERSEN/ POSTPARKEN 31 1 TV./ KASTRUP DK-2770/ DENMARK/ 01-513885
	DOMINICAN REP.	JOSE M. FLOREN/ ENS. PIANTINI/ CALLE 22 #42A/ SANTO DOMINGO/ DOMINICAN REP./ 565-8557/ 567-6515
SF-00510	FINLAND	JAN-HENRIK JOHANSSON/ PORVOONKATU 26 C 36/ HELSINKI 51 SF-00510/ FINLAND/ 90-140022
SF-02730	FINLAND	HEIKKI LEHTINEN/ LEHTITIE 16/ LAAKSOLAHTI SF-02730/ FINLAND
SF-33540	FINLAND	JYRKI TUOMI/ PELLERVONKATU 9/4009/ TAMPERE 54 SF-33540/ FINLAND/ 931-50000/570 (HOME)/ 931-162125 (WORK)
D-1000	GERMANY	BERND MARTENS/ SCHILLERSTR 84/ BERLIN 49 D-1000/ GERMANY
D-3000	GERMANY	HEINZ KLEENE/ LISTERMEILE 23/ HANNOVER D-3000/ GERMANY
D-5000	GERMANY	KARL KOEHNE/ INST. FUR MED. DOKUMENTATION/ UNIVERSITAET ZU KOELN/ JOSEF-STELZMANN STR 9/ KOELN D-5000/ GERMANY/ 0221 4784164
D-6000	GERMANY	PETER C. AKWAI/ SOPHIENSTR. 32/ FRANKFURT 90 D-6000/ GERMANY/ 0611-665-4331
D-6750	GERMANY	HANS-WILM WIPPERMANN/ FB INFORMATIK/ UNIV. OF KAISERSLAUTERN/ PFAFFENBERGSTR. 95/ KAISERSLAUTERN D-6750/ GERMANY/ (0631) 854 2635
D-6900	GERMANY	PETER T. SPECK/ EMBL/ POSTFACH 102209/ HEIDELBERG D-6900/ GERMANY
D-7750	GERMANY	DIRK KRONIG/ AEG-TELEFUNKEN/ POSTFACH 2154 / BUECKLESTRASSE 1-5/ KONSTANZ D-7750/ GERMANY/ 07531-862066
D-7910	GERMANY	TILL GEISER/ FALKENSTEINWEG 8/ NEU-ULM D-7910/ GERMANY
D-8000	GERMANY	P. E. FISCHER/ CSID/ OETTINGENSTR 8A/ MUNCHEN 22 D-8000/ GERMANY/ 089-229131
D-8000	GERMANY	FELIX POPPLEIN/ SELDENECKSTR. 10/ MUNCHEN 60 D-8000/ GERMANY
	INDONESIA	HARSONO/ PUSAT KOMPUTER/ INSTITUT TEKNOLOGI BANDUNG/ JALAN GANESHA 10 / TILPON 82051-82055/ BANDUNG/ INDONESIA
	ISRAEL	NAKHSHON YESHURUN/ COMPUTATION CENTER/ BEN GURION UNIV./ P.O.B. 2053/ BEER SHEVA/ ISRAEL/ 5764453
100	JAPAN	KOICHI FUKUNAGA/ MITSUBISHI RESEARCH INSTITUTE INC./ 1-6-1 OHTENACHI - CHIYODA-KU/ TOKYO 100/ JAPAN/ 03-214-5531
113	JAPAN	EIITI WADA/ DIVISION OF ENGINEERING/ INFORMATION ENGINEERING COURSE/ UNIVERSITY OF TOKYO/ BUNKYOKU TOKYO 113/ JAPAN/ (03) 812-2111 X7486
244	JAPAN	RYUJI TAKANUKI/ LANGUAGE APPLICATION DEPT./ SOFTWARE WORKS OF HITACHI LTD./ 5030 TOTSUKA-CHO - TOTSUKA-KU/ YOKOHAMA 244/ JAPAN/ 045-881-7161 X2102
	MEXICO	MARIO MAGIDIN/ BUHARROS 67/ MEXICO 20 D.F./ MEXICO/ 522.56.94
	MEXICO	FERNANDO JAIMES/ CENTRO ELECTRONICO DE CALCULO/ ITESM/ SUCURSAL CORREOS 'J'/ MONTERREY N.L./ MEXICO
	NORWAY	TERJE NOODT/ INSTITUTE OF INFORMATICS/ UNIVERSITY OF OSLO/ P.O. BOX 1080 / BLINDERN/ OSLO 3/ NORWAY/ (02) 466800
	NORWAY	ATTN: LIBRARY/ CONTROL DATA B.V./ J. C. VAN MARKENLAAN 5/ RIJSWIJK HOLLAND/ NORWAY/ 070-949344
N-1750	NORWAY	ATTN: OSTFOLD D H. LIBRARY/ OSTFOLD DISTRIKTSHOGSKOLE/ OS ALLE 5/ HALDEN N-1750/ NORWAY
N-3000	NORWAY	DAVID E. OLAVSSEN/ KONGSBERG INGENIORHOGSKOLE/ KONGSBERG N-3000/ NORWAY
N-5000	NORWAY	OLAV NAESS/ WELHAVENSGT.65/ BERGEN N-5000/ NORWAY
	PANAMA	THEODORE J. HERRMAN/ BOX 1778/ BALBOA CANAL ZONE/ PANAMA
	SINGAPORE	JACK PAGE/ PAGE-ASIA ASSOCIATES/ 279-M SELEGIE COMPLEX/ SINGAPORE-7/ SINGAPORE/ 326102
	SOUTH AFRICA	COLIN MIEROWSKY/ 101 FAIRWAYS/ CORLETT DRIVE - ILLOVO/ JOHANNESBURG/ SOUTH AFRICA/ 788 2474
	SOUTH AFRICA	J. F. DE BEER/ COMPUTER SCIENCE/ POTCHEFSTROOM UNIVERSITY/ POTCHEFSTROOM/ SOUTH AFRICA/ 22112
0001	SOUTH AFRICA	ATTN: PERIODICALS SECTION/ CSIR LIBRARY/ P.O. BOX 395/ PRETORIA 0001/ SOUTH AFRICA
2001	SOUTH AFRICA	ATTENTION: JUDY BISHOP/ APPLIED MATHS DEPT./ STAFF COMMON ROOM / UNIV. OF THE WITWATERSRAND/ JOHANNESBURG 2001/ SOUTH AFRICA/ (01) 394011 X8656
2191	SOUTH AFRICA	ALASDAIR D. STUART/ KENSINGTON/ 84 NOTTINGHAM RD./ JOHANNESBURG 2191/ SOUTH AFRICA/ 25 2553
6140	SOUTH AFRICA	M. HOWARD WILLIAMS/ COMPUTER SCIENCE DEPT./ RHODES UNIVERSITY/ GRAHAMSTOWN 6140/ SOUTH AFRICA/ 0461-2023
7405	SOUTH AFRICA	STEVEN B. RAKOFF/ 19 LOBELIA STREET/ MILNERTON 7405/ SOUTH AFRICA/ 521447 (CAPE TOWN)
7700	SOUTH AFRICA	I. N. CHRISTOFFERSON/ 3 GUILDFORD ROAD / ROSEBANK/ CAPE TOWN 7700/ SOUTH AFRICA/ 691875
7700	SOUTH AFRICA	S. R. SCHACH/ COMP. SCI. DEPT./ UNIVERSITY OF CAPE TOWN/ RONDEBOSCH 7700/ SOUTH AFRICA/ 698531 X172
	SPAIN	RAFAEL M. BONET/ PROVIDENCIA 137/ BARCELONA 24/ SPAIN/ 34-3-3257599
	SPAIN	PERE BOTELLA/ CASANOVA 148/ BARCELONA 36/ SPAIN/ (93) 253.60.70
S-115 43	SWEDEN	KEITH ELKIN/ DIANAVAGEN 30/ STOCKHOLM S-115 43/ SWEDEN
S-145 71	SWEDEN	MICHAEL EVANS/ BALDERS VAG 3 6TR/ NORSBORG S-145 71/ SWEDEN
S-145 71	SWEDEN	SVANTE HELLSING/ TORS VAG 8 - 6 TR/ NORSBORG S-145 71/ SWEDEN/ 0753-82033
S-721 83	SWEDEN	EGON JOHANSSON/ DEPT. KDTS/ ASEA/ VASTERAS S-721 83/ SWEDEN/ 021-102988
S-751 21	SWEDEN	HANS FLACK/ DEPT. COMP. TECHNOLOGY/ TEKNIKUM/ BOX 534/ UPPSALA S-751 21/ SWEDEN/ 018 10 04 70
S-901 87	SWEDEN	PER-AKE WEDIN/ INSTITUTE OF INFORMATION PROCESSING/ UNIV. OF UMEA/ UMEA S-901 87/ SWEDEN/ 090-125600
S-902 36	SWEDEN	GORAN LINDAHL/ MARIEHEMSV. 17A/206/ UMEA S-902 36/ SWEDEN/ 090-137803
CH-2000	SWITZERLAND	NORBERT EBEL/ CENTRE DE CALCUL/ UNIVERSITE/ CHATEMERLE 20/ NEUCHATEL CH-2000/ SWITZERLAND/ 038 25 64 34
CH-8021	SWITZERLAND	ATTN: HONEYWELL BULL S.A./ MINI OEM/ JAKOB FUGLISTR. 18 / P.O.B/ ZURICH CH-8021/ SWITZERLAND/ 01-474400/ 01-2416760
	THE NETHERLANDS	ATTENTION: C. V. D. WIJGAART/ TECHNISCH CENTRUM FSW/ UNIV. OF AMSTERDAM/ ROETERSSTRAAT 15/ AMSTERDAM/ THE NETHERLANDS
	THE NETHERLANDS	DAVID A. COOPER/ C/O CACI/ KEIZERS GRACHT 534/ AMSTERDAM/ THE NETHERLANDS
	THE NETHERLANDS	ATTN: BIBLIOTHEEK 05627/ TECHNISCHE HOGESCHOOL/ POSTBUS 513/ EINDHOVEN/ THE NETHERLANDS
	THE NETHERLANDS	NIGEL W. BENNEE/ OOSTEINDE 223/ VOORBURG ZH/ THE NETHERLANDS
	THE NETHERLANDS	D. R. GIBBY/ KROMWATER 60/ ZOETERMEER/ THE NETHERLANDS
1018 WB	THE NETHERLANDS	P. VAN EMDE BOAS/ ITW/VPW/ UNIVERSITEIT VAN AMSTERDAM/ ROETERSSTRAAT 15/ AMSTERDAM 1018 WB/ THE NETHERLANDS/ 020-522 3065
2501 BD	THE NETHERLANDS	P. A. SLATS/ INFORMATION PROCESSING AND STATISTICS/ INST. TNO FOR MATHEMATICS/ P.O. BOX 297/ THE HAGUE 2501 BD/ THE NETHERLANDS
9321 GN	THE NETHERLANDS	T. J. VAN WEERT/ ELZENLAAN 28/ PEIZE 9321 GN/ THE NETHERLANDS
PL4 8AA	UNITED KINDOM	PATRICIA HEATH/ COMPUTER CENTRE/ PLYMOUTH POLYTECHNIC/ DRAKE CIRCUS/ PLYMOUTH ENGLAND PL4 8AA/ UNITED KINDOM
	UNITED KINGDOM	J. M. MCCAIG/ SCHOOL OF MATHEMATICS/ KINGSTON POLYTECHNIC/ PENRHYN RD./ KINGSTON-UPON* SURREY/ UNITED KINGDOM
	UNITED KINGDOM	JOHN ROSCOE/ SYSTEMS ENGINEERING/ 11TH FLOOR 7/BLOCK/ I. C. L. / WENLOCK WAY / WESTGORTON/ MANCHESTER ENGLAND/ UNITED KINGDOM/ 061 223 1301 X2589
	UNITED KINGDOM	RICHARD CLAYTON/ 10A STATION ROAD / MERSTHAM/ REDHILL SURREY/ UNITED KINGDOM
	UNITED KINGDOM	C. T. BRITTON/ 63 GIBBS COUCH / CARPENTERS PARK/ WATFORD HERTS / UNITED KINGDOM
AL3 4RZ	UNITED KINGDOM	WILL PICKLES/ 166 FISHPOOL ST./ ST. ALBANS HERTS. AL3 4RZ/ UNITED KINGDOM
BN3 1RA	UNITED KINGDOM	BRIAN WILLIAMS/ 67 DAVIGDOR ROAD/ HOVE SUSSEX BN3 1RA/ UNITED KINGDOM/ 0273-778389
CO4 3SQ	UNITED KINGDOM	I. R. MAC CALLUM/ DEPT. OF COMPUTER SCIENCE/ UNIV. OF ESSEX/ P.O. BOX 23 / WIVENHOE PARK/ COLCHESTER ENGLAND CO4 3SQ/ UNITED KINGDOM/ (0206) 44144
CO4 3SQ	UNITED KINGDOM	IAN H. WITTEN/ EES DEPT./ UNIV. OF ESSEX/ WIVENHOE PARK/ COLCHESTER ENGLAND CO4 3SQ/ UNITED KINGDOM/ 0206 44144 X2285
CV21 2QE	UNITED KINGDOM	C. J. THODAY/ 1A BANK ST./ RUGBY WARWICKS CV21 2QE/ UNITED KINGDOM
DD2 1SJ	UNITED KINGDOM	S. AMBLER/ 12 KELSO STREET/ DUNDEE SCOTLAND DD2 1SJ/ UNITED KINGDOM
EH1 1GZ	UNITED KINGDOM	D. S. H. ROSENTHAL/ DEPT. OF ARCHITECTURE/ UNIV. OF EDINBURGH/ 22 CHAMBERS ST./ EDINBURGH SCOTLAND EH1 1GZ/ UNITED KINGDOM
EX4 4QL	UNITED KINGDOM	D. R. ALLUM/ DEPT. OF PHYSICS/ UNIVERSITY OF EXETER/ EXETER ROAD/ EXETER ENGLAND EX4 4QL/ UNITED KINGDOM
E11 1QL	UNITED KINGDOM	JOHN HUTCHINSON/ 13 D SYLVAN ROAD / WANSTEAD/ LONDON ENGLAND E11 1QL/ UNITED KINGDOM/ 01-980-4811 X778
GL52 5AJ	UNITED KINGDOM	M. J. L. YATES/ F/0603 X66HQ/ GOVERNMENT COMMUNICATIONS HQ/ OAKLEY PRIORS ROAD/ CHELTENHAM ENGLAND GL52 5AJ/ UNITED KINGDOM/ 0242 21491 X2192
HA9 0EE	UNITED KINGDOM	A. R. M. WAJIH/ GENERAL ENGINEERING DEPT/ PULLMAN KELLOGG LTD/ STADIUM WAY/ WEMBLEY ENGLAND HA9 0EE/ UNITED KINGDOM/ 01-903 8484 X3481
HU6 7LJ	UNITED KINGDOM	D. A. JOSLIN/ COMPUTER SERVICES/ HULL COLLEGE OF HIGHER EDUCATION/ INGLEMIRE AVE/ HULL ENGLAND HU6 7LJ/ UNITED KINGDOM/ (0482) 42157
IP5 7RE	UNITED KINGDOM	ROBERT KIRKBY/ 77 FLOOR 2 - R8.1.1/ RES D MARTLESHAM HEATH/ IPSWICH ENGLAND IP5 7RE/ UNITED KINGDOM/ IPSWICH 642 082
KT10 9EZ	UNITED KINGDOM	ATTN: PULSE TRAIN TECHNOLOGY LTD./ 15 LAKESIDE DR./ ESHER SURREY KT10 9EZ/ UNITED KINGDOM
KT22 9NF	UNITED KINGDOM	P. [illegible]. WALWYN/ LITTLE GABLES/ BELLLANE / FETCHAM/ SURREY ENGLAND KT22 9NF/ UNITED KINGDOM
LE1 9BH	UNITED KINGDOM	B. E. BARKER/ COMP. CENTRE/ LEICESTER POLYTECHNIC/ P.O. BOX 143/ LEICESTER ENGLAND LE1 9BH/ UNITED KINGDOM
LL57 1UT	UNITED KINGDOM	DAYFDD ROBERTS/ COMPUTING LABORATORY/ U.C.N.W./ BANGOR/ GWYNEDD WALES LL57 1UT/ UNITED KINGDOM
M13 9PL	UNITED KINGDOM	M. A. PELL/ DEPT. OF COMMUNITY MEDICINE/ UNIV. OF MANCHESTER/ OXFORD ROAD/ MANCHESTER ENGLAND M13 9PL/ UNITED KINGDOM/ 061-273 8241 X0 / X58
M13 9PL	UNITED KINGDOM	IAN ROBERT WILSON/ DEPT. OF COMPUTER SCI./ UNIVERSITY OF MANCHESTER/ OXFORD ROAD/ MANCHESTER ENGLAND M13 9PL/ UNITED KINGDOM/ 061-273-7121
M32 9BH	UNITED KINGDOM	DAVID J. SKYRME/ ARNDALE HOUSE/ DIGITAL EQUIPMENT CO. LTD./ CHESTER ROAD/ MANCHESTER ENGLAND M32 9BH/ UNITED KINGDOM/ 061-865-8676
M60 1QD	UNITED KINGDOM	GERALD C. KEIL/ DEPT. OF EUROPEAN STUDIES/ UMIST/ P.O. BOX 88/ MANCHESTER ENGLAND M60 1QD/ UNITED KINGDOM/ 061-236 3311 X2261
M60 1QD	UNITED KINGDOM	D. J. LEGGE/ DEPT. OF PHYSICS/ U.M.I.S.T./ P.O. BOX 88/ MANCHESTER ENGLAND M60 1QD/ UNITED KINGDOM

```
NW6 6DL UNITED KINGDOM  STEPHEN G. S. PROUT/ 2 KESLAKE ROAD/ LONDON ENGLAND NW6 6DL/ UNITED KINGDOM/ 01 960 4270
OX1 2DL UNITED KINGDOM  J. N. PAINE/ ST. PETER'S COLLEGE/ OXFORD UNIV./ OXFORD ENGLAND OX1 2DL/ UNITED KINGDOM/ OXFORD 48436
RG6 2LH UNITED KINGDOM  ROGER P. WRIGHT/ 16 RAGGLESWOOD CLOSE / EARLEY/ READING BERKS. RG6 2LH/ UNITED KINGDOM/ READING 663178
SE1 OTE UNITED KINGDOM  F. BOEUF/ HIGHWAY ENGR COMPUTER BRANCH/ ROOM 3/05 - ST. CHRISTOPHER HOUSE/ DEPT OF TRANSPORT/ SOUTHWARK STREET/ LONDON ENGLAND SE1 OTE/ UNITED KINGDO
                                             01 928 7999 X3026
SG1 2DY UNITED KINGDOM  JOE B. MONTGOMERY/ ICL/ CAVENDISH RD/ STEVENAGE HERTS SG1 2DY/ UNITED KINGDOM/ 0438 3361
S09 5NH UNITED KINGDOM  J. I. GOODSON/ DEPARTMENT OF MATHEMATICS/ COMPUTER STUDIES GROUP/ THE UNIVERSITY/ SOUTHAMPTON ENGLAND S09 5NH/ UNITED KINGDOM/ 0703-559122 X2387
S09 5NH UNITED KINGDOM  MIKE J. REES/ DEPT. OF MATHS./ COMPUTER STUDIES GROUP/ THE UNIVERSITY/ SOUTHAMPTON ENGLAND S09 5NH/ UNITED KINGDOM
ST16 2AJ UNITED KINGDOM S. STRUDWICK/ R.W. HOURD & SON LTD/ 7-8 MILL ST./ STAFFORD ENGLAND ST16 2AJ/ UNITED KINGDOM/ 0785-44221
ST7 1TL UNITED KINGDOM  D. K. MESSHAM/ I.C.L./ WEST AVENUE / KIDSGROVE/ STOKE-ON-TRENT STAFFS ST7 1TL/ UNITED KINGDOM/ (0782) 29681
SW7 2BY UNITED KINGDOM  P. DAVID ROSE/ DEPT. OF CHEMICAL ENGINEERING/ IMPERIAL COLLEGE / PRINCE CONSORT ROAD/ LONDON ENGLAND SW7 2BY/ UNITED KINGDOM
SW7 2BZ UNITED KINGDOM  PETER W. THROSBY/ DEPT. OF COMPUTING & CONTROL/ IMPERIAL COLLEGE/ QUEENSGATE/ LONDON ENGLAND SW7 2BZ/ UNITED KINGDOM/ 01-589 5111 X2742
S10 2TN UNITED KINGDOM  L. V. ATKINSON/ DEPT OF APPLIED MATH AND COMP SCIENCE/ UNIV. OF SHEFFIELD/ SHEFFIELD ENGLAND S10 2TN/ UNITED KINGDOM
TW11 OLW UNITED KINGDOM I. GOODE/ NATIONAL PHYSICAL LABORATORY/ DNACS/ TEDDINGTON MIDDLESEX TW11 OLW/ UNITED KINGDOM/ 01-977 3222
WC1N 3D4 UNITED KINGDOM ATTN: COMPUTER ANALYSTS & PROGRAMMERS/ 14-15 GREAT JAMES STREET/ LONDON ENGLAND WC1N 3D4/ UNITED KINGDOM/ 01-242-0021
YU-61000 YUGOSLAVIA     ATTN: FNT - ODDELEK ZA KEMIJO/ KNJIZNICA/ MURNIKOVA 6/ LJUBLJANA YU-61000/ YUGOSLAVIA
```

ANN S. ADAMS	08033	
HARRY S. ADAMS	91301	
KENNETH LEROY ADAMS	47907	
RICHARD E. ADAMS	43229	
KIM ADELMAN	55414	
KARL P. ADEY	07932	
JAMES L. AGIN	90278	
ARVIND AGRAWAL	92805	
THOMAS J. AHLBORN	19380	
JOHN AHLSTROM	95014	
PETER C. AKWAI	D-6000	GERMANY
JACK D. ALANEN	44106	
K. M. ALBRIGHT	30342	
ROSS ALEXANDER	POJ 1KO	CANADA
ROBERT ALKIRE	90260	
DAVID M. ALLAN	66102	
ROSS ALLARDYCE	94402	
BRUCE ALLEN	01810	
D. R. ALLUM	EX4 4QL	UNITED KINGDOM
RICHARD ALRUTZ	14580	
RICH ALTMAIER	94086	
FRANK ALVIANI	60626	
ALLEN L. AMBLER	95125	
S. AMBLER	DD2 1SJ	UNITED KINGDOM
CHARLES ANDERSON	08816	
FRANK ANDERSON	85061	
GARY S. ANDERSON	98043	
OLE ANDERSON	97330	
RICHARD L. ANDERSON	80306	
RON ANDERSON	55455	
WILLIAM F. ANDERSON	94133	
DEREK F. ANDREW	S7H 1B5	CANADA
DENNIS S. ANDREWS	94086	
DAVID ANDRUS	80301	
F. ANTONIO	92121	
PETER A. ARMSTRONG	98115	
BOB ARNOLD	55337	
GEORGE N. ARNOVICK	95927	
CHARLES N. ARROWSMITH	14850	
PETER R. ATHERTON	63166	
L. V. ATKINSON	S10 2TN	UNITED KINGDOM
NTION: COLIN G. CAMPBELL	77001	
NTION: C. V. D. WIJGAART		THE NETHERLANDS
TENTION: DIANNE CAMERON	M3C 1Z3	CANADA
ATTENTION: D. L. MYERS	80639	
ATTENTION: FRED BEVENSEE	75236	
ATTENTION: JUDY BISHOP	2001	SOUTH AFRICA
ATTENTION: LARRY LEWIS	90045	
ATTENTION: NANCY BROOKS	93111	
ATTENTION: ROY W. FILEGER	32204	
ATTENTION: WILLIAM MAIN	01945	
ATTN: ADP CENTER	50011	
TN: AIR FORCE WEAPONS LABORATORY	87117	
ATTN: AMERICAN COMPUTER SERVICES	90801	
ATTN: BIBLIOTHEEK 05627		THE NETHERLANDS
ATTN: BIBLIOTHEQUE CENTRALE		BELGIUM
ATTN: COMPUTATION CENTER	78363	
COMPUTER ANALYSTS & PROGRAMMERS	WC1N 3D4	UNITED KINGDOM
ATTN: DB/DC SOFTWARE ASSOC.	03103	
ATTN: DEPT. OF COMP. SCI.	3168	AUSTRALIA
ATTN: DEPT. OF MATHEMATICAL SCI.	P7B 5E1	CANADA
ATTN: D. M. MOFFETT	50307	
ATTN: FNT - ODDELEK ZA KEMIJO	YU-61000	YUGOSLAVIA
ATTN: HONEYWELL BULL S.A.	CH-8021	SWITZERLAND
ATTN: INFORMATION CENTER	01742	
ATTN: INFORMATION/RESOURCE CENTER	68588	
ATTN: J. M. P. ASSOCIATES	22030	
ATTN: KINDLER ASSOCIATES INC.	02142	
ATTN: LARRY MARKWORTH - LIBRARIAN	90404	
ATTN: LIBRARY		NORWAY
ATTN: LJS COMPUTER SERVICES	10530	
TN: OREGON MINI-COMPUTER SOFTWARE INC*	97201	
ATTN: OSTFOLD D H. LIBRARY	N-1750	NORWAY
ATTN: PERIODICALS SECTION	0001	SOUTH AFRICA
ATTN: PROGRAM LIBRARIAN	5001	AUSTRALIA
ATTN: PROGRAM LIBRARY	N6A 5B7	CANADA
ATTN: PULSE TRAIN TECHNOLOGY LTD.	KT10 9EZ	UNITED KINGDOM
ATTN: REGENSTRIEF INSTITUTE	46202	
ATTN: RUTH SUGARMAN	95014	
ATTN: SAM CALVIN	09175	
ATTN: SERIAL RECORDS	49008	
ATTN: SERIAL RECORDS DIV.	94305	
ATTN: TECHNICAL LIBRARY	01821	
ATTN: TECHNICAL LIBRARY 47-687	53202	
ATTN: TIMESHARING BUSINESS SYSTEMS	95128	
ATTN: UNIV. OF WASHINGTON	98195	
ATTN:INFORMATICS INC. BOOKSTORE	20852	
ATTN:WESTERN RESERVE COMMUNICATIONS	44512	
H. J. AU	A1C 5M3	CANADA
CHUCK AUGUSTINE	15213	
DAVID AULT	22101	
LEE D. AURICH	91711	
AYERS LOCKSMITHING	94941	
DAVE BAASCH	97210	
GUS BACOYANIS	90403	
ROBERT M. BAER	94941	
DONALD R. BAIN	M9A 3V3	CANADA
S. BALASUBRAMANIAN	77001	
LYNNE J. BALDWIN	68182	
L. DAVID BALDWIN	03055	
MICHAEL S. BALL	92152	
FRED BALLANTINE	98161	
RICHARD BALOCCA	61801	
STANLEY E. BAMMEL	75116	
DEAN BANDES	01741	
B. E. BARKER	LE1 9BH	UNITED KINGDOM
GORDON BARKER	N6A 5B7	CANADA
DAN +ROBIN BARNES	95926	

JAMES G. BARON	98105	
ART BARRETT	22312	
BRUCE A. BARRETT	94066	
RANDY BARTH	20810	
S. J. BATTORY JR.	01247	
ELMER T. BEACHLEY	15236	
FRANCIS H. BEARDEN	45241	
CHARLES R. BEAUREGARD	90712	
JAMES E. BECKLEY	60603	
RICHARD BEELER	92802	
C. Y. BEGANDY	15069	
LOUIS N. BELEOS	89502	
FRED BELLOMY	93120	
LEE A. BENBROOKS	90049	
NIGEL W. BENNEE		THE NETHERLANDS
BARBARA BERGER	10014	
SERGIO BERNSTEIN	87112	
ROBERT W. BERRY	87107	
SCOTT S. BERTILSON	55455	
JAMES L. BEUG	93407	
KASI SESHADRI BHASKAR	98020	
DAVE BIANCHI	55455	
ROB BIDDLECOMB	21203	
JEFFREY H. BIGGERS	30305	
DEAN BILLING	95401	
LARRY BILODEAU	97330	
JOHN BLACKWOOD	95660	
BRAD BLASING	55455	
LYNN BLICKENSTAFF	90065	
PETER BLONIARZ	12222	
P. VAN EMDE BOAS	1018 WB	THE NETHERLANDS
HAROLD L. BOERLIN II	55435	
F. BOEUF	SE1 OTE	UNITED KINGDOM
J. BOGAR	21701	
LARRY D. BOLES	37076	
RAFAEL M. BONET		SPAIN
TIM BONHAM	55454	
WILLIAM R. BONHAM	89511	
GARY J. BOOS	69341	
JOHN BORGELT	94025	
RONALD V. BOSSLET	02154	
PERE BOTELLA		SPAIN
SPEC BOWERS	21030	
KEN BOWLES	92067	
GORDON C. BOWRON	L1S 3B4	CANADA
CHRIS BOYLAN	55042	
ROBERT BOYLAN	08540	
ALEX BRADLEY	92714	
PAUL BRAINERD	55454	
JOE B. BRAME JR.	84105	
DAVID E. BREEDING	75234	
H. DICK BREIDENBACH	48033	
BILL BRENNAN	19401	
FRANK BREWSTER	16701	
WILLIAM A. BRIGGS	60010	
WILLIAM D. BRISCOE	07762	
C. T. BRITTON		UNITED KINGDOM
CHARLES L. BROOKS	02139	
BRUCE A. BROWN	92805	
MICHAEL D. BROWN	17331	
RICHARD T. BROWN	97207	
CHARLES H. BROWNING	10022	
C. H. BROWNING	10022	
JERRY F. BRUMBLE	90278	
HERBERT M. BRYANT JR.	33549	
S. R. BUCHANAN	53092	
ANNA BUCKLEY	47401	
JOHN BUCZEK	97850	
DAVID M. BULMAN	92123	
WILHELM BURGER	78712	
THOMAS K. BURGESS	77302	
DONALD D. BURN	01752	
JOHN W. BURNETT	95051	
ROGER F. BURROWS	K2H 5S3	CANADA
VINCENT BUSAM	95030	
R. BUSH	97459	
RICHARD A. BYERS	46205	
DAVID CALCATELLI	85002	
J. D. CALLAHAN	84108	
M. DIANNE CAMERON	M4R 1Z2	CANADA
GEORGE P. CAMPBELL	08052	
STEVEN CAMPBELL	03768	
MAL CAREY	04473	
T. A. CARGILL	N2C 3E0	CANADA
THERON D. CARLSON	80401	
JOHN CARNAL	80201	
MARIANN CARPENTER	14853	
EDDIE CARRIE	91311	
TERRENCE M. CASELLA	90069	
WAYNE CATLETT	53202	
AVERY CATLIN	22901	
D. A. CAUGHFIELD	79601	
DAVID F. CAUTLEY	97330	
STEVE CAVENDER	76059	
F. A. CELLINI	N2J 4G5	CANADA
MIKE CHALENBURG	56381	
G. CHALIFOUR	L7M 1K4	CANADA
JOHN L. CHANEY	72205	
P. & C.F.BLOMKE CHANG	91359	
JOHN P. CHAPMAN	64110	
BILL CHESWICK	19122	
LAURIAN M. CHIRICA	93106	
CARLOS CHRISTENSEN	02176	
KENNETH L. CHRISTENSEN	61701	
WILLIAM G. CHRISTIAN	30305	
DAVID B. CHRISTIE	60202	
I. N. CHRISTOFFERSON	7700	SOUTH AFRICA
GERALD W. CICHANOWSKI	55987	
RICHARD J. CICHELLI	18103	
JON F. CLAERBOUT	94305	
DONALD L. CLAPP	47272	
IRA A. CLARK	10024	

W. E. CLARK	92138	
JOHN C. CLARSON	23669	
M. B. CLAUSING	45424	
RICHARD CLAYTON		UNITED KINGDOM
JOE CLEMA	45432	
DAVID G. CLEMANS	90064	
DAVID C. CLINE	01581	
ROBERT COLE	18017	
PETER CONKLIN	01451	
DANIEL F. CONWAY	95132	
H. A. COOK	20903	
T. J. COOK	87544	
STEVEN L. COOL	01880	
RICH COON	01581	
DAVID A. COOPER		THE NETHERLANDS
NICK COPPING	91103	
JOHN J CORCORAN 3RD. III	87107	
RICHARD CORE	94088	
MARCUS C. CORNELL	96224	
THOMAS CORRIGAN	60684	
FRED COTTON	02174	
JAMES C. COZZIE	52402	
FRANK F. CRANDELL	92635	
JOHN EARL CRIDER	77043	
E. CRITTSINGER JR.	23505	
ELIZABETH CROCKER	94546	
WILLIAM E. CROSBY	92704	
DAVID B. CROUSE	15213	
HEODORE R. CROWLEY	01720	
JOHN P. CUCHES	30033	
HOWARD CUNNINGHAM	97077	
NICK CVETKOVIC	19172	
STEVE DALSIMER	90291	
JAMES A. DARLING	92027	
BRUCE DAVIDSON	M3C 1H7	CANADA
W. DAVIS	3046	AUSTRALIA
RICHARD L. DAY	19102	
J. F. DE BEER		SOUTH AFRICA
DAVID J. DE FANTI	02871	
ARTINE DE GERLACHE	B-1170	BELGIUM
HARD P. DE ROBERTS	30354	
JOHN DE ROSA JR.	01824	
JOHN L. DEBES	14502	
MICHAEL DEISEMROTH	47907	
ROBERT I. DEMROW	01810	
TIMOTHY DENNIS	06035	
PAUL DENNISON	N6A 4K1	CANADA
EDWARD DEPPE	55404	
SHAUN DEVLIN	48010	
GEORGE B. DIAMOND	08826	
ROBERIO DIAS		BRAZIL
LAURA L DICKINSON	52240	
MARY DIEGERT	13902	
BOB DIETRICH	97077	
RICHARD F. DILLON	K1S 5B6	CANADA
J. W. DISSELKAMP	37660	
FRED DITTRICH	65201	
WILLIAM H. DIUGIUD	27702	
J. SCOTT DIXON	02138	
RANDY DODGE	A1C 5S7	CANADA
ALLEN F. DOWNARD	33803	
DORSEY DRANE	97202	
KENNETH R. DRIESSEL	74102	
JEFFREY J. DRUMMOND	55455	
LARRY DUBY	22209	
DOUGLAS DUNLOP	23185	
BOB DUPREE	74145	
DAVID DYCHE	77001	
PAUL T. DYKE	22151	
RICHARD R. DYMANT	98188	
JEFF EASTMAN	80537	
NORBERT EBEL	CH-2000	SWITZERLANI
FRITZ EBERLE	01852	
PETER ECCLESINE	94043	
GLENN T. EDENS	94086	
BRUCE J. EDMUNDSON	95051	
P. S. EDWARDS	3130	AUSTRALIA
ARNIE EGEL	95129	
BILL EHLERT	80201	
JOHN D. EISENBERG	48105	
~~HOWARD EISENSTEIN~~	~~29206~~	
VINCENT ELIAS	46312	
KEITH ELKIN	S-115 43	SWEDEN
DENNIS R. ELLIS	80303	
MONTE ELLIS	33142	
LARRY E. ELLISON	08046	
ROBERT J. ELLISON	13323	
AGNES H. ELMORE	23284	
ROBERT EMERSON	98040	
R. D. EMRICK	33601	
TOM ENTERLINE	20822	
GLENN ENTIS	10016	
E. W. ERRICKSON	85613	
R. L. ESHELMAN	80210	
R. ETZI	06830	
TOM EUBANK	40206	
HERMAN EUREMA	08540	
MICHAEL EVANS	S-145 71	SWEDEN
ROGER H. EVANS	92123	
RICHARD A. EVERMAN	92717	
R. NEIL FAIMAN JR.	48228	
JOSEPH R. FALKNER	88130	
SHAL FARLEY	91126	
JOSEPH H. FASEL III	47907	
WALT FEESER	78758	
MICHAEL B. FELDMAN	20052	
LINWOOD FERGUSON	22923	
SID FERMI	97077	

JEANNE FERRANTE	12561	
ALAN B. FINGER	02154	
LLOYD D. FINK	23505	
JIM FINN	L5C 1C8	CANADA
HERMAN FISCHER	91409	
P. E. FISCHER	D-8000	GERMANY
GLENN FISHBINE	55404	
GENE FISHER	92651	
RICHARD B. FITZ	20016	
ROBERT G. FITZGERALD	22180	
HANS FLACK	S-751 21	SWEDEN
JOSE M. FLOREN		DOMINICAN REP.
WAYNE FLOURNOY	77074	
RUDY L. FOLDEN	92714	
JIM FOLEY	91364	
WARREN C. FORDHAM	28214	
RICHARD FOULK	96786	
AL FRANCIS	9460	
	6	
LEE FRANK	08002	
STEPHEN D. FRANKLIN	92717	
ROBERT FRANKSTON -COPY A	02139	
ROBERT FRANKSTON -COPY B	02154	
ROGER W. FRECH	78753	
KARL FRYXELL	91125	
KOICHI FUKUNAGA	100	JAPAN
GLEN FULLMER	97123	
DAN FYLSTRA	02134	
DAVID JON FYLSTRA	94305	
I. GANAPATHY	V5A 1S6	CANADA
LAWRENCE M. GARCIA	10580	
JOHN T. GARDNER	93501	
PATRICIA J. GARSON	06851	
PATRICK D. GARVEY	90291	
DALE GAUMER	46808	
EDWARD F. GEHRINGER	47907	
TILL GEISER	D-7910	GERMANY
W. MORVEN GENTLEMAN	N2L 3G1	CANADA
P. U. GEORGE	90403	
THOMAS L. GERBER	57401	
DANIEL E. GERMANN	55455	
D. R. GIBBY		THE NETHERLANDS
ROBERT A. GIBSON	22110	
MIKE GILBERT	01752	
SHELLEY GILES	97301	
J. GILMER	94105	
CHARLES O. GIMBER	92024	
BRIAN GLASSER	10003	
MAURY GOLDBERG	13203	
LOUISE GOLDSTEIN	11713	
PHILLIP I. GOOD	49001	
I. GOODE	TW11 OLW	UNITED KINGDOM
EUGENE K. GOODELL	09403	
J. I. GOODSON	SO9 5NH	UNITED KINGDOM
ROBERT GOODWIN	60510	
GEORGE S. GORDON JR.	02173	
KEITH GORLAND	60174	
E. GOTTWALD	14450	
JOHN S. GOURLAY	48169	
DAVID GRABEL	02173	
DAVE GRAHAM	94086	
JEFFREY W. GRAHAM	78873	
WILLIAM Q. GRAHAM	19711	
THOMAS E. GRANVOLD	94086	
DAVID N. GRAY	78769	
JOHN W. GRAY	02747	
ARTIE GREEN	95050	
STEVEN J. GREENFIELD	90010	
RICHARD GREENLAW	43230	
TIMOTHY GRIESER	02215	
DONALD E. GRIMES	95014	
JOHN R. GRINDON	63042	
JAMES GROSSMAN	33181	
JOHN GULBENK	94596	
GARY R. GUTH	60204	
JOHN G. GUTHRIE	20904	
HARRY P. HAIDUK	79109	
JOEL M. HALPERN	55403	
DAVID E. HAMILTON	23502	
E. MICHAEL HAMILTON	20052	
RICHARD W. HAMILTON	94702	
TERRY HAMM	97077	
MIKE HAMMAN	49464	
NICK HAMMOND	2600	AUSTRALIA
J. W. HANCOCK	C1A 4P3	CANADA
ALAIN J. HANOVER	01887	
BRIAN HANSON	55110	
DAVID R. HANSON	85721	
MARC HANSON	90254	
PETER J. HARRINGTON	11552	
MARGERY HARRIS	02173	
MIKE HARRIS	62702	
HARSONO		INDONESIA
JAMES F. HART	02195	
ORVAL F. HART JR	87544	
RONALD HARTUNG	22401	
JOHN P. HARVELL	75081	
STEPHEN HATCH	01730	
LARRY HAWLEY	91103	
W. F. HAYGOOD	84121	
ANTHONY R. HEALY	11530	
PATRICIA HEATH	PL4 8AA	UNITED KINDOM
WILLIAM HEILAND	55402	
JESSE HEINES	01505	
DICK HEISEN	90401	
SVANTE HELLSING	S-145 71	SWEDEN
PAUL HELVIG	56301	
JOHN M. HEMPHILL	76019	
CARROLL HENNICK	91320	
CHRISTOPHER J. HENRICH	07724	
RICHARD HERBERT	95610	
JAMES T. HERINGER	91105	
CLINTON HERLEY	45244	
THEODORE J. HERRMAN		PANAMA
H. F. HESSION	22101	
LOREN L. HEUN	49006	
BRUCE HIBBARD	06497	
DAVID HICKOK	50158	
THOMAS C. HICKS	54601	
D. R. HILL	45342	
JOSEPH N. HILTON	72701	
MAX HINCHMAN	94709	
THOM HOARD	55414	
PHILIP T. HODGE	46322	
BOB HOFKIN	92093	
DAVID HOLLAND	77079	
STEPHEN HOLLATZ	60559	
RALPH G. HOLLINGSWORTH JR.	43762	
JOHN B. HOLMBLAD	20005	
GEORGE E. HOLZ	07060	
DAN HOMER	83720	
GREGORY L. HOPWOOD	92714	
DAVID HORNBAKER	80203	
C. L. HORNEY	92803	
TOM HORSLEY	95376	
THOMAS P. HOVEKE	60618	
JAMES H. HOWARD	49931	
CAROL B. HOWELL	20770	
HUGO HSIUNG	60025	
GARY HUCKABAY	73501	
THOMAS W. HUEBNER	53201	
JON F. HUERAS	01945	
JAMES W. HUFFMAN	95014	
JAMES S. HUGGINS	77081	
GENE HUGHES	78231	
PHIL HUGHES	98507	
ALFRED J. HULBERT		
ALICE HUNT	92521	
DAVID HUSNIAN	73106	
STEPHEN G. HUSSAR	15222	
JOHN HUTCHINSON	E11 1QL	UNITED KINGDOM
LYNN C. HUTCHINSON	44092	
ELIZABETH IBARRA	91360	
ALAN J. ILIFF	60660	
CRAIG M. INGLIS	30021	
ARON K. INSINGA	01752	
AVRUM ITZKOWITZ	61820	
KENNETH K. IWASHIKA	90815	
GEORGE T. JACOBI	53201	
NORMAN J. JAFFE	V6K 2C1	CANADA
FERNANDO JAIMES		MEXICO
DEAN JAMES	33068	
SCOTT JAMESON	95014	
ROBERT L. JARDINE	92691	
GEORGE D. JELATIS	55417	
HAROLD D. JENKINS JR.	22151	
JOHN JENKINSON	75006	
GREGORY JENNINGS	53207	
JEFFREY C. JENNINGS	59801	
AUTHOR R. JETER	85028	
EGON JOHANSSON	S-721 83	SWEDEN
JAN-HENRIK JOHANSSON	SF-00510	FINLAND
GERALD C. JOHNS	63110	
CLARA L. JOHNSON	33528	
JUSTINA JOHNSON	10570	
MARK R. JOHNSON	T6G 2C2	CANADA
SUE JOHNSON	87545	
DAN B. JOHNSTON	4067	AUSTRALIA
RICHARD A. JOKIEL	19518	
DAVID TERRY JONES	94086	
D. A. JOSLIN	HU6 7LJ	UNITED KINGDOM
JOSEPH M. JOYCE	64110	
MARK JUNGWIRTH	93010	
SAMUEL C. KAHN	19898	
ALAN M. KANISS	19111	
FRED KATZMAN	80203	
HUGH M. KAWABATA	93106	
MATTHEW KAZLAVSKAS	12345	
JOE KEEFE	95014	
GENE KEENOY	07083	
GERALD C. KEIL	M60 1QD	UNITED KINGDOM
BOB KELLER	19317	
W. A. KELLEY	90274	
GINGER KELLY	77001	
WALLACE KENDALL	21043	
N. KERMAN	11714	
HENRY D. KERR III	30067	
MARK C. KERSTETTER	49008	
GURUPREM SINGH KHALSA	91101	
DENNIS F. KIBLER	92627	
TIM KIEFFER	02115	
JOHN H. KILFOIL	95126	
DANIEL R. KILLORAN	02139	
ROBERT J. KING	15221	
ROBERT KIRKBY	IP5 7RE	UNITED KINGDOM
JAMES KLAJA	60657	
HENRIETTE KLAWANS	60613	
HEINZ KLEENE	D-3000	GERMANY
BARCLAY R. KNERR	92646	
D. L. KNITTEL	92121	
JULIANA M. KNOX	95051	
EDWARD W. KNUDSEN	21204	
MIKE KNUDSON	01752	
DENNIS KODIMER	85260	
RICHARD A. KOEBBING	77024	
HARRIS M. KOEHN	60514	
KARL KOEHNE	D-5000	GERMANY
KURT KOHLER	97330	
DENIS KOMINSKY	01701	
ALAN A. KORTESOJA	48103	
RICHARD KRASIN	01886	
G. M. KREMBS	12401	
STUART J. KRETCH	65201	
DIRK KRONIG	D-7750	GERMANY
H. M. KUHLMANN	98115	
DARRYL KUHNS	89503	
MARVIN E. KURTTI	35801	
RICHARD J. KWAN	90406	
ARTHUR LACROIX LA CROIX	06460	
JOSEPH LACHMAN	60077	
RICHARD M. LADDEN	95035	
DAN M. LALIBERTE	55812	
MARY K. LANDAVER	92110	
LARRY D. LANDIS	64108	
H. LASHLEE	91030	
LUC LAVOIE	HC3 3J7	CANADA
THOMAS W. LAWHORN	80917	
GARY E. LAWRENCE	94501	
ED LEARY	10001	
WILLIAM G. LEDERER	48103	
VICTOR LEDIN	94127	
FRANCIS F. LEE	02139	
R. GARY LEE	46805	
TOM LEE	49684	
D. J. LEGGE	M60 1QD	UNITED KINGDOM
TOM LEGRAZIE	43778	
CLARENCE LEHMAN	55364	
HEIKKI LEHTINEN	SF-02730	FINLAND
RAYMOND M. LEONG	95051	
FRANK LEPERA	11973	
ALAN M. LESGOLD	15260	
MICHAEL H. LESKIN	10021	
HOWARD LEVERENZ	77023	
CHARLES T. LEWIS	18016	
DANIEL LEY	11432	
FELIX S. H. LI	77055	
PING K. LIAO	94545	
BOB LIDRAL	19020	
KARL LIEBERHERR	87131	
JACK LIEBSĆHUTZ	60614	
THOMAS L. LIGHT	80027	
TERRY LIITTSCHWAGER	97402	
JOHN E. LIND	55414	
GORAN LINDAHL	S-902 36	SWEDEN
STEPHEN LOCKE	53511	
MYRON C. LONG	90732	
JOHN DE LONGPRE	49503	
KENT LOOBEY	97401	
TOM LOVE	20850	
R. A. LOVESTEDT	98055	
JAMES R. LOW	14627	
TIM LOWERY	92627	
HOUSTON P. LOWRY	06032	
JOHN LOWRY	92110	
R. C. LUCKEY	99352	
WILLIAM LUITJE	48103	
RICHARD C. LUND	94114	
RICHARD G. LYMAN	84116	
GILL LYTTON	90066	
I. R. MAC CALLUM	CO4 3SQ	UNITED KINGDOM
LYNN MACEY	67460	
BRUCE MACKENZIE	01754	
PETER H. MACKIE	97005	
BARRIE D. MACLEOD	H9P 1J3	CANADA
IAN MACMILLAN	H3N 2T6	CANADA
JIM MADDEN	92093	
DAVID C. MADSEN	60658	
MARIO MAGIDIN		MEXICO
JAMES P. MAGNELL	03755	
RICHARD L. MAHN	43147	
DARYL E. MALENA	68154	
RAJ MALHOTRA	92041	
KAMRAN MALIK	97331	
T. M. MALIN	84109	
WESLEY E. MANGUS	48098	
J. F. P. MARCHAND	48824	
THOMAS A. MARCINIAK	20853	
RICK L. MARCUS	55455	
CHRIS D. MARLIN	5001	AUSTRALIA
GAYE MARR	01867	
BILL MARSHALL	03060	
DOUG MARSHALL	M2J 2W6	CANADA
HOWARD S. MARSHALL JR.	33065	
JON MARSHALL	97077	
BERND MARTENS	D-1000	GERMANY
M. MARVINNEY	44106	
GEORGE MASSAR SR	91367	
FRED A. MASTERSON	19711	
ROBERT J. MATHIAS JR	48043	
TOM MATHIEU	99352	
S. B. MATTHEWS	L5N 1W2	CANADA
WERNER G. MATTSON	91020	
J. M. MCCAIG		UNITED KINGDOM
JOHN W. MCCAIN	63367	
JOHN C. MCCALLUM	M3J 1P3	CANADA
JOHN K. MCCANDLISS	63188	
DONALD H. MCCLELLAND	90802	
JIM MCCORD	93017	
JOEL MCCORMACK	92008	
PAUL L. MCCULLOUGH	97077	
DAVID P. MCDONNELL	76107	
HENRY MCGILTON	95051	
JOHN P. MCGINITIE	94304	
JAMES A. MCGLINCHEY	19044	
MICHAEL R. MCGUIRE	98178	
L. MCHARDY	N6A 5B9	CANADA
J. W. MCINTOSH	55901	
JOHN MCMANUS JR.	90230	
KENNETH M. MCMILLIN	49931	
P. D. MCMORRAN	KOJ 1PO	CANADA
DAVID MCQUEEN	35803	
RODNEY MEBANE	19047	
WILLIAM MEIER	07009	
PAUL MEILAND	44106	
BERT MENDELSON	01060	
R. L. MERCER	90045	
D. K. MESSHAM	ST7 1TL	UNITED KINGDOM
WILLIAM R. METZ	45201	
BOB METZGER	48640	
KURT METZGER	48105	
DAVID MEYER	97401	
JOSEPH A. MEZZAROBA	08854	
COLIN MIEROWSKY		SOUTH AFRICA
C. A. MILLER	V6T 1W5	CANADA
DAVID MILLER	94042	
DEAN MILLER	94025	
JOHN MILLER	99163	
MARTIN MILLER	H3S 2L7	CANADA
PAUL MILLER	94109	
VICTOR S. MILLER	10598	
JUDITH MINAMIJI	90249	
JAMES F. MINER	55455	
DENNIS MISENER	B3L 4L5	CANADA
EDWARD E. L. MITCHELL	01742	
KEITH MITCHELL	98008	
W. MITCHELL	K2K 1X4	CANADA
ROBERT H. MIX JR.	95610	
JESSE D. MIXON	75961	
D. A. MOIR	R3H 0R9	CANADA
ROLF MOLICH	DK-2730	DENMARK
MORRIS MOLIVER	11566	
JOHN MONTAGUE	87545	
EUGENE P. MONTGOMERY	91342	
JOE B. MONTGOMERY	SG1 2DY	UNITED KINGDOM
G. D. MONTILLON	45215	
A. D. MOORE	66030	
JAMES L. MORAN	10005	
CHARLES ROBERT MORGAN	02138	
CLEMENT MORITZ	75088	
R. A. MORRIS	02125	
HERBERT E. MORRISON	90266	
DAN MORTON	19117	
DON MOXON	95014	
CHARLES F. MURPHY	94618	
JOHN MURRAY	95129	
TERRY MYHRER	55066	
OLAV NAESS	N-5000	NORWAY
BOBBY OTIS NASH	63105	
RICHARD J. NAST	32670	
JOHN NAUMAN	55414	
PETER A. NAYLOR	19422	
DAVID NEAL	07724	
THOMAS M. NEAL	92634	
PETER NEEDHAM	V6X 2L4	CANADA
R. CARLYLE NEELY JR.	20022	
GREGORY L. NELSON	94040	
JOHN E. NEWTON	78148	
ROBERT C. NICKERSON	94611	
RANDY NIELSEN	94702	
STUART C. NIMS	90266	

JOHN NOLAN	20755	
TERJE NOODT		NORWAY
KATIE NOONING	78664	
BARBARA K. NORTH	13760	
BARTON F. NORTON	21045	
FRANK NUSSBAUM	60626	
DAVE NUTTING	60005	
WILLIAM J. NYBACK	60606	
DAVID F. OHL	95014	
CHRISTOPHER OHLAND	94105	
DAVID E. OLAVSSEN	N-3000	NORWAY
A. OLDENBURG	53218	
ERIC OLSEN	92714	
RON OLSEN	80234	
ROBERT OSBORN	02178	
RONALD OTTO	22310	
JORGEN OXENBOLL	DK-2100	DENMARK
GREGORY J. O'BRIEN	02181	
JAMES W. O'CONNOR	07632	
MIKE O'DELL	73070	
G. O'SCHENECTADY	12202	
CLINTON PACE	95650	
R. K. PAETZOLD	08101	
F. G. PAGAN	62901	
JACK PAGE		SINGAPORE
J. N. PAINE	OX1 2DL	UNITED KINGDOM
W. O. PAINE	91103	
L. PAINTER	84737	
RICHARD PALCHIK	95014	
JEFF PALMER	64118	
PAUL J. PANTANO	19145	
T. L.(FRANK) PAPPAS	19083	
TED C. PARK	92408	
DENNIS PAULL	94022	
THOMAS J. PAULSON	92630	
FRANK PAVLIK	10533	
ERIC PEABODY	75042	
DONALD D. PECKHAM	92713	
FLEMMING PEDERSEN	DK-2770	DENMARK
M. A. PELL	M13 9PL	UNITED KINGDOM
RUSSELL J. PEPE	07207	
HAL PERKINS	14853	
WALT PERKO	55102	
ROBERT C. PERLE	08753	
JODY PAUL PERONI	02154	
ANDREW L. PERRIE	54901	
DAVID L. PETERSON	56301	
W. WESLEY PETERSON	96822	
CHRISTOPHER A. PHILLIPS	48093	
PAUL PICKELMANN	48109	
DAVID PICKENS	80302	
WILL PICKLES	AL3 4RZ	UNITED KINGDOM
STEPHEN PIKE	60419	
NORMAN V. PLYTER	14420	
FELIX POPPLEIN	D-8000	GERMANY
ANN PORCH	94137	
JOHN G. POSA	10020	
DAVID R. POSH	48184	
WALTER L. PRAGNELL	02149	
DARRELL PREBLE	30303	
KENNETH A. PRESCOTT JR.	92714	
CHARLES R. PRICE	80301	
RICHARD E. PRICE	60630	
WILLIAM C. PRICE	97068	
WILSON T. PRICE	94619	
STEPHEN G. S. PROUT	NW6 6DL	UNITED KINGDOM
JOHN L. PRUN	90630	
ERIC PUGH	90024	
CHARLES J. PURCELL	55113	
JAMES L. PYLES	02181	
STEVE QUALLINE	13210	
RAYMOND QUIRING	90019	
BILL RAEUBER	29210	
M. RAHILLY	3072	AUSTRALIA
STEVEN R. RAKITIN	07462	
STEVEN B. RAKOFF	7405	SOUTH AFRICA
JOHN F. RATTI	19128	
LINDA LEA RAY	68131	
EDWARD K. REAM	53705	
DAN REED	72554	
MIKE J. REES	SO9 5NH	UNITED KINGDOM
PHYLLIS A. REILLY	90277	
STEVEN A. REISMAN	55455	
PETER RENNICK	10023	
KEN RENWORTH	94086	
EDRICE REYNOLDS	98407	
ROBERT J. REYNOLDS	92138	
D. LLOYD RICE	90406	
PETER RICHETTA	16057	
DAN RICHMOND	92103	
GEORGE H. RICHMOND	80027	
LORIN RICKER	97225	
CHARLES RIDER	91326	
ROBERT W. RIEMANN	98112	
CARROLL B. ROBBINS JR.	28704	
DAYFDD ROBERTS	LL57 1UT	UNITED KINGDOM
JOE C. ROBERTS	75042	
TERRY R. ROBERTS	80202	
PARLEY P. ROBINSON	84602	
PEGGY ROBLYEN	32304	
BOB ROGERS	20855	
GORDON W. ROMNEY	84010	
JOHN ROSCOE		UNITED KINGDOM
P. DAVID ROSE	SW7 2BY	UNITED KINGDOM
SAUL ROSEN	47907	
DAVID ROSENBOOM	M5N 2Z6	CANADA
A. FREDERICK ROSENE	02194	
ESTHER ROSENSTOCK	11374	
D. S. H. ROSENTHAL	EH1 1GZ	UNITED KINGDOM
MICHAEL ROSIAK	19446	
BERNIE ROSMAN	01701	
PHILIP W. ROSS	18914	
RICHARD ROTH	06468	
JEAN-CLAUDE ROY	95127	
RUSSELL RUBY	97330	
BEARDSLEY RUML II	02146	
HOWARD RUMSEY JR.	91105	
MARTIN RUNYAN	60120	
LESTER SACHS	21235	
DAVID W. SALLUME	95051	
E. J. SAMMONS	75080	
MARCELO SANSEAU	RA-8000	ARGENTINA
LES SATENSTEIN	H3C 3A9	CANADA
LYNN SAUNDERS	97077	
ABRAHAM SAVITZKY	06856	
C. W. SAWYER	46201	
JAY SAX	90278	
PHILIP H. SAYRE	90045	
BOB SCARLETT	87545	
S. R. SCHACH	7700	SOUTH AFRICA
ANTHONY J. SCHAEFFER	49085	
IRVING S. SCHECHTMAN	08540	
R. S. SCHLAIFER	91030	
IAN SCHMIDT	46526	
WARREN SCHODER	07960	
ED SCHOELL	95051	
JAMES R. SCHRAGE	08854	
JAY SCHUMACHER	80302	
ROBERT SCHUTZ	11756	
FRANK SCHWARTZ	02173	
JOHN SCOBEY	55419	
CHARLIE SCOGIN	75229	
DAVID L. SEARLE	55441	
BRUCE S. SEELY	91343	
NORM SEETHOFF	98043	
JOHN SEITZ	JOB 2C0	CANADA
GERALD P. SHABE	22309	
JIM SHALLOW	19004	
BILL SHANNON	44107	
JACK P. SHAW	53012	
GARY B. SHELLY	92631	
PATRICIA SHELLY	20852	
CHARLES F. SHELON	76133	
BOB SHEPARDSON	95014	
FREDERICK E. SHIPLEY JR.	15230	
KIM L. SHIVELEY	75231	
LAWRENCE A. SHIVELY	45414	
ALBERT SHPUNTOFF	51106	
W. A. SHULL	45424	
BOB SIEGEL	11215	
STEVEN SIEGFRIED	55113	
WEBB SIMMONS	92111	
T. R. SIMONSON	94105	
MIKES SISIOS	95053	
CHARLES SISKA JR.	90405	
STEPHEN SKEDZELESKI	91103	
DAVID J. SKYRME	M32 9BH	UNITED KINGDOM
ROBERT C. SLATE	98115	
P. A. SLATS	2501 BD	THE NETHERLANDS
G. THOMAS SLUSSER	53217	
WARD SLY	55443	
SID SMART	61701	
JOSEPH W. SMITH	92127	
LYLE B. SMITH	60115	
RICHARD M. SMITH	03102	
REID SMITH-VANIZ	06430	
CRAIG A. SNOW	92138	
DAVID V. SOMMER	21401	
I. D. SOUTHWELL	94402	
FRANK S. SPARKMAN	30305	
JON L. SPEAR	55414	
PETER T. SPECK	D-6900	GERMANY
RICHARD P. SPRAGUE	92714	
PAUL SPRECHER	10024	
JIM SQUIRES	92660	
RICHARD A. STACK	60604	
VINCENT STANFORD	20037	
MICHAEL STAUFFER	22801	
ROD STEEL	97077	
LARRY STEIN	07067	
MIKE STEIN	56320	
PETER STEIN	01890	
JAMES STEINBERG	02142	
TIM STEVENS	60106	
T. Q. STEVENSON	20250	
O. ARTHUR STIENNON	53715	
R. A. STILLMAN	94043	
A. I. STOCKS	95051	
J. P. M. STOFBERG	19422	
WIBERTA STONE	93003	
THOMAS J. STOODLEY III	01824	
DON STOVER	52302	
JOHN M. STRAYHORN	02139	
S. STRUDWICK	ST16 2AJ	UNITED KINGDOM
ALASDAIR D. STUART	2191	SOUTH AFRICA
JERRY W. SUBLETT	94086	
CONRAD SUECHTING	74145	
J. MICHAEL SULLIVAN	61101	
GENE A. SUMNER	85008	
ASHOK SURI	94040	
DENNIS SUTHERLAND	52302	
MARY SUTTON	H4T 1N1	CANADA
ALAN H. SWANN	95955	
TOM SWANSON	99507	
G. B. SWARTZ	07764	
BEVERLY SWISSHELM	40583	
MYRON R. SYPHUS	84014	
KEITH G. TAFT	94087	
RYUJI TAKANUKI	244	JAPAN
JAMES E. TARVID	53927	
PAUL TEICHOLZ	94708	
C. J. THODAY	CV21 2QE	UNITED KINGDOM
RON THOMAS	55435	
JAMES B. THOMPSON JR.	07054	
R. C. THORNTON	90631	
PETER W. THROSBY	SW7 2BZ	UNITED KINGDOM
T. R. THURMAN	52302	
MIKE TILLER	55116	
RON TIPTON	64134	
CATHERINE C. TOBEY	91335	
ROBERT H. TODD JR	19020	
ANTHONY TOOGOOD	10021	
WILLIAM D. TORCASO	01002	
P. TORGRIMSON	94087	
CARL J. TOSETTO	75205	
GREGG TOWNSEND	85721	
ARON SHTULL TRAURING	15217	
MIKE TRAVIS	95051	
JOHN TROTTER	90240	
TOM A. TROTTIER	M4R 1V2	CANADA
JEAN TROUDT	80231	
JIM TSEVDOS	15213	
JYRKI TUOMI	SF-33540	FINLAND
ROBERT TUPPER	11968	
ROBERT L. TURPIN	76101	
FREDERICK JOHN TYDEMAN	78758	
FRANK W. TYRON JR.	06457	
JOHN URBANSKI	92704	
JAMES P. URONE	92630	
JOHN E. VAN DEUSEN III	83702	
ROBERT R. VAN TUYL	95132	
T. J. VAN WEERT	9321 GN	THE NETHERLANDS
DAVID VANCE	11790	
M. W. VANNIER	63132	
ANDREW VARANELLI	10038	
V. VINGE	92182	
LES VOGEL	95014	
HANS JONGE VOS	97223	
EMANUEL WACHSLER	01730	
ROBERTA WACHTER	15238	
EIITI WADA	113	JAPAN
KENNETH R. WADLAND	01420	
CLARK F. WAITE	92138	
A. R. M. WAJIH	HA9 0EE	UNITED KINGDOM
SCOTT WAKEFIELD	94305	
RICHARD WALCH	68106	
GEOFFREY F. WALKER	08822	
R. L. WALLACE	92675	
JAMES H. WALTERS	49003	
P. R. WALWYN	KT22 9NF	UNITED KINGDOM
JOHN B. WARDLAW	77024	
LES WARNER	48103	
DONALD WARREN	02174	
LOU WARSHAWSKY	60053	
PHILIP A. WASSON	90045	
JOE WATKINS	80302	
ANNA WATSON	32407	
CAM WATSON	91364	
DAN C. WATSON	45409	
JOHN J. WEDEL	91011	
PER-AKE WEDIN	S-901 87	SWEDEN
GARY L. WEIGEL	19380	
DAVID F. WEIL	98124	
STEPHEN J. WEINBERGER	98006	
DONALD G. WEISS	78721	
LARRY WEISS	75043	
JOHN H. WENSLEY	94022	
JOHN WEST	30327	
TERRY E. WEYMOUTH	01003	
WENDEL WHEELER	75221	
RICHARD WHIFFEN	19446	
MICHAEL A. WHITE III	63132	
WALTER A. WHITE	22205	
BILL WILDER	BOP 1X0	CANADA
LILLIAN WILHELMSON	05402	
BRIAN WILLIAMS	BN3 1RA	UNITED KINGDOM
E. HAROLD WILLIAMS	95050	
JAMES I. WILLIAMS	19342	
KENNETH L. WILLIAMS	01581	
KIM WILLIAMS	V3N 4N8	CANADA
M. HOWARD WILLIAMS	6140	SOUTH AFRICA
ARTHUR C. WILLIS	94086	
DAVID J. WILSON	48106	
DAVID T. WILSON	30305	
IAN ROBERT WILSON	M13 9PL	UNITED KINGDOM
GARY W. WINIGER	94088	
NIELS K. WINSOR	20375	
GREG WINTERHALTER	48130	
HANS-WILM WIPPERMANN	D-6750	GERMANY
IAN H. WITTEN	CO4 3SQ	UNITED KINGDOM
A. L. WOLBERT	60544	
WILLIAM WOLFSON	01778	
CHARLES WONG	55411	
JOHN WONG	97201	
HENRY WOOD	08540	
STEPHEN C. WOOD	87108	
WILLIAM T. WOOD	55343	
JAMES A. WOODS	94703	
JAY WOODS	98907	
RICHARD M. WOODWARD	95051	
JOHN D. WOOLLEY	98006	
ARDEN WOOTTON	85352	
ROGER P. WRIGHT	RG6 2LH	UNITED KINGDOM
NICHOLAS WYBOLT	01581	
M. J. L. YATES	GL52 5AJ	UNITED KINGDOM
NAKHSHON YESHURUN		ISRAEL
DAVID YOST	90046	
RAYMOND YOUNG	55165	
RONALD L. YOUNG	89119	
PETER YOUTZ	95051	
C. A. ZANONI	06455	
ALAN ZARING	78712	
HOWARD M. ZEIDLER	94025	
H. J. ZELL		
MARK ZIMMER		
ANDREW HARRIS ZIMMERMAN	95132	
DAVID J. ZOOK	60626	
DAN ZURAS	94061	

Applications

Please send all contributions for this section to Rich at the address below.

SOFTWARE TOOLS

by
Richard J. Cichelli
901 Whittier Dr.
Allentown, Pa. 18103
(215) 797-3153

ANPA/RI and
Lehigh University

THE "LONG AGO" PAST

In PN#6 of November 1976, I introduced the idea of a universal Pascal Software Tools set. Because the Software Tools section is now a part of PN, and PUG membership has increased by a factor of five since November 1976, it is relevant to re-state the ideas developed then.

SOFTWARE TOOLS FOR PASCAL

(*From "Pascal Potpouri" Pascal Newsletter #6 November 1976*)

Pascal implementations for new environments are occurring with ever increasing frequency. As Pascal is used for more and more production programming, it is important that a universal set of ancillary software tools be agreed upon. Some of these tools can be defined in an environment-independent way so that when written in standard Pascal, they can become part of a universal Pascal software development facility. I here propose an initial list. With PUG membership help, the list will develop into a working specification and a powerful set of programming tools.

PASCAL COMPILERS

Currently there exist Pascal compilers which produce absolute code, relocatable code, macro code (Pascal-J) and interpreted code (Pascal-P). Portable versions exist (Pascal-P and Pascal-J). Compiler trunks exist. A standard Pascal subset (Pascal-S) exists.

For compiler writers there should be a standard Pascal language test set. This universal set of Pascal programs would exercise new Pascal compilers and help implementors gain confidence in the correctness of their compilers.

An interactive interpreter should be developed. This system would provide interactive symbolic run time debugging facilities: breakpoints, interactive dumps, etc. It should be easy to do better than PL/I's Checkout compiler.

The Lecarme and Bochmann compiler writing systems are also important tools for any shop engaged in language development.

SOURCE PROGRAM TOOLS

Wirth has written a cross reference program. Perhaps, if the variable names were improved, a standard version of this program could be among the software tools. A formatter or "pretty printer" is essential for producing documentation quality listings. Mike Condict's might be a good starting place.

A code instrumenter is a very important debugging and refining tool. Instrumenters insert statement counters or timers so that reports of relative usage of code can be made. An instrumenter is invaluable in optimizing programs.

A high level macro preprocessor would also be a valuable facility.

SOURCE LIBRARIES

The CDC source library utility program UPDATE is currently used for distribution of the SCOPE versions of Pascal. It seems to me that a mini-version of UPDATE (with only sequential program libraries) could be implemented in Pascal. This would help standardize the distribution of Pascal Tools. (Incidentally, CDC's UPDATE is the best source library system I have ever seen. I think its quality should be emulated.)

For truly large systems (50,000+ lines) a source code data base is desirable. Such a system keeps track of which programs access what data and provides for standard file and record descriptions among programs, etc. I understand such a system for Pascal exists but is a deep, dark military secret.

DOCUMENTATION PREPARATION

W. Burger implemented part of Waite's PLAP in Pascal. We need a universal PLAP-like tool to maintain manuals and other documentation in machine readable form. Justification and hyphenation and facilities for producing high quality printing in upper and lower case should exist. Pascal documentation should be distributed in machine readable form for ease of publication and distribution.

OBJECT PROGRAM FACILITIES

Work is now in progress on programs which load Pascal absolute binaries. Facilities for overlay processing should be provided. Automated aids which help create effective overlay structures should be provided. A binary decoder is also a useful tool.

OTHER PROGRAMS

An efficient table processor with facilities like COBOL's Report Writer would be desirable. Current work on Pascal data base management systems, mathematical function libraries, and computer aided instruction systems augur the day of increased use of Pascal in business, engineering, and education. In the area of function libraries (for mathematics or business), facilities should be provided for not only linking in binary modules but also for including source modules.*

CONCLUSIONS

Obviously, where environmental conditions permit we should have a universal Pascal program implementing each software aid. Where the environmental factors prevent this, we should seek to provide a standard user interface to the desired functions.

*In my opinion, merging programs at the source level is to be preferred to binary level linking. Pascal compilers are typically faster than linking-loaders.

SOFTWARE TOOLS CONT'D

I believe that early article presented a viable perspective for future Pascal Software Tools (PST) work, but it left unanswered many important policy questions. The most critical of these was how to get PST to PUG members. One obvious answer was that implementors could distribute the tools with their distribution package. Of course, when the number of active implementors reached nearly 100, Andy and I were again unsure of what to do. About this time CACM stopped its Algorithms section. I almost cancelled my membership because of my feelings that the most important work of computer scientists is their programs. Without published programs, CACM is hollow for me.

I very much wanted to see quality programs in print. Andy was convinced that it was proper for PN to publish programs by the following arguments:

1) Publishing quality Pascal programs would help educate new PUB members. I believe reading good programs is the easiest way to learn programming techniques and style.

2) Publishing programs would give proper recognition to program authors.

3) Review and improvement of PST's by PUG members would be facilitated.

4) Published PST's would encourage implementors to adhere to the Pascal Standard.

5) Commercial users could require compiler vendors to use the PST's to test the conformity and performance of their implementations.

Andy was convinced and I even impressed myself with these arguments. The obvious result is the Software Tools section started in PN#12.

The above discussion should help PUG members understand and shape PN Software Tools policy and philosophy.

Applications

THE "IMMEDIATE" PAST

Of course, publishing programs has its own problems. The preliminary statement of what we are trying to do was in PN#12:

> We decided to create a new section for printing Pascal source programs for various applications including Software Tools and Algorithms. Additionally, here, we will print news of significant applications programs writen in Pascal. Jim Miner suggested we index each program so that they may be easily referenced for corrections and criticisms.
>
> Arthur Sale is very enthusiastic about the Algorithms section. He suggested that we allow for:
>
> 1) The provision for certification of the program by unrelated persons, with clear identification of the system used; and
>
> 2) Critiques of the program for:
>
>   a) standards conformance,
>   b) style,
>   c) algorithm,
>   d) output convenience and general design.
>
> We'll number programs starting with P-1, Software Tools starting with S-1, and Algorithms starting with A-1.

Already our numbering scheme is giving us problems. As S-2 (Augment and Analyze) made clear, Software Tools may not be just single programs but entire systems of programs. So that we can refer to text lines within programs uniquely, we will use the notation:

```
ReferenceNumber = Classification "-" System.
Classification = "S" | "T" | "A" | "V".
System = UnsignedInteger | UnsignedInteger "-" Program Designator.
ProgramDesignator = UnsignedInteger | ValidationSuiteDesignator.
```

Thus, Augment and Analyze are still S-2 but Augment, itself, becomes S-2-1 and Analyze S-2-2.

The validation suite designators (i.e. "V" programs) are being assigned by Brian Wichmann. So far there have been many favorable comments about PN#12's programs. Unfortunately, no certifications of the programs have been received to date. It is important that those members who bring up the programs comment on any problems they might have had. We really need to know the performance and ease of installation of the programs on various systems.

I'm sure the membership would be very interested in how well Jim Miner's Compare Program (S-1) performs against a Pascal implementatation ofPaul Heckel's file Comparison Algorithm. (CACM, April 1978, Vol. 21, Num. 4, "A Technique for Isolating Differences Between Files"). Sounds like a good term project to me.

THE PRESENT

This issue's pretty printers should help future PST submitters to produce camera-ready copy. We are experimenting with several publication styles. We want to be able to publish readable copy of large programs.

We may experiment with putting multiple simple statements per line and a vertical two-column format. It seems desirable to be able to publish systems of as many as 10,000 lines of code in a single issue. Thus, full compilers and large applications libraries could be accommodated. Anyone seeking to have a large program published should work closely with me and Andy. Incidentally, programs which are primarily tutorial in nature (i.e. not of general utility) should be incorporated in articles for publication in the Articles section of PN.

THE IMMEDIATE FUTURE

I'm sure it is of no surprise to any reader that almost all of the software tools described in PN#6 are now in existence. Unfortunately, many of these utilities need careful polishing before they are suitable for publication. We are working on getting those to which we have access into shape. Another problem is obtaining publication permission from authors and organizations. Work is going on in all these areas.

We believe the Software Tool set will grow in two ways: new utilities will be added and existing utilities will be modified or replaced by improved versions. We encourage the membership to help us carefully evaluate published programs. We hope all those who have developed Pascal Software Tools will try to submit them for publication.

WHAT'S NEW?

Condict's pretty printer allows us to publish program text in a consistent style. Equally important for program sharing are a source code library facility and a text formatter for documentation. After these high priority items have been published, PN will dazzle the PUG membership with extraordinary software.

In addition to the software mentioned in the PN#6 article, we hope to publish an APL interpreter written entirely in Pascal at Villanova. A program from the University of Montreal draws Nassi-Shneiderman diagrams for Pascal programs (see SIGPLAN notice, August 1973). We also have a program from North American Phillips Corporation which reads Pascal programs and marks them for standard conformance. The program handles most, but not all, standard Pascal programs. Anyone interested in polishing this program into recognizing the full standard should contact me immediately.

WHAT ARE WE LOOKING FOR?

An interactive editor in Pascal would be most welcome. A typesetting package would help newsletter production immensely. We've written a Motorola 6800 Assembler in Pascal at ANPA/RI. It and ones for the 8080, Z80, etc. would make good Software Tools entries.

I'd like to see a bootstrapable version of Pascal-S published in PN. Most programmers would be surprised at just how easy it is to compile Pascal. Reading a nice Pascal compiler written in Pascal would make every programmer reticent to muck with the standard. Incidentally, the Pascal-S system is an ideal tool for compiler writing courses.

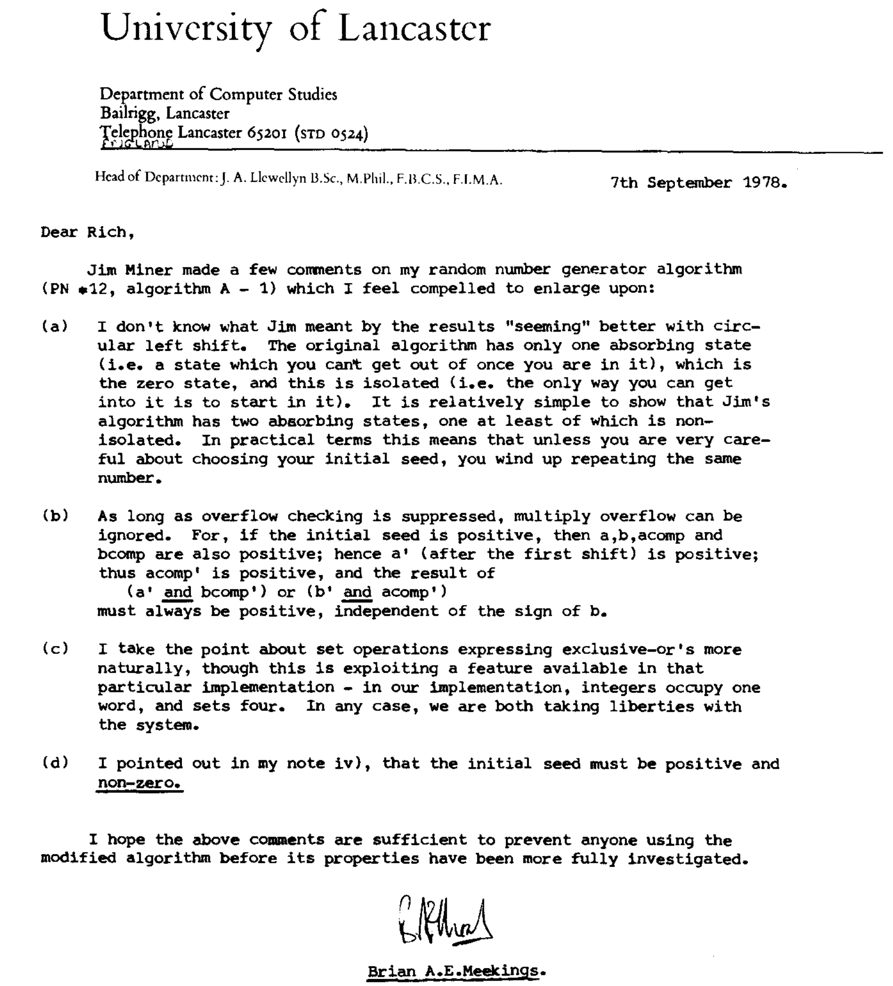
Lars Mossberg of Volvo Flygmotor in Sweden pointed out the importance of converting existing software systems to Pascal. We need FORTRAN, ALGOL, COBOL, and PL/1 to Pascal translators. Someone might also knock off a BASIC system in Pascal.

SOME FINAL WORDS

Those implementors and organizations, which insist on producing their own dialects of Pascal and foisting them on an unsuspecting public as being Pascal, are enemies of us all.

ALGORITHMS

A - 1 Random Number Generator (continued discussion)



University of Lancaster

Department of Computer Studies
Bailrigg, Lancaster
Telephone Lancaster 65201 (STD 0524)
ENGLAND

Head of Department: J. A. Llewellyn B.Sc., M.Phil., F.B.C.S., F.I.M.A.

7th September 1978.

Dear Rich,

Jim Miner made a few comments on my random number generator algorithm (PN #12, algorithm A - 1) which I feel compelled to enlarge upon:

(a) I don't know what Jim meant by the results "seeming" better with circular left shift. The original algorithm has only one absorbing state (i.e. a state which you can't get out of once you are in it), which is the zero state, and this is isolated (i.e. the only way you can get into it is to start in it). It is relatively simple to show that Jim's algorithm has two absorbing states, one at least of which is non-isolated. In practical terms this means that unless you are very careful about choosing your initial seed, you wind up repeating the same number.

(b) As long as overflow checking is suppressed, multiply overflow can be ignored. For, if the initial seed is positive, then a,b,acomp and bcomp are also positive; hence a' (after the first shift) is positive; thus acomp' is positive, and the result of
    (a' and bcomp') or (b' and acomp')
must always be positive, independent of the sign of b.

(c) I take the point about set operations expressing exclusive-or's more naturally, though this is exploiting a feature available in that particular implementation - in our implementation, integers occupy one word, and sets four. In any case, we are both taking liberties with the system.

(d) I pointed out in my note iv), that the initial seed must be positive and non-zero.

I hope the above comments are sufficient to prevent anyone using the modified algorithm before its properties have been more fully investigated.

Brian A.E.Meekings.

A - 3 Determine Real Number Environment

DOCUMENTATION : ENQUIRY

Language : Pascal

Written : A.H.J. Sale
Monday, 1978 March 20

Use

To allow programs to enquire into their environment (compiler + computer) and tailor their behaviour to the properties of the *real* arithmetic system. The procedure may be of use in programs that must be portable across many different PASCAL systems, and which are numerically oriented.

User documentation

Calling the *enquiry* procedure with the proper actual parameters determines the base and number of digits of the mantissa of the representation, and an indication of whether the arithmetic is truncated or not. Though the procedure works on a large range of computers, its correct operation depends on a number of assumptions about the representation of *real* numbers, and the operation of floating-point arithmetic. Programmers incorporating the procedure into programs are advised to cause the deductions to be printed so that end-users can check the accuracy of the deductions for their particular systems.

Installation

The *enquiry* procedure is standard PASCAL, in reference language form, and should compile on all systems. If assumption (b)(iv) is violated (as for example on the IBM 1130 which has more mantissa digits in its software accumulator than in the memory representation), rewriting the parenthesized expressions (and therefore the control structures) so that each parenthesized sub-expression is assigned to a memory cell will probably give the correct deductions for the memory representation. The same trick may be employed in defense against over-clever optimizing compilers that utilize properties of (mathematician's) real arithmetic, and re-organize expressions.

The displayed driver program illustrates how the best- and worst-case precision may be computed from the deductions about the arithmetic.

System documentation

The algorithm is an adaptation of one originally due to M.A. Malcolm (*Comm ACM, Vol. 15 No. 11 pp 949-951, November, 1972*).

Assumptions

It is assumed that:

(a) *Real* numbers are represented by floating-point representations which comply with the following conditions:

(i) There is a mantissa of a fixed number of digits to a fixed base.

(ii) There is an exponent which expresses a multiplying factor to be applied to the mantissa to obtain the exact representation value. The exponent only takes on integral values, and the multiplying factor is the base to the power given by the exponent.

(iii) The representation preserves maximum precision (no digits are lost unless the representation cannot accommodate them). In particular integral values with possible exact representations are exactly represented.

(b) *Real* arithmetic complies with the following rules:

(i) If operands and results are exactly represented integral values, no inaccuracy is introduced by the arithmetic.

(ii) The arithmetic is organized along the usual align, operate and normalize steps, where these are necessary.

(iii) It is presumed that when digits are lost due to the representation, they are either truncated (ignored), or true rounding takes place. (No other possibilities are taken into consideration.)

(iv) The intermediate results of arithmetic operations are held in a cell which has the same representational properties as the operands.

```
          B6700 PASCAL COMPILER   VERSION 2.8.002      FRIDAY, 1978 MARCH 17, 11:43 AM.
          ==========================================================================

$SET $ LIST LINEINFO STANDARD                                              00001000  003:0000:1
program investigaterepresentation;                                         00002000  003:0000:1
        {*************************}                                        00002500  003:0000:1
var                                                                        00003000  003:0000:1
    base,                                                                  00004000  003:0000:1
    numberofdigits,                                                        00005000  003:0000:1
    i                   : integer;                                         00006000  003:0000:1
    rounding            : boolean;                                         00007000  003:0000:1
    epsilon             : real;                                            00008000  003:0000:1
                                                                           00009000  003:0000:1
procedure enquiry(var radix,digits : integer; var rounds : boolean);      00010000  003:0000:1
          {*******}                                                        00011000  003:0000:1
var                                                                        00012000  003:0000:1
    number,                                                                00013000  004:0000:1
    increment           : real;                                            00014000  004:0000:1
begin                                                                      00015000  004:0000:1
    { find large integral value just beyond integer limits }               00016000  004:0000:1
    number:=2;                                                             00017000  004:0000:1
    while (((number+1)-number) = 1) do number:=number*2;                   00018000  004:0001:0
    { manufacture the next largest real value }                            00019000  004:0004:5
    increment:=2;                                                          00020000  004:0004:5
    while ((number+increment) = number) do increment:=2*increment;         00021000  004:0005:4
    { subtract these to give radix of representation }                     00022000  004:0009:2
    radix:=trunc((number+increment)-number);                               00023000  004:0009:2
    { see if it rounds or truncates by adding (radix-1) }                  00024000  004:000B:3
    rounds:=((number+(radix-1)) <> number);                                00025000  004:000B:3
    { work out how many digits in mantissa }                               00026000  004:000D:4
    digits:=0;                                                             00027000  004:000D:4
    number:=1;                                                             00028000  004:000E:3
    while (((number+1)-number) = 1) do begin                               00029000  004:000F:1
        digits:=digits+1;                                                  00030000  004:0011:1
        number:=number*radix                                               00031000  004:0012:3
    end                                                                    00032000  004:0012:5
end; { of enquiry procedure }                                              00033000  004:0013:5
                                                                           00034000  004:0016:1
begin { of main program body }                                             00035000  004:0016:1
    { find out basic properties }                                          00036000  003:0000:1
    enquiry(base,numberofdigits,rounding);                                 00037000  003:0000:1
    writeln(output,' BASE=',base:5);                                       00038000  003:0002:2
    writeln(output,' NUMBER OF DIGITS=',numberofdigits:5);                 00039000  003:000A:2
    if rounding then                                                       00040000  003:0012:2
        writeln(output,' ROUNDED')                                         00041000  003:0013:1
    else                                                                   00042000  003:0019:3
        writeln(output,' TRUNCATED');                                      00043000  003:001A:0
    { compute the precision bounds }                                       00044000  003:0020:2
    epsilon:=1;                                                            00045000  003:0020:2
    for i:=1 to numberofdigits do epsilon:=epsilon/base;                   00046000  003:0021:0
    if rounding then epsilon:=epsilon/2;                                   00047000  003:0027:5
    { print the best and worst precision }                                 00048000  003:002A:0
    writeln(output,' BEST AND WORST PRECISIONS ARE ',                      00049000  003:002A:0
        epsilon,(epsilon*base))                                            00050000  003:002F:2
end.                                                                       00051000  003:0034:3
```

Certification for Burroughs B6700

The following output is produced when running the test program on a Burroughs B6748 processor with the University of Tasmania compiler and is correct:

```
BASE=    8
NUMBER OF DIGITS=   13
ROUNDED
BEST AND WORST PRECISIONS ARE   0.9094947E-12  0.7275958E-11
```

SOFTWARE TOOLS

One important aspect about Pascal coding style is consistency, although styles certainly differ from one programmer to the next. The two software tools in this issue are both Pascal Prettyprinters, which aid Pascal programmers in their coding activities. They represent 2 vastly contrasting philosophies, and so I think it is appropriate that we print both, and are assured that we have two of the best in existence. S-3, Prettyprint adheres to the philosophy that there are serious issues in prettyprinting, and that it is only necessary to impose a minimum set of restrictions in prettyprinting--not be heavyhanded, not do full syntax analysis, and not provide a voluminous set of options. Prettyprint does prettyprinting on a local basis and thus can handle Pascal program fragments, and even incorrect programs. The important principle is that all blank lines and blanks supplied in the original source are preserved.

S-4, Format indeed does provide a large set of options because no prettyprinting style can please everyone, and by allowing complete control over the process, one can achieve pleasing results. Indeed at our site where both of these prettyprinters are available, Format is the choice over Prettyprint by 3 to 1. I use both myself.

Prettyprint was first announced in Pascal Newsletter #6 page 70, in November, 1976. Henry Ledgard reports that they lost a lot of money distributing it. Charles Fischer was kind enough to provide some small corrections (indicated in the program) before we published it.

Format has been around for the last 3 years, and remains in my opinion, one of the all-time, best-looking Pascal programs in existence because of its use of long and meaningful identifiers. It looks all the sharper in upper-and-lower case!

There has been quite a bit of noise in the literature about Pascal prettyprinting. We cited Singer, et al.'s article "A Basis for Executing Pascal Programmers" in PUGN 9/10 page 9; Peterson's article "On the Formatting of Pascal Programs" in PUGN 11 page 10; Sale's article "Stylistics in Languages with Compound Statements" in PUGN 12 page 10, and in this issue; Mohilner's article "Prettyprinting Pascal Programs" in this issue; and I now find Crider's article "Structured Formatting of Pascal Programs" in the November, 1978 SIGPLAN Notices.

Unfortunately, both prettyprinters could do better in their treatment of comments. They are living examples of their results, because they have been run through themselves! And as such I am very pleased that we can present them here together with their superb documentation. (*Please excuse my role therein.*) If you want to use these prettyprinters, key them in, or request that your Pascal compiler distributor include them on the distribution tape for your favorite Pascal system. CDC-6000 Pascal Release 3 will include both Prettyprint and Format. Happy prettyprinting '79!

- Andy Mickel

S - 3 Prettyprint

```
{=================================================================}
{                                                                 }
{  Program Title: Pascal Prettyprinting Program                   }
{                                                                 }
{  Authors: Jon F. Hueras and Henry F. Ledgard                    }
{           Computer and Information Science Department           }
{           University of Massachusetts, Amherst - August, 1976   }
{           (Earlier versions and contributions by                }
{           Randy Chow and John Gorman).                          }
{                                                                 }
{           Bugs corrected by Charles Fischer, Department of      }
{           Computer Science, University of Wisconsin, Madison.   }
{           1977.  Indicated by <<<.                              }
{                                                                 }
{           Modified for CDC-6000 Pascal Release 3 by Rick L. Marcus }
{           University Computer Center, University of Minnesota.  }
{           30 September 1978.                                    }
{                                                                 }
{  Program Summary:                                               }
{                                                                 }
{     This program takes as input a Pascal program and            }
{     reformats the program according to a standard set of       }
{     prettyprinting rules. The prettyprinted program is given    }
{     as output.  The prettyprinting rules are given below.       }
{                                                                 }
{     An important feature is the provision for the use of extra  }
{     spaces and extra blank lines.  They may be freely inserted by }
{     the user in addition to the spaces and blank lines inserted }
{     by the prettyprinter.                                       }
{                                                                 }
{     No attempt is made to detect or correct syntactic errors in }
{     the user's program.  However, syntactic errors may result in }
{     erroneous prettyprinting.                                   }
{                                                                 }
{                                                                 }
{  Input File: input    - a file of characters, presumably a      }
{                         Pascal program or program fragment.     }
{                                                                 }
{  Output File: output - the prettyprinted program.               }
{                                                                 }
{                                                                 }
{                                                                 }
{=================================================================}


{=================================================================}
{                                                                 }
{                   Pascal Prettyprinting Rules                   }
{                                                                 }
{                                                                 }
{  [ General Prettyprinting Rules ]                               }
{                                                                 }
{   1.   Any spaces or blank lines beyond those generated by the  }
{     prettyprinter are left alone.  The user is encouraged, for the }
{     sake of readability, to make use of this facility.          }
{        In addition, comments are left where they are found, unless }
{     they are shifted right by preceeding text on a line.        }
{                                                                 }
{   2.   All statements and declarations begin on separate lines. }
{                                                                 }
{   3.   No line may be greater than 72 characters long.  Any line }
{     longer than this is continued on a separate line.           }
{                                                                 }
{   4.   The keywords "BEGIN", "END", "REPEAT", and "RECORD" are  }
{     forced to stand on lines by themselves (or possibly follwed by }
{     supporting comments).                                       }
{        In  addition, the "UNTIL" clause of a "REPEAT-UNTIL" state- }
{     ment is forced to start on a new line.                      }
{                                                                 }
{   5.   A blank line is forced before the keywords "PROGRAM",    }
{     "PROCEDURE", "FUNCTION", "LABEL", "CONST", "TYPE", and "VAR". }
{                                                                 }
{   6.   A space is forced before and after the symbols ":=" and  }
{     "=".  Additionally, a space is forced after the symbol ":". }
{     Note that only "="s in declarations are formatted.  "="s in }
{     expressions are ignored.  <<<.                              }
{                                                                 }
{                                                                 }
{  [ Indentation Rules ]                                          }
{                                                                 }
{   1.   The bodies of "LABEL", "CONST", "TYPE", and "VAR" declara- }
{     tions are indented from their corresponding declaration header }
{     keywords.                                                   }
{                                                                 }
{   2.  The bodies of "BEGIN-END", "REPEAT-UNTIL", "FOR", "WHILE",   }
{     "WITH", and "CASE" statements, as well as "RECORD-END" struc-  }
{     tures and "CASE" variants (to one level) are indented from     }
{     their header keywords.                                         }
{                                                                    }
{   3.  An "IF-THEN-ELSE" statement is indented as follows:          }
{                                                                    }
{               IF <expression>                                      }
{                  THEN                                              }
{                     <statement>                                    }
{                  ELSE                                              }
{                     <statement>                                    }
{                                                                    }
{                                                                    }
{====================================================================}


{====================================================================}
{                                                                    }
{                       General Algorithm                            }
{                                                                    }
{                                                                    }
{      The strategy of the prettyprinter is to scan symbols from     }
{   the input program and map each symbol into a prettyprinting      }
{   action, independently of the context in which the symbol         }
{   appears.  This is accomplished by a table of prettyprinting      }
{   options.                                                         }
{                                                                    }
{      For each distinguished symbol in the table, there is an       }
{   associated set of options.  If the option has been selected for  }
{   the symbol being scanned, then the action corresponding with     }
{   each option is performed.                                        }
{                                                                    }
{      The basic actions involved in prettyprinting are the indent-  }
{   ation and de-indentation of the margin.  Each time the margin is }
{   indented, the previous value of the margin is pushed onto a      }
{   stack, along with the name of the symbol that caused it to be    }
{   indented.  Each time the margin is de-indented, the stack is     }
{   popped off to obtain the previous value of the margin.           }
{                                                                    }
{      The prettyprinting options are processed in the following     }
{   order, and invoke the following actions:                         }
{                                                                    }
{                                                                    }
{     crsuppress      - If a carriage return has been inserted       }
{                       following the previous symbol, then it is    }
{                       inhibited until the next symbol is printed.  }
{                                                                    }
{     crbefore        - A carriage return is inserted before the     }
{                       current symbol (unless one is already there) }
{                                                                    }
{     blanklinebefore - A blank line is inserted before the current  }
{                       symbol (unless already there).               }
{                                                                    }
{     dindentonkeys   - If any of the specIfied keys are on top of   }
{                       of the stack, the stack is popped, de-inden- }
{                       ting the margin.  The process is repeated    }
{                       until the top of the stack is not one of the }
{                       specified keys.                              }
{                                                                    }
{     dindent         - The stack is unconditionally popped and the  }
{                       margin is de-indented.                       }
{                                                                    }
{     spacebefore     - A space is inserted before the symbol being  }
{                       scanned (unless already there).              }
{                                                                    }
{     [ the symbol is printed at this point ]                        }
{                                                                    }
{     spaceafter      - A space is inserted after the symbol being   }
{                       scanned (unless already there).              }
{                                                                    }
{     gobbleSymbols   - Symbols are continuously scanned and printed }
{                       without any processing until one of the      }
{                       specified symbols is seen (but not gobbled). }
{                                                                    }
{     indentbytab     - The margin is indented by a standard amount  }
{                       from the previous margin.                    }
{                                                                    }
{     indenttoclp     - The margin is indented to the current line   }
{                       position.                                    }
{                                                                    }
{     crafter         - A carriage return is inserted following the  }
{                       symbol scanned.                              }
{                                                                    }
{                                                                    }
{                                                                    }
{====================================================================}


program prettyprint( { from }  input,
                     { to }    output );


const

      maxsymbolsize = 200; { the maximum size (in characters) of a    }
                           { symbol scanned by the lexical scanner.   }

      maxstacksize  = 100; { the maximum number of symbols causing    }
                           { indentation that may be stacked.         }

      maxkeylength  =  10; { the maximum length (in characters) of a  }
                           { pascal reserved keyword.                 }
      maxlinesize   =  72; { the maximum size (in characters) of a    }
                           { line output by the prettyprinter.        }

      slofail1      =  30; { up to this column position, each time    }
                           { "indentbytab" is invoked, the margin     }
                           { will be indented by "indent1".           }

      slofail2      =  48; { up to this column position, each time    }
                           { "indentbytab" is invoked, the margin     }
                           { will be indented by "indent2".  beyond   }
                           { this, no indentation occurs.             }

      indent1       =   3;

      indent2       =   1;


      space = ' ';


type

     keysymbol = ( progsym,     funcsym,      procsym,
                   labelsym,    constsym,     typesym,     varsym,
                   beginsym,    repeatsym,    recordsym,
                   casesym,     casevarsym,   ofsym,
                   forsym,      whilesym,     withsym,     dosym,
                   ifsym,       thensym,      elsesym,
                   endsym,      untilsym,
                  becomes,    opencomment, closecomment,
                  semicolon,  colon,       equals,
                  openparen,  closeparen,  period,
                  endoffile,
                  othersym );

    option = ( crsuppress,
               crbefore,
               blanklinebefore,
               dindentonkeys,
               dindent,
               spacebefore,
               spaceafter,
               gobblesymbols,
               indentbytab,
               indenttoclp,
               crafter );

    optionset = set of option;

    keysymset = set of keysymbol;

    tableentry = record
                    optionsselected  : optionset;
                    dindentsymbols   : keysymset;
                    gobbleterminators: keysymset
                 end;

    optiontable = array [ keysymbol ] of tableentry;


    key = packed array [ 1..maxkeylength ] of char;


    keywordtable = array [ progsym..untilsym ] of key;


    specialchar = packed array [ 1..2 ] of char;

    dblchrset = set of becomes..opencomment;

    dblchartable = array [ becomes..opencomment ] of specialchar;

    sglchartable = array [ semicolon..period ] of char;


    string = array [ 1..maxsymbolsize ] of char;

    symbol = record
                name        : keysymbol;
                valu        : string;
                length      : integer;
                spacesbefore: integer;
                crsbefore   : integer
             end;

    symbolinfo = ^symbol;


    charname = ( letter,    digit,    blank,    quote,
                 endofline, filemark, otherchar       );

    charinfo = record
                  name : charname;
                  valu : char
               end;
```

```


    stackentry = record
                    indentsymbol: keysymbol;
                    prevmargin  : integer
                 end;

    symbolstack = array [ 1..maxstacksize ] of stackentry;


var
    recordseen: boolean;

    currchar,
    nextchar: charinfo;

    currsym,
    nextsym: symbolinfo;

    crpending: boolean;

    ppoption: optiontable;

    keyword: keywordtable;

    dblchars: dblchrset;

    dblchar: dblchartable;
    sglchar: sglchartable;

    stack: symbolstack;
    top  : integer;

    startpos,   { starting position of last symbol written  } <<<
    currlinepos,
    currmargin :  integer;

procedure getchar( { from input }

                   { updating }  var nextchar  : charinfo;
                   { returning } var currchar  : charinfo );

begin { getchar }

   currchar := nextchar;

   with nextchar do
      begin

         if eof(input)
            then
               name  := filemark

    else if eoln(input)
            then
               name  := endofline

    else if input^ in ['a'..'z']
            then
               name  := letter

    else if input^ in ['0'..'9']
            then
               name  := digit

    else if input^ = ''''
```

```
            then
               name  := quote

    else if input^ = space
            then
               name  := blank

    else name := otherchar;


         if name in [ filemark, endofline ]
            then
               valu := space
            else
               valu := input^;

         if name <> filemark
            then
               get(input)

      end { with }

end; { getchar }


procedure storenextchar( { from input }
                         { updating }    var length    : integer;
                                           var currchar,
                                               nextchar  : charinfo;
                         { placing in }  var valu      : string    );

begin { storenextchar }

   getchar( { from input }
            { updating }  nextchar,
            { returning } currchar  );

   if length < maxsymbolsize
      then
         begin

            length := length + 1;

            valu [length] := currchar.valu

         end

end; { storenextchar }


procedure skipspaces(
                      { updating }  var currchar,
                                          nextchar     : charinfo;
                      { returning } var spacesbefore,
                                          crsbefore    : integer  );

begin { skipspaces }

   spacesbefore := 0;
   crsbefore    := 0;

   while nextchar.name in [ blank, endofline ] do
      begin

         getchar( { from input }
                  { updating }  nextchar,
                  { returning } currchar  );

         case currchar.name of

            blank     : spacesbefore := spacesbefore + 1;

            endofline : begin
                           crsbefore    := crsbefore + 1;
                           spacesbefore := 0
                        end

         end { case }

      end { while }

end; { skipspaces }


procedure getcomment( { from input }
                      { updating } var currchar,
                                         nextchar   : charinfo;
                                     var name       : keysymbol;
                                     var valu       : string;
                                     var length     : integer   );

begin { getcomment }

   name := opencomment;

   while not(    ((currchar.valu = '*') and (nextchar.valu = ')'))
              or (nextchar.name = endofline)
              or (nextchar.name = filemark)) do

      storenextchar( { from input }
                     { updating } length,
                                  currchar,
                                  nextchar,
                     { in }       valu      );


   if (currchar.valu = '*') and (nextchar.valu = ')')
      then
         begin

            storenextchar( { from input }
                           { updating } length,
                                        currchar,
                                        nextchar,
                           { in }       valu      );

            name := closecomment

         end

end; { getcomment }


function idtype( { of }        valu   : string;
                 { using }     length : integer )
                 { returning }                      : keysymbol;

var
    i: integer;

    keyvalu: key;

    hit: boolean;

    thiskey: keysymbol;


begin { idtype }

   idtype := othersym;

   if length <= maxkeylength
      then
         begin

            for i := 1 to length do
               keyvalu [i] := valu [i];

            for i := length+1 to maxkeylength do
               keyvalu [i] := space;

            thiskey := progsym;
            hit     := false;

            while not(hit or (thiskey = succ(untilsym))) do           <<<
               if keyvalu = keyword [thiskey]
                  then
                     hit := true
                  else
                     thiskey := succ(thiskey);

            if hit
               then
                  idtype := thiskey

         end;

end; { idtype }


procedure getidentifier( { from input }
                         { updating }  var currchar,
                                             nextchar  : charinfo;
                         { returning } var name      : keysymbol;
                                         var valu      : string;
                                         var length    : integer   );

begin { getidentifier }

   while nextchar.name in [ letter, digit ] do

      storenextchar( { from input }
                     { updating } length,
                                    currchar,
                                    nextchar,
                     { in }       valu      );


   name := idtype( { of }    valu,
                   { using } length );

   if name in [ recordsym, casesym, endsym ]
      then
         case name of

            recordsym : recordseen := true;

            casesym   : if recordseen
                           then
                              name := casevarsym;

            endsym    : recordseen := false

         end { case }

end; { getidentifier }


procedure getnumber( { from input }
                     { updating }  var currchar,
                                         nextchar  : charinfo;
                     { returning } var name      : keysymbol;
                                     var valu      : string;
                                     var length    : integer   );

begin { getnumber }

   while nextchar.name = digit do

      storenextchar( { from input }
                     { updating } length,
                                    currchar,
                                    nextchar,
                     { in }       valu      );


   name := othersym

end; { getnumber }


procedure getcharliteral( { from input }
                          { updating }  var currchar,
                                              nextchar  : charinfo;
                          { returning } var name      : keysymbol;
                                          var valu      : string;
                                          var length    : integer   );

begin { getcharliteral }

   while nextchar.name = quote do
      begin

         storenextchar( { from input }
                        { updating } length,
                                       currchar,
                                       nextchar,
                        { in }       valu      );

         while not(nextchar.name in [ quote, endofline, filemark ]) do

            storenextchar( { from input }
                           { updating } length,
                                          currchar,
                                          nextchar,
                           { in }       valu      );


         if nextchar.name = quote
            then
               storenextchar( { from input }
                              { updating } length,
                                             currchar,
                                             nextchar,
                              { in }       valu      )

      end;


   name := othersym

end; { getcharliteral }


function chartype( { of }         currchar,
                                    nextchar : charinfo )
                   { returning }                      : keysymbol;

var
    nexttwochars: specialchar;

    hit: boolean;

    thischar: keysymbol;


begin { chartype }

   nexttwochars[1] := currchar.valu;
   nexttwochars[2] := nextchar.valu;

   thischar := becomes;
   hit      := false;

   while not(hit or (thischar = closecomment)) do
      if nexttwochars = dblchar [thischar]
         then
            hit := true
         else
            thischar := succ(thischar);

   if not hit
      then
         begin

            thischar := semicolon;

            while not(hit or (pred(thischar) = period)) do
               if currchar.valu = sglchar [thischar]
                  then
                     hit := true
                  else
                     thischar := succ(thischar)

         end;

   if hit
      then
         chartype := thischar
      else
         chartype := othersym

end; { chartype }


procedure getspecialchar( { from input }
                          { updating }  var currchar,
                                              nextchar  : charinfo;
                          { returning } var name      : keysymbol;
                                          var valu      : string;
                                          var length    : integer   );

begin { getspecialchar }

   storenextchar( { from input }
                  { updating } length,
                                 currchar,
                                 nextchar,
                  { in }       valu      );

   name := chartype( { of } currchar,
                              nextchar );

   if name in dblchars
      then

         storenextchar( { from input }
                        { updating } length,
                                       currchar,
                                       nextchar,
                        { in }       valu      )

end; { getspecialchar }


procedure getnextsymbol( { from input }
                         { updating }  var currchar,
                                             nextchar  : charinfo;
                         { returning } var name      : keysymbol;
                                         var valu      : string;
                                         var length`   : integer   );

begin { getnextsymbol }

   case nextchar.name of

      letter      : getidentifier( { from input }
                                   { updating }  currchar,
                                                   nextchar,
                                   { returning } name,
                                                   valu,
                                                   length    );

      digit       : getnumber( { from input }
                               { updating }  currchar,
                                               nextchar,
                               { returning } name,
                                               valu,
                                               length    );

      quote       : getcharliteral( { from input }
                                    { updating }  currchar,
                                                    nextchar,
                                    { returning } name,
                                                    valu,
                                                    length    );

      otherchar   : begin

                       getspecialchar( { from input }
                                       { updating }  currchar,
                                                       nextchar,
                                       { returning } name,
                                                       valu,
                                                       length    );

                        if name = opencomment
                           then
                              getcomment( { from input }
                                          { updating } currchar,
                                                       nextchar,
                                                       name,
                                                       valu,
                                                       length    )

                     end;

        filemark    : name := endoffile

     end { case }

end; { getnextsymbol }


procedure getsymbol( { from input }
                     { updating }  var nextsym   : symbolinfo;
                     { returning } var currsym   : symbolinfo );

var
    dummy: symbolinfo;


begin { getsymbol }

   dummy   := currsym;
   currsym := nextsym;
   nextsym := dummy   ;

   with nextsym^ do
      begin

         skipspaces(
                     { updating }  currchar,
                                   nextchar,
                     { returning } spacesbefore,
                                   crsbefore     );
         length := 0;

         if currsym^.name = opencomment
            then
               getcomment( { from input }
                           { updating }  currchar,
                                         nextchar,
                           { returning } name,
                                         valu,
                                         length    )
            else
               getnextsymbol( { from input }
                              { updating }  currchar,
                                            nextchar,
                              { returning } name,
                                            valu,
                                            length    )

      end { with }

end; { getsymbol }


procedure initialize( { returning }


                          var topofstack   : integer;

                          var currlinepos,
                              currmargin   : integer;

                          var keyword      : keywordtable;

                          var dblchars     : dblchrset;

                          var dblchar      : dblchartable;

                          var sglchar      : sglchartable;

                          var recordseen   : boolean;

                          var currchar,
                              nextchar     : charinfo;

                          var currsym,
                              nextsym      : symbolinfo;

                          var ppoption     : optiontable    );


begin { initialize }

   linelimit(output,maxint);

   topofstack  := 0;
   currlinepos := 0;
   currmargin  := 0;


   keyword [ progsym    ] := 'program   ' ;
   keyword [ funcsym    ] := 'function  ' ;
   keyword [ procsym    ] := 'procedure ' ;
   keyword [ labelsym   ] := 'label     ' ;
   keyword [ constsym   ] := 'const     ' ;
   keyword [ typesym    ] := 'type      ' ;
   keyword [ varsym     ] := 'var       ' ;
   keyword [ beginsym   ] := 'begin     ' ;
   keyword [ repeatsym  ] := 'repeat    ' ;
   keyword [ recordsym  ] := 'record    ' ;
   keyword [ casesym    ] := 'case      ' ;
   keyword [ casevarsym ] := 'case      ' ;
   keyword [ ofsym      ] := 'of        ' ;
   keyword [ forsym     ] := 'for       ' ;
   keyword [ whilesym   ] := 'while     ' ;
   keyword [ withsym    ] := 'with      ' ;
   keyword [ dosym      ] := 'do        ' ;
   keyword [ ifsym      ] := 'if        ' ;
   keyword [ thensym    ] := 'then      ' ;
   keyword [ elsesym    ] := 'else      ' ;
   keyword [ endsym     ] := 'end       ' ;
   keyword [ untilsym   ] := 'until     ' ;


   dblchars := [ becomes, opencomment ];

   dblchar [ becomes     ]  := ':=' ;
   dblchar [ opencomment ]  := '(*' ;

   sglchar [ semicolon  ]   := ';' ;
   sglchar [ colon      ]   := ':' ;
   sglchar [ equals     ]   := '=' ;
   sglchar [ openparen  ]   := '(' ;
   sglchar [ closeparen ]   := ')' ;
   sglchar [ period     ]   := '.' ;

   recordseen := false;


   getchar( { from input }
            { updating }  nextchar,
            { returning } currchar  );

   new(currsym);
   new(nextsym);

   getsymbol( { from input }
              { updating }  nextsym,
              { returning } currsym  );


   with ppoption [ progsym ] do
      begin
         optionsselected    := [ blanklinebefore,
                                 spaceafter ];
         dindentsymbols     := [];
         gobbleterminators  := []
      end;

   with ppoption [ funcsym ] do
      begin
         optionsselected    := [ blanklinebefore,
                                 dindentonkeys,
                                 spaceafter ];
         dindentsymbols     := [ labelsym,
                                 constsym,
                                 typesym,
                                 varsym ];
         gobbleterminators  := []
      end;

   with ppoption [ procsym ] do
      begin
         optionsselected    := [ blanklinebefore,
                                 dindentonkeys,
                                 spaceafter ];
         dindentsymbols     := [ labelsym,
                                 constsym,
                                 typesym,
                                 varsym ];
         gobbleterminators  := []
      end;

   with ppoption [ labelsym ] do
      begin
         optionsselected    := [ blanklinebefore,
                                 spaceafter,
                                 indenttoclp ];
         dindentsymbols     := [];
         gobbleterminators  := []
      end;

   with ppoption [ constsym ] do
      begin
         optionsselected    := [ blanklinebefore,
                                 dindentonkeys,
                                 spaceafter,
                                 indenttoclp ];
         dindentsymbols     := [ labelsym ];
        gobbleterminators := []
    end;

  with ppoption [ typesym ] do
    begin
        optionsselected   := [ blanklinebefore,
                               dindentonkeys,
                               spaceafter,
                               indenttoclp ];
        dindentsymbols    := [ labelsym,
                               constsym ];
        gobbleterminators := []
    end;

  with ppoption [ varsym ] do
    begin
        optionsselected   := [ blanklinebefore,
                               dindentonkeys,
                               spaceafter,
                               indenttoclp ];
        dindentsymbols    := [ labelsym,
                               constsym,
                               typesym ];
        gobbleterminators := []
    end;

  with ppoption [ beginsym ] do
    begin
        optionsselected   := [ dindentonkeys,
                               indentbytab,
                               crafter ];
        dindentsymbols    := [ labelsym,
                               constsym,
                               typesym,
                               varsym ];
        gobbleterminators := []
    end;

  with ppoption [ repeatsym ] do
    begin
        optionsselected   := [ indentbytab,
                               crafter ];
        dindentsymbols    := [];
        gobbleterminators := []
    end;

  with ppoption [ recordsym ] do
    begin
        optionsselected   := [ indentbytab,
                               crafter ];
        dindentsymbols    := [];
        gobbleterminators := []
    end;

  with ppoption [ casesym ] do
    begin
        optionsselected   := [ spaceafter,
                               indentbytab,
                               gobblesymbols,
                               crafter ];
        dindentsymbols    := [];
        gobbleterminators := [ ofsym ]
    end;

  with ppoption [ casevarsym ] do
    begin
        optionsselected   := [ spaceafter,
                               indentbytab,
                               gobblesymbols,
                               crafter ];
        dindentsymbols    := [];
        gobbleterminators := [ ofsym ]
    end;

  with ppoption [ ofsym ] do
    begin
        optionsselected   := [ crsuppress,
                               spacebefore ];
        dindentsymbols    := [];
        gobbleterminators := []
    end;

  with ppoption [ forsym ] do
    begin
        optionsselected   := [ spaceafter,
                               indentbytab,
                               gobblesymbols,
                               crafter ];
        dindentsymbols    := [];
        gobbleterminators := [ dosym ]
    end;

  with ppoption [ whilesym ] do
    begin
        optionsselected   := [ spaceafter,
                               indentbytab,
                               gobblesymbols,
                               crafter ];
        dindentsymbols    := [];
        gobbleterminators := [ dosym ]
    end;

  with ppoption [ withsym ] do
    begin
        optionsselected   := [ spaceafter,
                               indentbytab,
                               gobblesymbols,
                               crafter ];
        dindentsymbols    := [];
        gobbleterminators := [ dosym ]
    end;

  with ppoption [ dosym ] do
    begin
        optionsselected   := [ crsuppress,
                               spacebefore ];
        dindentsymbols    := [];
        gobbleterminators := []
    end;

  with ppoption [ ifsym ] do
    begin
        optionsselected   := [ spaceafter,
                               indentbytab,
                               gobblesymbols,
                               crafter ];
        dindentsymbols    := [];
        gobbleterminators := [ thensym ]
    end;

  with ppoption [ thensym ] do
    begin
        optionsselected   := [ indentbytab,
                               crafter ];
        dindentsymbols    := [];
        gobbleterminators := []
    end;

  with ppoption [ elsesym ] do
    begin
        optionsselected   := [ crbefore,
                               dindentonkeys,
                               dindent,
                               indentbytab,
                               crafter ];
        dindentsymbols    := [ ifsym,
                               elsesym ];
        gobbleterminators := []
    end;

  with ppoption [ endsym ] do
    begin
        optionsselected   := [ crbefore,
                               dindentonkeys,
                               dindent,
                               crafter ];
        dindentsymbols    := [ ifsym,
                               thensym,
                               elsesym,
                               forsym,
                               whilesym,
                               withsym,
                               casevarsym,
                               colon,
                               equals ];
        gobbleterminators := []
    end;

  with ppoption [ untilsym ] do
    begin
        optionsselected   := [ crbefore,
                               dindentonkeys,
                               dindent,
                               spaceafter,
                               gobblesymbols,
                               crafter ];
        dindentsymbols    := [ ifsym,
                               thensym,
                               elsesym,
                               forsym,
                               whilesym,
                               withsym,
                               colon,
                               equals ];
        gobbleterminators := [ endsym,
                               untilsym,
                               elsesym,
                               semicolon ];
    end;

  with ppoption [ becomes ] do
    begin
        optionsselected   := [ spacebefore,
                               spaceafter,
                               gobblesymbols ];
        dindentsymbols    := [];
        gobbleterminators := [ endsym,
                               untilsym,
                                elsesym,
                                semicolon ]
      end;

   with ppoption [ opencomment ] do
      begin
         optionsselected   := [ crsuppress ];
         dindentsymbols    := [];
         gobbleterminators := []
      end;

   with ppoption [ closecomment ] do
      begin
         optionsselected   := [ crsuppress ];
         dindentsymbols    := [];
         gobbleterminators := []
      end;

   with ppoption [ semicolon ] do
      begin
         optionsselected   := [ crsuppress,
                                dindentonkeys,
                                crafter ];
         dindentsymbols    := [ ifsym,
                                thensym,
                                elsesym,
                                forsym,
                                whilesym,
                                withsym,
                                colon,
                                equals ];
         gobbleterminators := []
      end;

   with ppoption [ colon ] do
      begin
         optionsselected   := [ spaceafter,
                                indenttoclp ];
         dindentsymbols    := [];
         gobbleterminators := []
      end;

   with ppoption [ equals ] do
      begin
         optionsselected   := [ spacebefore,
                                spaceafter,
                                indenttoclp ];
         dindentsymbols    := [];
         gobbleterminators := []
      end;

   with ppoption [ openparen ] do
      begin
         optionsselected   := [ gobblesymbols ];
         dindentsymbols    := [];
         gobbleterminators := [ closeparen ]
      end;

   with ppoption [ closeparen ] do
      begin
         optionsselected   := [];
         dindentsymbols    := [];
         gobbleterminators := []
      end;

   with ppoption [ period ] do
      begin
         optionsselected   := [ crsuppress ];
         dindentsymbols    := [];
         gobbleterminators := []
      end;

   with ppoption [ endoffile ] do
      begin
         optionsselected   := [];
         dindentsymbols    := [];
         gobbleterminators := []
      end;

   with ppoption [ othersym ] do
      begin
         optionsselected   := [];
         dindentsymbols    := [];
         gobbleterminators := []
      end


end; { initialize }


function stackempty { returning } : boolean;

begin { stackempty }

   if top = 0
      then
         stackempty := true
      else
         stackempty := false

end; { stackempty }


function stackfull { returning } : boolean;

begin { stackfull }

   if top = maxstacksize
      then
         stackfull := true
      else
         stackfull := false

end; { stackfull }


procedure popstack( { returning } var indentsymbol : keysymbol;
                                  var prevmargin   : integer);

begin { popstack }

   if not stackempty
      then
         begin

            indentsymbol := stack[top].indentsymbol;
            prevmargin   := stack[top].prevmargin;

            top := top - 1

         end

      else
         begin
            indentsymbol := othersym;
            prevmargin   := 0
         end

end; { popstack }


procedure pushstack( { using } indentsymbol : keysymbol;
                                 prevmargin   : integer   );

begin { pushstack }

   top := top + 1;

   stack[top].indentsymbol := indentsymbol;
   stack[top].prevmargin   := prevmargin

end; { pushstack }


procedure writecrs( { using }              numberofcrs : integer;
                    { updating }   var currlinepos : integer
                    { writing to output }                       );

var
    i: integer;


begin { writecrs }

   if numberofcrs > 0
      then
         begin

            for i := 1 to numberofcrs do
               writeln(output);

            currlinepos := 0

         end

end; { writecrs }


procedure insertcr( { updating }   var currsym     : symbolinfo
                    { writing to output }                           );

const
      once = 1;


begin { insertcr }

   if currsym^.crsbefore = 0
      then
         begin

            writecrs( once, { updating }   currlinepos
                            { writing to output }        );

            currsym^.spacesbefore := 0

         end

```

```
end; { insertcr }


procedure insertblankline( { updating }   var currsym : symbolinfo
                           { writing to output }                   );

const
      once  = 1;
      twice = 2;


begin { insertblankline }

   if currsym^.crsbefore = 0
      then
         begin

            if currlinepos = 0
               then
                  writecrs( once, { updating }   currlinepos
                                 { writing to output }      )
               else
                  writecrs( twice, { updating }   currlinepos
                                   { writing to output }      );

            currsym^.spacesbefore := 0

         end

      else
         if currsym^.crsbefore = 1
            then
               if currlinepos > 0
                  then
                     writecrs( once, { updating }   currlinepos
                                      { writing to output }      )

end; { insertblankline }


procedure lshifton( { using } dindentsymbols : keysymset );

var
    indentsymbol: keysymbol;
    prevmargin  : integer;


begin { lshifton }

   if not stackempty
      then
         begin

            repeat

               popstack( { returning } indentsymbol,
                                         prevmargin   );

               if indentsymbol in dindentsymbols
                  then
                     currmargin := prevmargin

            until not(indentsymbol in dindentsymbols)
                   or (stackempty);

            if not(indentsymbol in dindentsymbols)
               then
                  pushstack( { using } indentsymbol,
                                         prevmargin   )

         end

end; { lshifton }


procedure lshift;

var
    indentsymbol: keysymbol;
    prevmargin  : integer;


begin { lshift }

   if not stackempty
      then
         begin
            popstack( { returning } indentsymbol,
                                      prevmargin   );
            currmargin := prevmargin
         end

end; { lshift }


procedure insertspace( { using }      var symbol     : symbolinfo
                       { writing to output }                   );

begin { insertspace }

   if currlinepos < maxlinesize
      then
         begin

            write(output, space);

            currlinepos := currlinepos + 1;

            with symbol^ do
               if (crsbefore = 0) and (spacesbefore > 0)
                  then
                     spacesbefore := spacesbefore - 1

         end

end; { insertspace }


procedure movelinepos( { to }       newlinepos  : integer;
                       { from } var currlinepos : integer
                       { writing to output }             );

var
   i: integer;


begin { movelinepos }

   for i := currlinepos+1 to newlinepos do
      write(output, space);

   currlinepos := newlinepos
```

```
end; { movelinepos }


procedure printsymbol( { in }             currsym     : symbolinfo;
                       { updating }   var currlinepos : integer
                       { writing to output }                       );

var
   i: integer;


begin { printsymbol }

   with currsym^ do
      begin

         for i := 1 to length do
            write(output, valu[i]);

         startpos := currlinepos; { save start pos for tab purposes } <<<.
         currlinepos := currlinepos + length

      end { with }

end; { printsymbol }


procedure ppsymbol( { in }             currsym    : symbolinfo
                    { writing to output }                      );

const
      once  = 1;

var
    newlinepos: integer;


begin { ppsymbol }

   with currsym^ do
      begin

         writecrs( { using }      crsbefore,
                   { updating }   currlinepos
                   { writing to output }      );

         if (currlinepos + spacesbefore > currmargin)
            or (name in [ opencomment, closecomment ])
            then
               newlinepos := currlinepos + spacesbefore
            else
               newlinepos := currmargin;

         if newlinepos + length > maxlinesize
            then
               begin

                  writecrs( once, { updating }   currlinepos
                                  { writing to output }       );

                  if currmargin + length <= maxlinesize
                     then
                        newlinepos := currmargin
                     else
                        if length < maxlinesize
                           then
                              newlinepos := maxlinesize - length
                           else
                              newlinepos := 0

               end;

         movelinepos( { to }    newlinepos,
                      { from }  currlinepos
                      { in output }        );

         printsymbol( { in }         currsym,
                      { updating }   currlinepos
                      { writing to output }      )

      end { with }

end; { ppsymbol }


procedure rshifttoclp( { using } currsym : keysymbol );
   forward;

procedure gobble( { symbols from input }
                  { up to }            terminators : keysymset;
                  { updating }     var currsym,
                                         nextsym     : symbolinfo
                  { writing to output }                         );

begin { gobble }

   rshifttoclp( { using } currsym^.name );

   while not(nextsym^.name in (terminators + [endoffile])) do
      begin

         getsymbol( { from input }
                    { updating }  nextsym,
                    { returning } currsym   );

         ppsymbol( { in }         currsym
                   { writing to output }   )

      end; { while }

   lshift

end; { gobble }


procedure rshift( { using } currsym : keysymbol );

begin { rshift }

   if not stackfull
      then
         pushstack( { using } currsym,
                                currmargin);

   { if extra indentation was used, update margin. }    <<<.
   if startpos > currmargin                             <<<.
      then                                              <<<.
         currmargin := startpos;                        <<<.

   if currmargin < slofaill
      then
```

```
         currmargin := currmargin + indent1
      else
         if currmargin < slofail2
            then
               currmargin := currmargin + indent2

end; { rshift }


procedure rshifttoclp;

begin { rshifttoclp }

   if not stackfull
      then
         pushstack( { using } currsym,
                                currmargin);

   currmargin := currlinepos

end; { rshifttoclp }


begin { prettyprint }

   initialize(  top,         currlinepos,
               currmargin, keyword,    dblchars,    dblchar,
               sglchar,    recordseen, currchar,    nextchar,
               currsym,    nextsym,    ppoption );

   crpending := false;

   while (nextsym^.name <> endoffile) do
      begin

         getsymbol( { from input }
                    { updating }  nextsym,
                    { returning } currsym    );

         with ppoption [currsym^.name] do
            begin

               if (crpending and not(crsuppress in optionsselected))
                 or (crbefore in optionsselected)
                  then
                     begin
                        insertcr( { using }       currsym
                                  { writing to output }  );
                        crpending := false
                     end;

               if blanklinebefore in optionsselected
                  then
                     begin
                        insertblankline( { using }       currsym
                                         { writing to output }  );
                        crpending := false
                     end;

               if dindentonkeys in optionsselected
                  then
                     lshifton(dindentsymbols);

               if dindent in optionsselected
                  then
                     lshift;

               if spacebefore in optionsselected
                  then
                     insertspace( { using }       currsym
                                  { writing to output }    );

               ppsymbol( { in }         currsym
                         { writing to output }    );

               if spaceafter in optionsselected
                  then
                     insertspace( { using }       nextsym
                                  { writing to output }     );

               if indentbytab in optionsselected
                  then
                     rshift( { using } currsym^.name );

               if indenttoclp in optionsselected
                  then
                     rshifttoclp( { using } currsym^.name );

               if gobblesymbols in optionsselected
                  then
                     gobble( { symbols from input }
                             { up to }         gobbleterminators,
                             { updating }      currsym,
                                                 nextsym
                             { writing to output }                 );

               if crafter in optionsselected
                  then
                     crpending := true

            end { with }

      end; { while }

   if crpending
      then
         writeln(output)

end.
```

S - 4 Format

PASCAL PROGRAM FORMATTER

- Michael N. Condict
- Rick L. Marcus
- Andy Mickel

What Format Does

Format is a flexible prettyprinter for Pascal programs. It takes as input a syntactically-correct Pascal program and produces as output an equivalent but reformatted Pascal program. The resulting program consists of the same sequence of Pascal symbols and comments, but they are rearranged with respect to line boundaries and columns for readability.

Format maintains consistent spacing between symbols, breaks control and data structures onto new lines if necessary, indents lines to reflect the syntactic level of statements and declarations, and more. Miscellaneous features such as supplying line-numbers and automatic comments, or deleting all unnecessary blanks to save space, are described below.

The flexibility of Format is accomplished by allowing you to supply various directives (options) which override the default values. Rather than being a rigid prettyprinter which decides for you how your program is to be formatted, you have the ability to control how formatting is done, not only prior to execution but also during execution through the use of prettyprinter directives embedded in your program.

Experience with Format over the last three years has shown that most users can find a set of values for the directives which produce satisfactory results. The default values are typical.

How To Use Format

The use of Format will vary from implementation to implementation, but will involve one major input file containing a Pascal program and one output file for the reformatted program. Additionally it may be possible to supply the initial values of directives to Format when it begins execution.

Directives to Format may always be specified in the program itself inside comments with a special syntax. Thus the first line of a program is an ideal spot for a comment containing directives. Subsequent use of embedded directives allows you to change the kind of formatting for different sections of your program. The syntax of these special comments is given below (The syntax is given using "EBNF"--Extended Backus-Naur Form--see Communications ACM, November, 1977, page 822.):

```
DirectiveComment = "(*" DirectiveList "*)" |
                      "(*$" CompilerOptionList CommentText DirectiveList "*)".

  DirectiveList = "[" Directive {"," Directive} "]" CommentText.

      Directive = Letter Setting.

         Letter = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
                  "I" | "L" | "N" | "P" | "R" | "S" | "W".

        Setting = Switch | Value | Range.

         Switch = "+" | "-" .

          Value = "=" UnsignedInteger.

          Range = "=" UnsignedInteger "-" UnsignedInteger ["<" | ">"].

UnsignedInteger = Digit{Digit}.

    CommentText = {Any character except "]" or close-comment}.
```

Note: As defined above, a Directive may be within a comment specifying a Pascal CompilerOptionList. On most implementations this is a "$" followed by a series of letters and values ("+", "-", or digits), separated by commas. See your local manual.

Examples of DirectiveComments:

```
     (*[A=15, E=3, N=1,1<]*)  -  good for publication quality.
     (*[G=0, W=1-100, C+]*)  -  good for compact storage.
     (*$U+ [R=1-72, I=2]*)  - an example of a DirectiveList with a
                              CompilerOptionList.
```

Directives to Format

A=n Align declarations.
The A directive forces the alignment of ":" and "=" in declarations. If A is set to a value greater than 0, then n should be equal to the maximum identifier length for that section of your program. The A directive visually clarifies the declaration part of your program. See example below.
Default: A=0 (no alignment).

B+ or B- Bunch statements and declarations reasonably.
B+ will place as many statements or declarations onto one line as will fit within the specified write margins (W directive) subject to readability constraints. Bunching (B+) when the display is off (D-) has no effect. In general, B+ saves paper and prevents your program from becoming overly stretched in the vertical direction. See example below.
Default: B- (one statement or statement part per line).

C+ or C- Fully Compress program.
C+ removes all non-essential blanks, end-of-lines, and comments from your program. A compilable, packed program will be written within the specified write margins (W directive). The number of spaces specified by the G directive will still be written between symbols. C+ might save some space on long-term storage media such as disk; you might store a program in compressed form and expand it later by reformatting with C-.
Default: C-.

D+ or D- Turn Display on or off.
D allows you to selectively display portions of your program during formatting. Therefore, D must be switched on and off with directives which are appropriately placed in your program. D is perhaps useful to obtain program fragments for publication (such as one or more procedures) without having to print the whole program.
Default: D+.

E=n Supply END comments.
The E directive generates comments after "END" symbols if none are already there. Common Pascal coding styles frequently employ these comments. E=1 creates comments after the "END" symbol in compound statements which are within structured statements, as well as those constituting procedure and function bodies. The comments take the form: (*StatementPart*) or (*ProcedureName*). E=2 creates comments after the "BEGIN" and "END" symbols constituting procedure and function bodies only. E=0 creates no comments at all. E=3 means E=1 and E=2. See example below.
Default: E=2.

F+ or F- Turn Formatting on or off.
F allows you to format selected portions of your program. F- causes Format to copy the input program directly with no changes. Therefore by switching F on and off with directives which are appropriately placed in your program, you can preserve text which is already properly formatted (such as comments).
Default: F+ (of course!).

G=n Specify symbol Gap.
The G directive determines the number of spaces placed between Pascal symbols during formatting. G=0 still places one space between two identifiers and reserved words. The symbols [ ] ( ) , and : are handled independently of G.
Default: G=1.

I=n Specify Indent tab.
I indents each nesting level of statements and declarations a given number of columns. Using I=2 or I=1 helps prevent excessively-narrow lines within the specified write margins (W directive) where there are heavily-nested constructs.
Default: I=3.

L=n Specify Line-wraparound indent tab.
L determines the indentation of the remainder of statements or declarations which are too long to fit on one line.
Default: L=3.

N=x-y< or N=x-y> Generate line-numbers on the left or right.
The N directive indicates the starting line-number (x) and the increment (y) for

each succeeding line-number. If y > 0 then line-numbers are written outside the specified write margins for the formatted program in sequential order starting at x; y = 0 shuts off line-numbering. "<" writes up to 4-digit, right-justified line-numbers together with a trailing space to the left of each line. ">" writes 6-digit, zero-filled line-numbers to the right of each line. Use the N directive along with the W directive.
Default: N=0-0> (no line-numbers).

P=n Specify spacing between Procedure and function declarations.
The P directive determines the number of blank lines to be placed between procedure and function declarations. n>2 makes procedures and functions visually stand out.
Default: P=2.

R=x-y Specify Read margins.
The R directive indicates which columns are significant when Format reads from input file. R allows Format to accept files which have line-numbers in the first (x-1) columns or after the yth column.
Default: R=1-999 (large enough to read to end-of-line in most cases).

S=n Specify Statement separation.
The S directive determines the number of spaces between statements bunched on the same line by the use of the B+ directive. Note that this directive is only in effect if B+ is used.
Default: S=3.

W=x-y Specify Write margins.
The W directive indicates which columns are used for writing the reformatted program on the output file. Any line-numbers generated (N directive) are written outside these margins.
Default: W=1-72.

Examples

The A Directive

Here is a sample program fragment before using Format:

```
PROGRAM SAMPLE(OUTPUT);
CONST A=6; ABC='LETTERS'; THREE=3;
TYPE RANGE=1..6;
COLOUR=(RED,BLUE);
VAR
I,I2,I33,I444,I5555:RANGE;
YES,NO,MAYBE:BOOLEAN;
BEGIN END.
```

Here is the output from Format with all defaults set:

```
PROGRAM SAMPLE(OUTPUT);

CONST
   A = 6;
   ABC = 'LETTERS';
   THREE = 3;

TYPE
   RANGE = 1 .. 6;
   COLOUR =
      (RED, BLUE);

VAR
   I, I2, I33, I444, I5555: RANGE;
   YES, NO, MAYBE: BOOLEAN;

BEGIN
END (*SAMPLE*).
```

Here is the output from Format having added a line with the A=5 directive:

```
(*[A=5] ALIGN DECLARATIONS.  *)
PROGRAM SAMPLE(OUTPUT);

CONST
      A = 6;
    ABC = 'LETTERS';
  THREE = 3;

TYPE
  RANGE = 1 .. 6;
 COLOUR = (RED, BLUE);

VAR
       I,
      I2,
     I33,
    I444,
   I5555: RANGE;
     YES,
      NO,
   MAYBE: BOOLEAN;

BEGIN
END (*SAMPLE*).
```

The B Directive

If the input to Format is:

```
PROGRAM T(OUTPUT);
CONST INCREMENT = 5;
VAR I,J,N:INTEGER;
BEGIN
N:=0;
J:=3; I:=SQR(N); N:=N+INCREMENT;
IF N>73 THEN BEGIN DOTHIS; DOTHAT END ;
IF N>5 THEN  IF J>6 THEN  DOSOMETHINGELSE;
END.
```

then the output from Format (using the default, B-) is:

```
PROGRAM T(OUTPUT);

CONST
   INCREMENT = 5;

VAR
   I, J, N: INTEGER;

BEGIN
   N := 0;
   J := 3;
   I := SQR(N);
   N := N + INCREMENT;
   IF N > 73 THEN
      BEGIN
         DOTHIS;
         DOTHAT
      END;
   IF N > 5 THEN
      IF J > 6 THEN
         DOSOMETHINGELSE;
END (*T*).
```

and the output from Format with B directives embedded is:

```
(*[B+] BUNCH STATEMENTS.  *)
PROGRAM T(OUTPUT);

CONST
   INCREMENT = 5;

VAR
   I, J, N: INTEGER;

BEGIN
   N := 0;   J := 3;   I := SQR(N);   N := N + INCREMENT;
   IF N > 73  THEN BEGIN DOTHIS;   DOTHAT END;
(*[B-] UNBUNCH.  *)
   IF N > 5 THEN
      IF J > 6 THEN
         DOSOMETHINGELSE;
END (*T*).
```

The E Directive

Suppose that a Pascal program fragment looked like:

```
PROCEDURE SAMPLE;
  PROCEDURE INNER;
  BEGIN END;
BEGIN
  IF X=3 THEN
            BEGIN X:=1; I:=I+1
            END
         ELSE
            BEGIN X:=X+I; I:=0
            END;
  WHILE (CH<>'X') AND FLAG1 DO
    BEGIN I:=I+3; INNER END; END;
```

then using Format with E=3 produces:

```
PROCEDURE SAMPLE;


   PROCEDURE INNER;

      BEGIN
      END (*INNER*);


   BEGIN (*SAMPLE*)
      IF X = 3
      THEN
         BEGIN
            X := 1;
            I := I + 1
         END (*IF*)
      ELSE
         BEGIN
            X := X + I;
            I := 0
         END (*ELSE*);
      WHILE (CH <> 'X') AND FLAG1 DO
         BEGIN
            I := I + 3;
            INNER
         END (*WHILE*);
   END (*SAMPLE*);
```

How Format Works

Format parses your program by performing syntax analysis similar to the Pascal compiler: recursive descent within nested declarations and statements. It gathers characters into a buffer in which the indenting count of each character is maintained. The characters are being continually emptied from the buffer as new ones are added.

Format has limited error-recovery facilities, and no results are guaranteed if a syntactically-incorrect program is supplied as input.

The bane of most Pascal prettyprinters is the treatment of comments. Format considers them in the context of a declaration or statement. Therefore using comments like:

```
CONST  LS=6 (*LINESIZE*);
```

is a good idea because Format will carry the comment along with the declaration. Similarly:

```
BEGIN (* 'Z' < CH <= ' ' *)
```

is also okay.

Stand-alone comments however, receive rough treatment from Format. The first line of such comments are always left-justified and placed on a separate line. See the F directive. Thus:

```
CONST  LS=6; (*LINESIZE*)
```

will be reformatted as:

```
CONST
   LS = 6;
(*LINESIZE*)
```

Proper treatment of comments is certainly an area of future development for Format.

Format issues the following error messages:

1. " *** 'PROGRAM' EXPECTED."
The Pascal program you fed to Format did not contain a Standard Pascal program declaration.

2. " *** ERRORS FOUND IN PASCAL PROGRAM."
Your program is syntactically incorrect. The output from Format probably does not contain all of the text from your input file. The cause could be any syntactic error, most commonly unmatched "BEGIN-END" pairs, or the lack of semicolons, string quotation marks, or the final period.

3. " *** STRING TOO LONG."
Your program contains a character string (including both the quotes) which is wider than the specified write margins (W directive).

4. " *** NO PROGRAM FOUND TO FORMAT."
The input file given to Format is empty.

History

Format was originally written in 1975 by Michael Condict as a class project in a Pascal programming course taught by Richard Cichelli at Lehigh University using CDC-6000 Pascal. After that, making improvements and adding directives became, temporarily, an obsession with the author (note limited usefulness of the D directive). Fortunately, the program eventually stabilized and is now in general use by Pascal programmers at Lehigh University and other institutions. After graduation the author transported Format in 1977 to a PDP-11 running under the Swedish Pascal compiler and RSX-11 with a total effort of 2 days.

Assistance in bringing up Format may be obtained by writing to Michael Condict at Pattern Analysis and Recognition Corporation, 228 Liberty Plaza, Rome, NY 13440. Format has been made as portable as possible, but portable programs are hampered by non-standard character sets and non-standard techniques for associating program objects (e.g. file variables) with operating system objects (e.g. files as mass-storage devices).

The PDP-11 version of Format uses a procedure:
"ConnectFileVarsToExtFiles"
which serves a function similar to standard Pascal program headers for external files. This version accepts initial values for directives after it types a prompt for you at your interactive terminal.

On the other hand, the CDC-6000 version accepts initial values for directives after a "/" on the operating system control statement which executes Format.

Format was modified for inclusion with Release 3 of CDC-6000 Pascal by Rick L. Marcus and Andy Mickel, University Computer Center, University of Minnesota, in November, 1978.

```
{[A=20,B+,R=1-100,I=2,S=2] FORMATTER DIRECTIVES. }
{
   *************************************************************
   *                                                           *
   *      P A S C A L   P R O G R A M   F O R M A T T E R      *
   *      -----------------------------------------------      *
   *                                                           *
   *               AUTHOR: MICHAEL N. CONDICT, 1975.           *
   *                       LEHIGH UNIVERSITY                   *
   *      CURRENT ADDRESS: PAR CORP.                           *
   *                       228 LIBERTY PLAZA                   *
   *                       ROME, NY 13440                      *
   *                                                           *
   *              UPDATED: AUGUST, 1978.                       *
   *                                                           *
   *************************************************************
}
program Format(Input, Output);

label
  13;

const
             AlfaLeng = 10;
{ !!!!!!!! IMPLEMENTATION DEPENDENCY: !!!!!!!! }
{ MINIMUM AND MAXIMUM Char VALUES. }
              MinChar = 0;
              MaxChar = 127;
       LastPascSymbol = 29;
{ THE FOLLOWING CONSTANTS MUST ALL BE CHANGED TOGETHER, SO THAT THEIR
   VALUES AGREE WITH THEIR NAMES:
}
           BufferSize = 160;
             BuffSzP1 = 161;
             BuffSzM1 = 159;
          BuffSzDiv10 = 16;
{                           }
      MaxReadRightCol = 999;
     MaxWriteRightCol = 72;

type
                 Alfa = packed array [1 .. AlfaLeng] of Char;
{ !!!!!!!! IMPLEMENTATION DEPENDENCY: !!!!!!!! }
{ SET SIZE MAY NOT ALLOW SET OF Char. }
              CharSet = set of Char;
         StatmntTypes = (ForWithWhileStatement, RepeatStatement,
                         IfStatement, CaseStatement, CompoundStatement,
                         OtherStatement);
              Symbols = (ProgSymbol, Comment, BeginSymbol, EndSymbol,
                         Semicolon, ConstSymbol, TypeSymbol,
                         RecordSymbol, ColonSymbol, EqualSymbol,
                         PeriodSymbol, Range, CaseSymbol, OtherSymbol,
                         IfSymbol, ThenSymbol, ElseSymbol, DoSymbol,
                         OfSymbol, ForSymbol, WithSymbol, WhileSymbol,
                         RepeatSymbol, UntilSymbol, Identifier,
                         VarSymbol, ProcSymbol, FuncSymbol, LeftBracket,
                         RightBracket, CommaSymbol, LabelSymbol,
                         LeftParenth, RightParenth, AlphaOperator);
                Width = 0 .. BufferSize;
              Margins = - 100 .. BufferSize;
            SymbolSet = set of Symbols;
           OptionSize = - 99 .. 99;
          CommentText = array [1 .. BuffSzDiv10] of Alfa;
         SymbolString = array [Width] of Char;

var
              ChIsEOL,
           NextChIsEOL: Boolean;
                     I: Integer {USED AS FOR LOOP INDEX};
            Character: Char;
           ReadColumn,
         ReadRightCol: 0 .. 1000;
            OutputCol,
          WriteColumn,
           LeftMargin,
     ActualLeftMargin,
          ReadLeftCol,
         WriteLeftCol,
        WriteRightCol: Margins;
          DisplayIsOn,
      ProcNamesWanted,
    EndCommentsWanted,
          PackerIsOff,
           SavedBunch,
          BunchWanted,
         NoFormatting: Boolean;
           LineNumber,
            Increment: Integer;
          IndentIndex,
       LongLineIndent,
            SymbolGap,
      DeclarAlignment,
     StatmtSeparation,
       ProcSeparation: OptionSize;
           LastSymbol,
           SymbolName: Symbols;
         AlphaSymbols,
             EndLabel,
             EndConst,
              EndType,
               EndVar: SymbolSet;
               Symbol: SymbolString;
               Length: Width;
       SymbolIsNumber,
  LastProgPartWasBody: Boolean;
               Digits,
     LettersAndDigits: CharSet;
               Oldest: Width;
            CharCount: Integer { COUNT OF TOTAL CHARS READ,
                  BUT IS OFF BY BufferSize AFTER END OF FIRST BODY.
                  IT IS IMPERATIVE THAT CharCount BE MONOTONICALLY
                  INCREASING DURING PROCESSING OF A BODY, AND THAT IT
                  NEVER RETURN TO A VALUE <= BufferSize, AFTER PASSING
                  THAT POINT.  THUS "DoBlock" MAY RESET IT AS LOW AS
                  POSSIBLE, LIMITING THE LENGTH OF A PROCEDURE TO
                  "MaxInt - BufferSize" CHARACTERS. };
                 Main: CommentText;
         MainNmLength: Width;
               Blanks,
               Zeroes: Alfa;
            UnWritten: array [Width] of record
                                                       Ch: Char;
                                              ChIsEndLine: Boolean;
                                           IndentAfterEOL: Margins
                                        end;
         PascalSymbol: array [1 .. LastPascSymbol] of Alfa;
       PascSymbolName: array [1 .. LastPascSymbol] of Symbols;
               NameOf: array [Char] of Symbols;
      StatementTypeOf: array [Symbols] of StatmntTypes;


procedure ConstantsInitialization;

  begin
    Main[1] := 'MAIN      ';  MainNmLength := 4;
    Blanks := '          ';  Zeroes := '0000000000';
    for I := 0 to BuffSzM1 do
      with UnWritten[I] do
        begin Ch := 'A';  ChIsEndLine := False;  IndentAfterEOL := 0
        end;
    for Character := Chr(MinChar) to Chr(MaxChar) do
      NameOf[Character] := OtherSymbol;
    Character := ' ';  NameOf['('] := LeftParenth;
    NameOf[')'] := RightParenth;  NameOf['='] := EqualSymbol;
    NameOf[','] := CommaSymbol;  NameOf['.'] := PeriodSymbol;
    NameOf['['] := LeftBracket;  NameOf[']'] := RightBracket;
    NameOf[':'] := ColonSymbol;  NameOf['<'] := EqualSymbol;
    NameOf['>'] := EqualSymbol;  NameOf[';'] := Semicolon;
    PascalSymbol[1] := 'PROGRAM   ';  PascalSymbol[2] := 'BEGIN     ';
    PascalSymbol[3] := 'END       ';  PascalSymbol[4] := 'CONST     ';
    PascalSymbol[5] := 'TYPE      ';  PascalSymbol[6] := 'VAR       ';
    PascalSymbol[7] := 'RECORD    ';  PascalSymbol[8] := 'CASE      ';
    PascalSymbol[9] := 'IF        ';  PascalSymbol[10] := 'THEN      ';
    PascalSymbol[11] := 'ELSE      ';  PascalSymbol[12] := 'DO        ';
    PascalSymbol[13] := 'OF        ';  PascalSymbol[14] := 'FOR       ';
    PascalSymbol[15] := 'WHILE     ';  PascalSymbol[16] := 'WITH      ';
    PascalSymbol[17] := 'REPEAT    ';  PascalSymbol[18] := 'UNTIL     ';
    PascalSymbol[19] := 'PROCEDURE ';  PascalSymbol[20] := 'FUNCTION  ';
    PascalSymbol[21] := 'LABEL     ';  PascalSymbol[22] := 'IN        ';
    PascalSymbol[23] := 'MOD       ';  PascalSymbol[24] := 'DIV       ';
    PascalSymbol[25] := 'AND       ';  PascalSymbol[26] := 'OR        ';
    PascalSymbol[27] := 'NOT       ';  PascalSymbol[28] := 'ARRAY     ';
    PascalSymbol[29] := 'NOSYMBOL  ';  PascSymbolName[1] := ProgSymbol;
    PascSymbolName[2] := BeginSymbol;  PascSymbolName[3] := EndSymbol;
    PascSymbolName[4] := ConstSymbol;  PascSymbolName[5] := TypeSymbol;
    PascSymbolName[6] := VarSymbol;  PascSymbolName[7] := RecordSymbol;
    PascSymbolName[8] := CaseSymbol;  PascSymbolName[9] := IfSymbol;
    PascSymbolName[10] := ThenSymbol;  PascSymbolName[11] := ElseSymbol;
    PascSymbolName[12] := DoSymbol;  PascSymbolName[13] := OfSymbol;
    PascSymbolName[14] := ForSymbol;  PascSymbolName[15] := WhileSymbol;
    PascSymbolName[16] := WithSymbol;
    PascSymbolName[17] := RepeatSymbol;
    PascSymbolName[18] := UntilSymbol;
    PascSymbolName[19] := ProcSymbol;  PascSymbolName[20] := FuncSymbol;
    PascSymbolName[21] := LabelSymbol;
    PascSymbolName[29] := Identifier;
    for I := 22 to 28 do PascSymbolName[I] := AlphaOperator;
    for SymbolName := ProgSymbol to AlphaOperator do
      StatementTypeOf[SymbolName] := OtherStatement;
    StatementTypeOf[BeginSymbol] := CompoundStatement;
    StatementTypeOf[CaseSymbol] := CaseStatement;
    StatementTypeOf[IfSymbol] := IfStatement;
    StatementTypeOf[ForSymbol] := ForWithWhileStatement;
    StatementTypeOf[WhileSymbol] := ForWithWhileStatement;
    StatementTypeOf[WithSymbol] := ForWithWhileStatement;
    StatementTypeOf[RepeatSymbol] := RepeatStatement;
  end {ConstantsInitialization};


procedure WriteA(Character: Char);

  var
                      I: Width;
                 TestNo: Integer;

  begin
    CharCount := CharCount + 1;  Oldest := CharCount mod BufferSize;
    with UnWritten[Oldest] do
      begin
        if CharCount > BuffSzP1
        then
          begin
            if ChIsEndLine
            then
              begin
                if IndentAfterEOL < 0
                then
                  begin
                    Write(Blanks: - IndentAfterEOL);
                    OutputCol := OutputCol - IndentAfterEOL;
                  end
                else
                  begin
                    if Increment < 0
                    then
                      begin
                        I := WriteRightCol - OutputCol + 1;
                        if I > 0  then Write(Blanks: I);
                        TestNo := LineNumber;  I := 0;
                        repeat TestNo := TestNo div 10;  I := I + 1;
                        until TestNo = 0;
                        Write(Zeroes: (6 - I), LineNumber: I);
                        LineNumber := LineNumber - Increment;
                        if LineNumber > 9999
                        then LineNumber := LineNumber - 10000;
                        WriteLn;
                      end
                    else
                      begin
                        WriteLn;
                        if Increment > 0
                        then
                          begin
                            Write(LineNumber: 4, ' ');
                            LineNumber := LineNumber + Increment;
                          end
                      end;
                    if IndentAfterEOL > 0
                    then Write(Blanks: IndentAfterEOL);
                    OutputCol := IndentAfterEOL + 1;
                  end;
                ChIsEndLine := False;
              end {IF ChIsEndLine}
            else
              begin Write(Ch);  OutputCol := OutputCol + 1;
              end {ELSE};
          end {IF CharCount > };
        Ch := Character;  WriteColumn := WriteColumn + 1;
      end {WITH};
  end {WriteA};


procedure FlushUnwrittenBuffer;

  begin
    WriteA(' ');
    with UnWritten[Oldest] do
      begin ChIsEndLine := True;  IndentAfterEOL := 0; end;
    WriteColumn := 0;  for I := 0 to BuffSzM1 do WriteA(' ');
  end {FlushUnwrittenBuffer};


procedure StartNewLineAndIndent;

  begin
    if PackerIsOff and DisplayIsOn
    then
      begin
        WriteA(' ');  LastSymbol := PeriodSymbol;
        with UnWritten[Oldest] do
          begin
            ChIsEndLine := True;
            IndentAfterEOL := WriteLeftCol + LeftMargin - 1;
          end;
        WriteColumn := WriteLeftCol + LeftMargin;
      end {IF PackerIsOff};
  end {StartNewLineAndIndent};


procedure ReadACharacter;

  begin
    if ReadColumn > ReadRightCol
    then
      begin
        if ReadRightCol < MaxReadRightCol
        then begin NextChIsEOL := True;  ReadLn end
        else ReadColumn := 2;
      end
    else
      if ReadColumn = 1 then
        while ReadColumn < ReadLeftCol do
          begin
            if EOLn(Input)  then ReadColumn := 1
            else begin ReadColumn := ReadColumn + 1;  Get(Input) end
          end;
    if NextChIsEOL
    then
      begin
        Character := ' ';  NextChIsEOL := False;  ChIsEOL := True;
        ReadColumn := 1;
        if NoFormatting
        then
          begin
            WriteA(' ');
            with UnWritten[Oldest] do
              begin
                ChIsEndLine := True;
                IndentAfterEOL := WriteLeftCol - 1;
              end;
            WriteColumn := WriteLeftCol;
          end;
      end
    else
      if not EOF(Input)
      then
        begin
          Character := Input ^;  ReadColumn := ReadColumn + 1;
          NextChIsEOL := EOLn(Input);  Get(Input);  ChIsEOL := False;
          if NoFormatting  then WriteA(Character);
        end
      else begin FlushUnwrittenBuffer;  goto 13 end
  end {ReadACharacter};


procedure WriteSymbol;

  var
                       I: Width;
     NumberBlanksToWrite: OptionSize;

  begin
    if DisplayIsOn
    then
      begin
        NumberBlanksToWrite := SymbolGap;
        if (LastSymbol in [LeftParenth, LeftBracket, PeriodSymbol]) or
           (SymbolName in [Semicolon, RightParenth, RightBracket,
           CommaSymbol, PeriodSymbol, ColonSymbol]) or (SymbolName in
           [LeftBracket, LeftParenth]) and (LastSymbol = Identifier)
        then NumberBlanksToWrite := 0
        else
          if (SymbolName in AlphaSymbols) and (LastSymbol in
             AlphaSymbols)
          then
            if WriteColumn <= WriteRightCol then
              begin WriteA(' ');  NumberBlanksToWrite := SymbolGap - 1;
              end;
        if WriteColumn + Length + NumberBlanksToWrite - 1 >
           WriteRightCol
        then
          begin
            WriteA(' ');
            with UnWritten[Oldest] do
              begin
                ChIsEndLine := True;
                if PackerIsOff
                then
                  begin
                    if WriteLeftCol + LeftMargin + LongLineIndent +
                       Length - 1 > WriteRightCol
                    then Length := 10;
                    IndentAfterEOL := WriteLeftCol - 1 + LeftMargin +
                      LongLineIndent;
                    WriteColumn := WriteLeftCol + LeftMargin +
                      LongLineIndent;
                  end
                else
                  begin
                    if Length > WriteRightCol - WriteLeftCol + 1
                    then Length := WriteRightCol - WriteLeftCol + 1;
                    IndentAfterEOL := WriteLeftCol - 1;
                    WriteColumn := WriteLeftCol;
                  end;
              end {WITH};
          end
        else for I := 1 to NumberBlanksToWrite do WriteA(' ');
        for I := 1 to Length do WriteA(Symbol[I]);
      end {IF DisplayIsOn};
    LastSymbol := SymbolName;
  end {WriteSymbol};


procedure CopyACharacter;

  begin
    if DisplayIsOn
    then
      begin
        if WriteColumn > WriteRightCol then
          begin
            while (Character = ' ') and not ChIsEOL do
              ReadACharacter;
            if not ChIsEOL  then StartNewLineAndIndent;
          end;
        if ChIsEOL
        then
          begin
            LeftMargin := 0;  StartNewLineAndIndent;
            LeftMargin := ActualLeftMargin;
          end
        else WriteA(Character);
      end;
    ReadACharacter
  end {CopyACharacter};


procedure DoFormatterDirectives;

  const
                Invalid = - 1;

  type
             ParamCount = 1 .. 2;
                 Params = array [ParamCount] of Integer;

  var
           Specification: Params;
            FormatOption: Char;
             PrevDisplay,
        PrevNoFormatting: Boolean;
              EndDirectv: CharSet;


  procedure ReadIn(N: ParamCount; var Specification: Params);

    var
                        I: ParamCount;

  begin
    for I := 1 to N do
      begin
        while not (Character in (Digits + EndDirectv)) do
          CopyACharacter;
        Specification[I] := 0;
        if not (Character in EndDirectv)
        then
          repeat
            Specification[I] := 10 * Specification[I] + Ord(Character)
               - Ord('0');
            CopyACharacter;
          until not (Character in Digits)
        else Specification[I] := Invalid;
      end {FOR};
  end {ReadIn};


begin {DoFormatterDirectives}
  EndDirectv := ['*', ']'];
  repeat
    if Character in ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'I', 'L', 'N',
       'P', 'R', 'S', 'W']
    then
      begin
        FormatOption := Character;
        case FormatOption of
          'A', 'E', 'I', 'G', 'P', 'L', 'S':
            begin
              ReadIn(1, Specification);
              if (Specification[1] < WriteRightCol - WriteLeftCol - 9)
                 or (FormatOption = 'P')
              then
                case FormatOption of
                  'A': DeclarAlignment := Specification[1];
                  'E':
                    if Specification[1] < 4 then
                      begin
                        ProcNamesWanted := Specification[1] > 1;
                        EndCommentsWanted := Odd(SPECIFICATON[1]);
                      end;
                  'G': SymbolGap := Specification[1];
                  'I': IndentIndex := Specification[1];
                  'L': LongLineInde***ecification[1];
                  'P': ProcSeparation := Specification[1];
                  'S': StatmtSeparation := Specification[1]
                end {CASE};
            end {SINGLE PARAMETERS};
          'W', 'R', 'N':
            begin
              ReadIn(2, Specification);
              if Specification[2] <> Invalid
              then
                case FormatOption of
                  'W':
                    if (Specification[1] > 0) and (Specification[2] <
                       BufferSize - 2) and (Specification[2] -
                       Specification[1] > 8)
                    then
                      begin
                        WriteLeftCol := Specification[1];
                        WriteRightCol := Specification[2];
                      end;
                  'R':
                    if (Specification[1] > 0) and (Specification[2] -
                       Specification[1] > 8)
                    then
                      begin
                        ReadLeftCol := Specification[1];
                        ReadRightCol := Specification[2];
                      end;
                  'N':
                    begin
                      LineNumber := Specification[1];
                      Increment := Specification[2];
                      while not (Character in (['<'] + EndDirectv))
                         and (Character <> '>') do
                        CopyACharacter;
                      if Character = '>'
                      then Increment := - Increment
                    end
                end {CASE};
            end {DOUBLE PARAMETERS};
          'B', 'C', 'D', 'F':
            begin
              repeat CopyACharacter;
              until Character in (['+', '-'] + EndDirectv);
              if Character in ['+', '-']
              then
                case FormatOption of
                  'B':
                    if DisplayIsOn
                    then BunchWanted := Character = '+';
                  'C': PackerIsOff := Character = '-';
                  'D':
                    begin
                      PrevDisplay := DisplayIsOn;
                      DisplayIsOn := Character = '+';
                      if PrevDisplay and not DisplayIsOn
                      then
                        begin
                          WriteA('*');  WriteA(')');
                          SavedBunch := BunchWanted;
                          BunchWanted := False;
                        end
                      else
                        if not PrevDisplay and DisplayIsOn then
                          begin
                            StartNewLineAndIndent;  WriteA('(');
                            WriteA('*');  BunchWanted := SavedBunch;
                          end {IF NOT PREV};
                    end { 'D': };
                  'F':
                    begin
                      PrevNoFormatting := NoFormatting;
                      NoFormatting := Character = '-';
                      DisplayIsOn := not NoFormatting;
                      if PrevNoFormatting and not NoFormatting
                      then ReadACharacter;
                      if not PrevNoFormatting and NoFormatting
                      then WriteA('-');
                    end;
                end {CASE};
            end {Boolean PARAMETERS}
        end {CASE};
      end {THEN}
    else if not (Character in EndDirectv)  then CopyACharacter;
  until Character in EndDirectv;
  if (Character = ']') then CopyACharacter;
end {DoFormatterDirectives};


procedure ReadSymbol;

  const
             ReadNextCh = True;
         DontReadNextCh = False;

  var
              TestSymbol: Alfa;
              CharNumber: Width;
                       I: Width;


  procedure SkipComment;

    begin
      repeat while Character <> '*' do ReadACharacter;  ReadACharacter
      until Character = ')';
      ReadACharacter;  LastSymbol := Comment;  ReadSymbol
    end {SkipComment};


  procedure DoComment;

    var
                         I: OptionSize;


    procedure CompilerDirectives;

      begin repeat CopyACharacter;  until Character in ['[', '*']
      end {CompilerDirectives};


    begin {DoComment}
      begin
        if LastSymbol in [Comment, Semicolon] then
          begin
            LeftMargin := 0;  StartNewLineAndIndent;
            LeftMargin := ActualLeftMargin;
          end;
        WriteSymbol;  if Character = '$'  then CompilerDirectives;
        if Character = '['  then DoFormatterDirectives;
        repeat
          while Character <> '*' do CopyACharacter;  CopyACharacter;
        until Character = ')';
        CopyACharacter;  LastSymbol := Comment;  ReadSymbol;
      end;
    end {DoComment};


  procedure CheckFor(SecondChar: Char; TwoCharSymbol: Symbols;
     ReadAllowed: Boolean);

    begin
      if ReadAllowed then
        begin
          Length := 1;  Symbol[1] := Character;
          SymbolName := NameOf[Character];  ReadACharacter;
        end;
      if Character = SecondChar
      then
        begin
          Symbol[2] := Character;  Length := 2;
          SymbolName := TwoCharSymbol;  ReadACharacter;
          if (not PackerIsOff) and (SymbolName = Comment)
          then Length := 0
        end;
    end {CheckFor};


  begin {ReadSymbol}
    if (Character in ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
       'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W',
       'X', 'Y', 'Z', '0' .. '9', ' ', '(', '.', ':', '''', '<', '>'])
    then
      case Character of
        '(':
          begin
            CheckFor('*', Comment, ReadNextCh);
            if (SymbolName = Comment) and PackerIsOff  then DoComment
            else if SymbolName = Comment  then SkipComment;
          end;
        'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
           'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
           'Z':
          begin
            CharNumber := 1;  SymbolIsNumber := False;
            repeat
              Symbol[CharNumber] := Character;  ReadACharacter;
              CharNumber := CharNumber + 1
            until not (Character in LettersAndDigits);
            Length := CharNumber - 1;
            for CharNumber := CharNumber to AlfaLeng do
              Symbol[CharNumber] := ' ';
            Pack(Symbol, 1, TestSymbol);  I := 1;
            PascalSymbol[LastPascSymbol] := TestSymbol;
            while PascalSymbol[I] <> TestSymbol do I := I + 1;
            SymbolName := PascSymbolName[I];
          end {LETTER};
        '0', '1', '2', '3', '4', '5', '6', '7', '8', '9':
          begin
            SymbolIsNumber := True;  CharNumber := 1;
            repeat
              Symbol[CharNumber] := Character;  ReadACharacter;
              CharNumber := CharNumber + 1
            until not (Character in Digits + ['.']);
            if Character in ['B', 'E']
            then
              begin
                Symbol[CharNumber] := Character;  ReadACharacter;
                CharNumber := CharNumber + 1;
                if Character in Digits + ['+', '-'] then
                  repeat
                    Symbol[CharNumber] := Character;  ReadACharacter;
                    CharNumber := CharNumber + 1
                  until not (Character in Digits)
              end;
            Length := CharNumber - 1;  SymbolName := Identifier;
          end {NUMBER};
        ' ':
          begin
            repeat ReadACharacter  until Character <> ' ';  ReadSymbol
          end;
        '>', ':': CheckFor('=', OtherSymbol, ReadNextCh);
        '<':
          begin
            CheckFor('=', OtherSymbol, ReadNextCh);
            if SymbolName <> OtherSymbol
            then CheckFor('>', OtherSymbol, DontReadNextCh);
          end;
        '.':
          if LastSymbol <> EndSymbol
          then CheckFor('.', Range, ReadNextCh)
          else SymbolName := PeriodSymbol;
        '''':
          begin
            CharNumber := 1;
            repeat
              repeat
                Symbol[CharNumber] := Character;
                CharNumber := CharNumber + 1;  ReadACharacter;
              until Character = '''';
              Symbol[CharNumber] := Character;
              CharNumber := CharNumber + 1;  ReadACharacter;
            until Character <> '''';
            Length := CharNumber - 1;  SymbolName := OtherSymbol;
            if Length > WriteRightCol - WriteLeftCol + 1
            then
              begin
                FlushUnwrittenBuffer;  WriteLn;
                WriteLn(' *** STRING TOO LONG.');
                goto 13
              end;
          end {STRING}
      end {CASE}
    else
      begin
        Symbol[1] := Character;  SymbolName := NameOf[Character];
        Length := 1;  ReadACharacter
      end
  end {ReadSymbol};


procedure ChangeMarginTo(NewLeftMargin: Margins);

  var
      IndentedLeftMargin: Margins;

  begin
    ActualLeftMargin := NewLeftMargin;  LeftMargin := NewLeftMargin;
    if LeftMargin < 0  then LeftMargin := 0
    else
      begin
        IndentedLeftMargin := WriteRightCol - 9 - LongLineIndent;
        if LeftMargin > IndentedLeftMargin
        then LeftMargin := IndentedLeftMargin
      end
  end {ChangeMarginTo};


procedure DoDeclarationUntil(EndDeclaration: SymbolSet);


  procedure DoParentheses;

    var
              SavedLgLnId: OptionSize;

    begin
      SavedLgLnId := LongLineIndent;
      if DeclarAlignment > 0
      then
        begin
          LongLineIndent := WriteColumn + SymbolGap + 1 - LeftMargin -
             WriteLeftCol;
          repeat WriteSymbol;  ReadSymbol;
          until SymbolName = RightParenth;
          WriteSymbol;  ReadSymbol;
      end
    else
      begin
        LongLineIndent := 1;
        ChangeMarginTo(ActualLeftMargin + IndentIndex);
        StartNewLineAndIndent;
        repeat WriteSymbol;  ReadSymbol
        until SymbolName = RightParenth;
        WriteSymbol;  ReadSymbol;
        ChangeMarginTo(ActualLeftMargin - IndentIndex);
      end {ELSE};
    LongLineIndent := SavedLgLnId;
  end {DoParentheses};


procedure DoFieldListUntil(EndFieldList: SymbolSet);

  var
                 LastEOL: Margins;
             AlignColumn: Width;


  procedure DoRecord;

    var
           SavedLeftMargin: Width;

    begin
      SavedLeftMargin := ActualLeftMargin;  WriteSymbol;  ReadSymbol;
      ChangeMarginTo(WriteColumn - 6 + IndentIndex - WriteLeftCol);
      StartNewLineAndIndent;  DoFieldListUntil([EndSymbol]);
      ChangeMarginTo(ActualLeftMargin - IndentIndex);
      StartNewLineAndIndent;  WriteSymbol;  ReadSymbol;
      ChangeMarginTo(SavedLeftMargin);
    end {DoRecord};


  procedure DoVariantRecordPart;

    var
           SavedLeftMargin,
          OtherSavedMargin: Margins;

    begin
      OtherSavedMargin := ActualLeftMargin;
      if DeclarAlignment > 0
      then
        begin
          repeat WriteSymbol;  ReadSymbol;
          until SymbolName in [ColonSymbol, OfSymbol];
          if SymbolName = ColonSymbol
          then
            begin
              WriteSymbol;  ReadSymbol;
              with UnWritten[LastEOL] do
                begin
                  IndentAfterEOL := IndentAfterEOL + AlignColumn -
                     WriteColumn;
                  if IndentAfterEOL < 0  then IndentAfterEOL := 0;
                end;
              WriteColumn := AlignColumn;
              ChangeMarginTo(ActualLeftMargin + AlignColumn -
                 WriteColumn);
            end;
        end;
      if SymbolName <> OfSymbol then
        repeat WriteSymbol;  ReadSymbol;  until SymbolName = OfSymbol;
      ChangeMarginTo(ActualLeftMargin + IndentIndex);
      repeat
        WriteSymbol;  ReadSymbol;
        if SymbolName <> EndSymbol
        then
          begin
            StartNewLineAndIndent;
            repeat WriteSymbol;  ReadSymbol;
            until SymbolName in [LeftParenth, Semicolon, EndSymbol];
            if SymbolName = LeftParenth
            then
              begin
                WriteSymbol;  ReadSymbol;
                SavedLeftMargin := ActualLeftMargin;
                ChangeMarginTo(WriteColumn - WriteLeftCol);
                DoFieldListUntil([RightParenth]);  WriteSymbol;
                ReadSymbol;  ChangeMarginTo(SavedLeftMargin);
              end;
          end;
      until SymbolName <> Semicolon;
      ChangeMarginTo(OtherSavedMargin);
    end {DoVariantRecordPart};


  begin {DoFieldListUntil}
    LastEOL := Oldest;
    if LastSymbol = LeftParenth
    then for I := 1 to DeclarAlignment - Length do WriteA(' ');
    AlignColumn := LeftMargin + WriteLeftCol + DeclarAlignment + 1;
    while not (SymbolName in EndFieldList) do
      begin
        if LastSymbol in [Semicolon, Comment] then
          if SymbolName <> Semicolon
          then begin StartNewLineAndIndent;  LastEOL := Oldest end;
        if SymbolName in [RecordSymbol, CaseSymbol, LeftParenth,
           CommaSymbol, ColonSymbol, EqualSymbol]
        then
          case SymbolName of
            RecordSymbol: DoRecord;
            CaseSymbol: DoVariantRecordPart;
            LeftParenth: DoParentheses;
            CommaSymbol, ColonSymbol, EqualSymbol:
              begin
                WriteSymbol;
                if DeclarAlignment > 0
                then
                  if not (EndLabel <= EndFieldList)
                  then
                    begin
                      with UnWritten[LastEOL] do
                        begin
                          IndentAfterEOL := IndentAfterEOL +
                             AlignColumn - WriteColumn;
                          if IndentAfterEOL < 0
                          then IndentAfterEOL := 0;
                          WriteColumn := AlignColumn;
                        end;
                      if SymbolName = CommaSymbol then
                        begin
                          StartNewLineAndIndent;  LastEOL := Oldest;
                        end;
                    end {IF DeclarAlignment};
                ReadSymbol;
              end {  ,   :   = }
          end {CASE}
          else begin WriteSymbol;  ReadSymbol end;
        end {WHILE};
    end {DoFieldListUntil};


  begin {DoDeclarationUntil}
    StartNewLineAndIndent;  WriteSymbol;
    ChangeMarginTo(ActualLeftMargin + IndentIndex);
    StartNewLineAndIndent;  ReadSymbol;
    DoFieldListUntil(EndDeclaration);  StartNewLineAndIndent;
    ChangeMarginTo(ActualLeftMargin - IndentIndex);
  end {DoDeclarationUntil};


procedure DoBlock(BlockName: CommentText; BlockNmLength: Width);

  var
                       I: Width;
       IfThenBunchNeeded: Boolean;
         AtProcBeginning: Boolean;


  procedure DoProcedures;

    var
                         I: 0 .. 20;
                  ProcName: CommentText;
              ProcNmLength: Width;

    begin
      for I := 2 to ProcSeparation do StartNewLineAndIndent;
      StartNewLineAndIndent;  WriteSymbol;  ReadSymbol;
      for I := 0 to (Length - 1) div AlfaLeng do
        Pack(Symbol, I * AlfaLeng + 1, ProcName[I + 1]);
      ProcNmLength := Length;  WriteSymbol;  ReadSymbol;
      if SymbolName = LeftParenth then
        begin
          WriteSymbol;
          repeat ReadSymbol;  WriteSymbol
          until SymbolName = RightParenth;
          ReadSymbol;
        end;
      if SymbolName = ColonSymbol then
        repeat WriteSymbol;  ReadSymbol;  until SymbolName = Semicolon;
      WriteSymbol;  ReadSymbol;
      ChangeMarginTo(ActualLeftMargin + IndentIndex);
      StartNewLineAndIndent;  LastProgPartWasBody := False;
      DoBlock(ProcName, ProcNmLength);  LastProgPartWasBody := True;
      ChangeMarginTo(ActualLeftMargin - IndentIndex);  WriteSymbol;
      ReadSymbol;  StartNewLineAndIndent;
    end {DoProcedures};


  procedure DoStatement(var AddedBlanks: Width; StatmtSymbol:
     CommentText; StmtSymLength: Width);

    var
                         I: Width;
           StatmtBeginning: Integer;
                StatmtPart: array [1 .. 4] of Integer;
          BlksOnCurrntLine,
       BlksAddedByThisStmt: Integer;
                Successful: Boolean;


    procedure Bunch(Beginning, Breakpt, Ending: Integer;
       StatmtSeparation: OptionSize);

      begin
        if BunchWanted or IfThenBunchNeeded
        then
          begin
            if StatmtSeparation < 1  then StatmtSeparation := 1;
            BlksOnCurrntLine := BlksOnCurrntLine + StatmtSeparation - 1;
            Successful := ((Ending - Beginning + BlksOnCurrntLine +
               UnWritten[Beginning mod BufferSize].IndentAfterEOL) <
               WriteRightCol) and (CharCount - Beginning < BufferSize);
            if Successful
            then
              begin
                BlksAddedByThisStmt := BlksAddedByThisStmt +
                   StatmtSeparation - 1;
                UnWritten[Breakpt mod BufferSize].IndentAfterEOL := -
                   StatmtSeparation;
              end;
          end;
      end {Bunch};


    procedure WriteComment;

      var
                             I: Width;
                   SavedLength: Width;
               SavedSymbolName: Symbols;
                    SavedChars: SymbolString;

      begin
        SavedSymbolName := SymbolName;
        for I := 1 to Length do SavedChars[I] := Symbol[I];
        SavedLength := Length;  SymbolName := OtherSymbol;
        Symbol[1] := '(';  Symbol[2] := '*';  Length := 2;  WriteSymbol;
        for I := 0 to (StmtSymLength - 1) div AlfaLeng do
          Unpack(StatmtSymbol[I + 1], Symbol, (I * AlfaLeng + 1));
        Length := StmtSymLength;  SymbolName := PeriodSymbol;
        LastSymbol := PeriodSymbol;  WriteSymbol;  Symbol[1] := '*';
        Symbol[2] := ')';  Length := 2;  WriteSymbol;
        SymbolName := SavedSymbolName;  Length := SavedLength;
        for I := 1 to Length do Symbol[I] := SavedChars[I];
      end {WriteComment};


    procedure DoStmtList(EndList: Symbols);

      var
               BlksAfterPrt2: Width;
                   AtProcEnd: Boolean;

      begin
        AtProcEnd := AtProcBeginning;  WriteSymbol;  ReadSymbol;
        StatmtPart[1] := CharCount + 1;  StatmtPart[2] := StatmtPart[1];
        if SymbolName <> EndList
        then
          begin
            if ProcNamesWanted then
              if AtProcBeginning then
                if LastProgPartWasBody
                then if LastSymbol = BeginSymbol  then WriteComment;
            AtProcBeginning := False;
            DoStatement(AddedBlanks, StatmtSymbol, StmtSymLength);
            BlksAfterPrt2 := AddedBlanks;
            BlksAddedByThisStmt := BlksAddedByThisStmt + AddedBlanks;
        while SymbolName <> EndList do
          begin
            WriteSymbol;  ReadSymbol;
            if SymbolName <> EndList
            then
              begin
                StatmtPart[3] := CharCount + 1;
                DoStatement(AddedBlanks, StatmtSymbol,
                  StmtSymLength);
                BlksOnCurrntLine := AddedBlanks + BlksAfterPrt2;
                BlksAddedByThisStmt := BlksAddedByThisStmt +
                   AddedBlanks;
                Bunch(StatmtPart[2], StatmtPart[3], CharCount,
                   StatmtSeparation);
                if not Successful
                then
                  begin
                    BlksAfterPrt2 := AddedBlanks;
                    StatmtPart[2] := StatmtPart[3];
                  end
                else BlksAfterPrt2 := BlksOnCurrntLine;
              end;
          end {WHILE SymbolName <> EndList};
      end {IF SymbolName <> EndList};
    BlksOnCurrntLine := BlksAddedByThisStmt;
    Bunch(StatmtBeginning, StatmtPart[1], CharCount, SymbolGap);
    StartNewLineAndIndent;  StatmtPart[1] := CharCount;
    repeat WriteSymbol;  ReadSymbol;
    until SymbolName in [Semicolon, UntilSymbol, EndSymbol,
    ElseSymbol, PeriodSymbol];
    if Successful
    then
      begin
        if EndList = UntilSymbol
        then StatmtPart[4] := StatmtSeparation
        else StatmtPart[4] := SymbolGap;
        Bunch(StatmtBeginning, StatmtPart[1], CharCount,
           StatmtPart[4]);
      end {IF Successful};
    if not (Successful and BunchWanted)
    then
      if EndList = EndSymbol then
        if LastSymbol = EndSymbol then
          if AtProcEnd and ProcNamesWanted  then WriteComment
          else if EndCommentsWanted  then WriteComment;
  end {DoStmtList};


begin {DoStatement}
  BlksOnCurrntLine := 0;  Successful := False;
  BlksAddedByThisStmt := 0;
  ChangeMarginTo(ActualLeftMargin + IndentIndex);
  StartNewLineAndIndent;  StatmtBeginning := CharCount;
  if SymbolIsNumber
  then
    begin
      with UnWritten[Oldest] do
        begin
          IndentAfterEOL := IndentAfterEOL - 1 - Length - SymbolGap;
          if IndentAfterEOL < 0  then IndentAfterEOL := 0;
        end;
      WriteSymbol;  ReadSymbol {Write LABEL};  WriteSymbol;
      ReadSymbol {Write COLON};
    end;
  case StatementTypeOf[SymbolName] of
    ForWithWhileStatement:
      begin
        Pack(Symbol, 1, StatmtSymbol[1]);  StmtSymLength := Length;
        repeat WriteSymbol;  ReadSymbol
        until SymbolName = DoSymbol;
        WriteSymbol;  ReadSymbol;  StatmtPart[1] := CharCount + 1;
        DoStatement(AddedBlanks, StatmtSymbol, StmtSymLength);
        BlksOnCurrntLine := BlksOnCurrntLine + AddedBlanks;
        BlksAddedByThisStmt := BlksAddedByThisStmt + AddedBlanks;
        Bunch(StatmtBeginning, StatmtPart[1], CharCount, SymbolGap);
      end;
    RepeatStatement: DoStmtList(UntilSymbol);
    IfStatement:
      begin
        Pack(Symbol, 1, StatmtSymbol[1]);  StmtSymLength := Length;
        repeat WriteSymbol;  ReadSymbol
        until SymbolName = ThenSymbol;
        StartNewLineAndIndent;  StatmtPart[1] := CharCount;
        WriteSymbol;  ReadSymbol;  StatmtPart[2] := CharCount + 1;
        DoStatement(AddedBlanks, StatmtSymbol, StmtSymLength);
        BlksOnCurrntLine := AddedBlanks;
        BlksAddedByThisStmt := AddedBlanks;
        Bunch(StatmtPart[1], StatmtPart[2], CharCount, SymbolGap);
        if Successful
        then
          Bunch(StatmtBeginning, StatmtPart[1], CharCount,
             StatmtSeparation)
        else IfThenBunchNeeded := True;
        if SymbolName = ElseSymbol
        then
          begin
            Pack(Symbol, 1, StatmtSymbol[1]);
            StmtSymLength := Length;  IfThenBunchNeeded := False;
            StartNewLineAndIndent;  StatmtPart[3] := CharCount;
            WriteSymbol;  ReadSymbol;
            StatmtPart[4] := CharCount + 1;
            DoStatement(AddedBlanks, StatmtSymbol, StmtSymLength);
            BlksOnCurrntLine := AddedBlanks;
            BlksAddedByThisStmt := BlksAddedByThisStmt +
               AddedBlanks;
            Bunch(StatmtPart[3], StatmtPart[4], CharCount,
               SymbolGap);
            BlksOnCurrntLine := BlksAddedByThisStmt;
            if Successful then
              Bunch(StatmtBeginning, StatmtPart[3], CharCount,
                 StatmtSeparation);
          end
        else
          if (CharCount - StatmtBeginning) < BufferSize
          then
            begin
              BunchWanted := not BunchWanted;
              BlksOnCurrntLine := 0;
              Bunch(StatmtBeginning, StatmtPart[1], StatmtPart[2],
                 SymbolGap);
              BunchWanted := not BunchWanted;
            end;
        IfThenBunchNeeded := False;
      end {IfStatement};
    CaseStatement:
      begin
        repeat WriteSymbol;  ReadSymbol
        until SymbolName = OfSymbol;
        WriteSymbol;  ReadSymbol;
        ChangeMarginTo(ActualLeftMargin + IndentIndex);
        while SymbolName <> EndSymbol do
          begin
```

```
              StartNewLineAndIndent;  StatmtPart[1] := CharCount;
              for I := 0 to (Length - 1) div AlfaLeng do
                Pack(Symbol, (I * AlfaLeng + 1), StatmtSymbol[I + 1]);
              StmtSymLength := Length;
              repeat WriteSymbol;  ReadSymbol
              until SymbolName = ColonSymbol;
              WriteSymbol;  ReadSymbol;
              if not (SymbolName in [Semicolon, EndSymbol])
              then
                begin
                  StatmtPart[2] := CharCount + 1;
                  DoStatement(AddedBlanks, StatmtSymbol,
                     StmtSymLength);
                  BlksOnCurrntLine := AddedBlanks;
                  BlksAddedByThisStmt := BlksAddedByThisStmt +
                     AddedBlanks;
                  Bunch(StatmtPart[1], StatmtPart[2], CharCount,
                     SymbolGap);
                end {IF NOT(SymbolName...)};
              if SymbolName = Semicolon
              then begin WriteSymbol;  ReadSymbol; end;
            end;
          ChangeMarginTo(ActualLeftMargin - IndentIndex);
          StartNewLineAndIndent;  WriteSymbol;  ReadSymbol;
          if EndCommentsWanted and (LastSymbol = EndSymbol) then
            begin
              StatmtSymbol[1] := 'CASE      ';  StmtSymLength := 4;
              WriteComment;
            end;
        end {CaseStatement};
      OtherStatement:
        begin
          while not (SymbolName in [Semicolon, UntilSymbol, EndSymbol,
             ElseSymbol]) do
            begin WriteSymbol;  ReadSymbol end;
        end {OTHER};
      CompoundStatement: DoStmtList(EndSymbol)
    end {CASE};
    AddedBlanks := BlksAddedByThisStmt;
    ChangeMarginTo(ActualLeftMargin - IndentIndex);
  end {DoStatement};


begin {DoBlock}
  LastProgPartWasBody := LastProgPartWasBody and (SymbolName =
     BeginSymbol);
  if SymbolName = LabelSymbol  then DoDeclarationUntil(EndLabel);
  if SymbolName = ConstSymbol  then DoDeclarationUntil(EndConst);
  if SymbolName = TypeSymbol  then DoDeclarationUntil(EndType);
  if SymbolName = VarSymbol  then DoDeclarationUntil(EndVar);
  while SymbolName in [FuncSymbol, ProcSymbol] do DoProcedures;
  if SymbolName = BeginSymbol
  then
    begin
      if LastProgPartWasBody
      then for I := 2 to ProcSeparation do StartNewLineAndIndent;
      IfThenBunchNeeded := False;  AtProcBeginning := True;
      ChangeMarginTo(ActualLeftMargin - IndentIndex);
      DoStatement(I, BlockName, BlockNmLength) { I IS DUMMY PARAM };
      LastProgPartWasBody := True;
      ChangeMarginTo(ActualLeftMargin + IndentIndex);
    end
  else begin WriteSymbol;  ReadSymbol {Write FORWARD} end
end {DoBlock};


procedure Initialize;

  var
                       I: Width;

  begin { CONSTANTS:  }
    Digits := ['0' .. '9'];
    LettersAndDigits := ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
       'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
       'W', 'X', 'Y', 'Z'] + Digits;
    AlphaSymbols := [ProgSymbol, BeginSymbol, EndSymbol, ConstSymbol,
       TypeSymbol, RecordSymbol, CaseSymbol, IfSymbol, ThenSymbol,
       ElseSymbol, DoSymbol, OfSymbol, ForSymbol, WithSymbol,
       WhileSymbol, RepeatSymbol, UntilSymbol, Identifier, VarSymbol,
       ProcSymbol, FuncSymbol, LabelSymbol, AlphaOperator];
    EndLabel := [ConstSymbol, TypeSymbol, VarSymbol, ProcSymbol,
       FuncSymbol, BeginSymbol];
    EndConst := EndLabel - [ConstSymbol];
    EndType := EndConst - [TypeSymbol];
    EndVar := EndType - [VarSymbol];
{ Initialize COLUMN DATA:  }
    WriteColumn := 0;  LeftMargin := 0;  ActualLeftMargin := 0;
    OutputCol := 1;  ReadLeftCol := 1;  ReadRightCol := MaxReadRightCol;
    WriteLeftCol := 1;  WriteRightCol := MaxWriteRightCol;  Oldest := 1;
    CharCount := 1;  LineNumber := 0;  Increment := 0;
{ Initialize Boolean PARAMETERS:  }
    PackerIsOff := True;  BunchWanted := False;  DisplayIsOn := True;
    ProcNamesWanted := True;  EndCommentsWanted := False;
    NoFormatting := False;
{ Initialize NUMERIC PARAMETERS:  }
    IndentIndex := 3;  LongLineIndent := 3;  ProcSeparation := 2;
    SymbolGap := 1;  StatmtSeparation := 3;  DeclarAlignment := 0;
{ Initialize INPUT CONTEXT DATA:  }
    ReadColumn := 1;  ChIsEOL := False;  NextChIsEOL := False;
    for I := 0 to BufferSize do Symbol[I] := ' ';
    LastSymbol := PeriodSymbol;  LastProgPartWasBody := False;
  end {Initialize};


begin {MainProgram}
  ConstantsInitialization;  Initialize;
  if EOF(Input)  then WriteLn(' *** NO PROGRAM FOUND TO FORMAT.')
  else
    begin
      ReadACharacter;  ReadSymbol;
      if SymbolName <> ProgSymbol
      then WriteLn(' *** "PROGRAM" EXPECTED.')
      else
        begin
{
      ******************************************************************
      *                                                                *
      *            F O R M A T    T H E    P R O G R A M               *
      *            - - - - - -    - - -    - - - - - - -               *
      *                                                                *
      ******************************************************************
}
          StartNewLineAndIndent;
          WriteSymbol;  ReadSymbol;
          for I := 0 to (Length - 1) div AlfaLeng do
            Pack(Symbol, (I * AlfaLeng + 1), Main[I + 1]);
          MainNmLength := Length;
          repeat WriteSymbol;  ReadSymbol;  until SymbolName = Semicolon;
          WriteSymbol;  ReadSymbol;  StartNewLineAndIndent;
          DoBlock(Main, MainNmLength);  WriteA('.');
          FlushUnwrittenBuffer;
        end;
    end;
  13:
end {MainProgram}.
```

MOVING A LARGE PASCAL PROGRAM FROM AN LSI-11 TO A CRAY-1

Richard L. Sites, APIS Department, UC/San Diego 92093

In March, 1978, I had occasion to move a 2400-line PASCAL program from an LSI-11 at the University of California/San Diego (UCSD) to a Cray-1 at Los Alamos Scientific Laboratory (LASL). At both places, the compiler is a variant of the P4 portable compiler. This note summarizes the experience and makes several major points about PASCAL:

1. It was possible to move a substantial PASCAL program from a small slow machine to one approximately 150 times bigger. No other language has compatible full-language implementation across such a wide range of machines--essentially from the world's slowest micro to the world's fastest supercomputer.

2. There were compile-time and run-time incompatibilities which should not have existed. The last part of this note is directed to implementors, with a plea to avoid such problems.

3. Using a table-top LSI-11 system, an on-going project is developing production software for the Cray-1. This would not be feasible in BASIC, FORTRAN, or assembly language.

Before describing the problems encountered in moving the program, a little background is needed. The P4 portable PASCAL compiler is about 4000 lines of PASCAL source, and translates from PASCAL to an intermediate language called P-CODE. P-CODE is the machine language for a pseudo-machine that has a simple stack and about 50 operations. The P-CODE version of a program consists exclusively of a stream of these simple operations, with no associated side tables or assumed information.

On the Cray-1, P-CODE is translated by another 4000 line PASCAL program into Cray-1 assembly language, which then cascades into the standard assembler and loader. This sequence allowed a running, reasonably efficient PASCAL system to be brought up on the Cray-1 with very few months of effort.

On the LSI-11, P-CODE is represented in a very compact form, and is interpreted directly. This has two advantages over compiling to native PDP-11 machine code: First, the P-CODE form of a program is more compact than the machine code, typically by a factor of two. This space compactness is the sole reason that the compiler is able to compile itself in a 56K-byte memory. Second, by changing only the interpreter, the identical P-CODE can be run on other micros, allowing the entire compiler and operating system to be transported to other machines.

The program moved from UCSD to LASL is the skeleton of a machine-independent optimizer for P-CODE. The initial version of the optimizer will work on Cray-1 P-CODE, but later versions should work on other variants, and hence one set of optimizing algorithms may eventually be running on a wide variety of machines. The characteristics of the two machines and the initial 2400 line skeleton are summarized in Table 1.

Source program moved	
Pascal source lines	2400
Pascal procedures	69
P-CODE instructions (Cray-1)	9200
Cray-1 instructions	19100

Compile times in seconds	LSI-11	Cray-1	Approximate ratio
Pascal to P-CODE (lines/min.)	245 (600)	1.19 (120000)	200 : 1
P-CODE to Cray-1 asm	-	2.30	
Cray-1 asm to binary	-	4.62	
Cray-1 loader	-	0.66	
TOTALS	245	8.77 (16400)	30 : 1
Execution times in seconds			
75 data lines	51	0.32	150 : 1
2400 data lines	n.a.	3.43	
Memory sizes in bytes	56K	8000K	150 : 1

Table 1. Summary of source program moved and machines used.

The rest of this note describes the six major portability problems encountered, along with my suggestions for solutions. Some of these comments parallel those found in other articles on these pages over the past few years. The entire process of bringing the program up on the Cray-1 took 1-1/2 days, although I originally expected it to take 1/2 a day. The extra time was wasted on the problems below.

Portability problem #1

The Cray-1 compiler recognizes only lower-case ASCII reserved words, while the UCSD compiler recognizes only upper-case ASCII. This meant that the first compilation died immediately, looking for the word "program".

This problem clearly subverts the essential idea of ASCII as a standard Code for Information Exchange. It is not sufficient just to have the compiler convert all input to a single case, because (1) character string constants must not be changed, and (2) ALIAS SPELLINGS of identifiers should not be allowed. An alias spelling is defined to be one that may or may not be recognized as the same as an original spelling, depending on the details of a particular compiler implementation. In our current context, a variable declared as:

```
VAR XYZ : INTEGER;
```

could have alias spellings of "xyz" and "Xyz", among others. As a matter of principle, I believe that such spellings should not be allowed because they serve only to introduce confusion about whether the original programmer intended three distinct variables or one. The standard example program for this issue is:

```
BEGIN
VAR XYZ : INTEGER;
PROCEDURE ABC;
   VAR xyz : REAL;
   BEGIN
   XYZ := 12; (* which block,  inter or outer ??*)
   END;
...
```

Converting all identifiers to upper case resolves the assignment to the REAL variable, while treating the case shifted names as distinct resolves the assignment to the INTEGER variable. I believe that the original programmer's intent in such a program is truly ambiguous, so the program should not be allowed in the first place. The declaration xyz : REAL should generate a compile-time error (or at least be flagged with a warning) on the basis that an alias spelling of the same variable already exists. Thus, the issue of how to resolve the assignment never comes up. In quick summary, my proposed portable upper- lower-case rules are:

(1) Reserved words, such as BEGIN are recognized independent of the case of the individual letters, so that "BEGIN", "begin", and "BeGiN" are all recognized as reserved words.

(2) An identifier used in a declaration may have its individual letters in any case, and that particular spelling is inserted into the symbol table, SO LONG AS NO ALIAS SPELLING ALREADY EXISTS in the symbol table.

(3) An identifier used in the body of a program must exactly match the spelling in the symbol table, including each letter being of the correct case.

These rules allow any program to be compiled, so long as words in it are consistently spelled with the same pattern of upper- and lower-case letters.

(ASIDE: These same rules can be used to detect most cases of identifiers which differ after the first eight letters, without needing to store more than eight letters in the symbol table. Most compiler symbol tables store 7-bit ASCII characters in 8-bit bytes. If all the characters after the first 8 in an identifier are hashed and the hash value stored in the unused bits of these bytes, then rule (2) above can be interpreted to mean "an alias spelling exists (and hence an error/warning message is generated) if some existing identifier in the symbol table has the same first 8 characters, but a different hash code for the remaining characters." Rule (3) above can be interpreted to mean "the first 8 characters of an identifier and the hash code for the remaining characters must match exactly." This idea completely clears up the concern of A.H.J. Sales (Pascal News, Feb. 1978, p. 78), except when the hash codes for two different tails turn out to be identical; this can be made rare, and can be guaranteed not to happen for single-character differences. End of ASIDE.)

Portability problem #2

Contrary to the Report, the Cray-1 compiler does not recognize empty field lists in variant record declarations, RECORD CASE I:BOOLEAN OF TRUE:(X,Y: INTEGER); FALSE:()END , nor does it recognize untagged variant records, RECORD CASE BOOLEAN OF ... . The lesson here is clear--recognize the entire language as defined, without taking shortcuts.

Portability problem #3

UCSD Pascal includes non-standard procedures OPEN and CLOSE. I had to rework the calls to use the standard RESET and REWRITE, which lack two useful capabilities: (1) there is no way to close a file explicitly, and hence there is no way to release a file for other uses before the program terminates; there also is no way to specify whether the file is to be disposed of (a temporary disk file) or kept (a disk output file) after termination; (2) there is no way to open a file explicitly, supplying a character-string file name at that time. These are limitations I can live with, but I would prefer to see some agreed-upon standard extensions in this area.

Portability problem #4

Type checking was inconsistent. UCSD Pascal accepted

```
TYPE WHOLENUM = 0..32767;
VAR I          :INTEGER;
FUNCTION F(...):WHOLENUM;
   ...
I := F(...)*I;
```

while the Cray-1 compiler complained about operand incompatability at the multiply. Inconsistent type checking is a well-known problem in Pascal, so I won't dwell on it. In this particular case, though, I am frustrated because the whole purpose of introducing the type WHOLENUM is to convey to the reader (and the compiler) the idea that all WHOLENUMs are intended to be non-negative. My temporary fix was : TYPE WHOLENUM = INTEGER;

At this point in the process, my 2400 line program compiled properly and executed for the first time. I had fixed problem #1 by converting the entire program to lower case, and this fix now came back to haunt me, because the input data file was still in upper case, and hence did not match any of my lower-case character-string constants. Converting the entire data file to lower case also did not quite do the trick, because my program's output (remember, the 2400 line program optimizes P-CODE) cascades eventually into the Cray-1 loader, which demands standard procedure names (such as SIN) in upper case. Clearly, the case shift problem was taking more energy than it should.

Portability problem #5

The first real problem to crop up in execution was that my hash function always returned the same value, zero, instead of reasonably well distributed series of numbers in the range 0..127. The hash function was built using (conceptual) shifts and exclusive-or's, and in fact did a fair amount of lying with variant records to jump between character, integer (I+I used for left shift of one bit), and set (S1+S2, S1*S2, and ALLBITS-S used to build XOR) representations. There is a serious issue here of how to build a portable hash function. The problem will be even harder if strong typing advocates' remove variant records as an "escape hatch". Try it yourself -- build a function which accepts a PACKED ARRAY [0..7] OF CHAR and returns an integer in the range 0..127. The particular hash function desired XORs the 8 characters, each one offset one bit from the next to get a 15-bit intermediate. The upper 8 bits and the lower 7 are then XORed, and the lower 7 bits of this are returned. (This particular function guarantees different hash values for inputs which differ by any one character, or which differ by a transposition. In addition, no overflow is generated on a 16-bit machine.)

Portability problem #6

The final output of the program was spaced funny. In the statement

```
WRITELN(3,4);
```

UCSD Pascal inserts no blanks around the fields, giving "34", while Cray-1 Pascal uses a default field width, giving "   3   4". In my application, the blanks are not wanted, but Pascal output editing is not precisely defined, so many implementations supply extra blanks. Often, these blanks reflect a legitimate desire to separate items of output when specified field widths are exceeded, as in WRITELN(100:2, 200:2) which normally prints as "100 200". I propose that a specified field width of zero mean no padding blanks, and that the exact details of output editing be specified somewhere.

Overall, moving a 2400-line Pascal program proved surprisingly successful, and having done it once should make it easy to move a 5000-line program this summer.

(* Received 78/05/01 *)

On the Article "What to do After a While"

Roy A. Wilsker - Mass. State College Computer Network

INTRODUCTION

The letter by A.H.J. Sale[1] and the article by Barron and Mullins[2] in PASCAL News #11 address themselves to an ambiguity in the definition of the PASCAL language: should Boolean expressions be evaluated in a parallel or sequential manner?

For example, when we write "P and Q", do we mean

1) (parallel or "logical" evaluation)

$$P \wedge Q \quad (= Q \wedge P)$$

or

2) (sequential evaluation)

```
if P then Q else false
```

I argue here for the parallel approach.

THE PROBLEM OF PSYCHOLOGICAL SET

My first objection to sequential evaluation is that it **looks** parallel to anyone who has had any exposure to symbolic logic. This is the problem of "psychological set", first discussed by Gerald M. Weinberg[3]. This term connotes a state in which our way of thinking about a situation blinds us to its reality. For example, a common error encountered by programmers who use languages in which variables need not be declared is the use of misspelled variable names which "look like" other (valid) names. This kind of error can be extremely hard to find.

Thus, a maintenance programmer who runs into the expression

```
while (i <= maxsize) and (a[i] <> item) do
```

and later finds

```
while (a[i] <> item) and (i <= maxsize) do
```

may not even see them as different expressions!

PROVING PROGRAMS CORRECT

The advent of structured control statements has generated a great deal of interest in the problem of proving, either by hand or automatically, the correctness of programs[4]. My second objection to symbolic evaluation is that it will probably increase the difficulty of doing such verification by an order of magnitude. This belief is based on the fact that, in abstract mathematics and logic, non-commutative (i.e., order-dependent) objects are much harder to handle than commutative objects.

ON "THE SPIRIT OF PASCAL"

Barron and Mullins argue that sequential evaluation allows us to program "more in the spirit of PASCAL". Whatever that patriotic remark means, I strongly disagree. Let's look at the example they give. We are to search a table for a given item. Using sequential evaluation, their solution is:

```
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" *)
```

There are two fundamental flaws in this solution:

1) The solution twists the algorithm to fit a given data structure.

One of the great advantages of PASCAL over most other languages is the ability it gives the user to create data structures which work well with a given problem. Consider the following solution to the table search problem[5]:

```
var table : array [0..maxsize] of whatever;
.
.
table[0] := item; (* put in sentinel for end of search *)
index := maxsize;
while table[index] <> item
   do index := index - 1;
(* condition for item not found is "index = 0" *)
```

2) The repetitive construct mixes together logical and iterative repetition.

Indeed, in Algol 68[6], a cleaner way to write the Barron-Mullins algorithm would be:

```
index := 1;
for i from 1 to maxsize while table[i] <> item
   do index := i + 1 od;
# condition for item not found is "index > maxsize" #
```

The problem with this technique is that on exiting the loop, one does not know if termination was caused by the count being exceeded or by the logical condition failing. This is a common error-causing situation, better known as "exiting a loop to the same place from the side and the bottom"[7].

ON "EFFICIENT" ALGORITHMS

Finally, I would like to take a moment to talk about efficiency. Barron and Mullins say:

> "... But in the Pascal community we should have gotten beyond judging features solely in terms of implementation efficiency. What matters is being able to write correct programs that are easily comprehensible."

The answer to the question of efficiency is not so simple. True, the first consideration of the designer should always be the correctness and clarity of the design. But efficiency often comes in a close second, and sometimes it's a dead heat: in certain circumstances (e.g., real time applications or CAI) if the program is not efficient enough, in terms of either size or execution time, it is irrelevant as to whether or not it's clear, or even correct - the program is unusable.

As Donald Knuth[8] and others[9] have pointed out, the problem is generally not that the designer has made efficiency a consideration, but how he has tried to make the design efficient. The villain is not efficiency itself, but micro and premature optimization.

In fact, the algorithm given in this article is a good example of how to optimize a program: by improving its data structures and algorithms. In a Ratfor preprocessor written in PASCAL, the substitution of the algorithm given above for the original one (which was essentially the Barron-Mullins algorithm) resulted in a 30% decrease in the preprocessor's execution time with no impairment of the clarity of the program.

REFERENCES

1. PASCAL News #11, p. 76-78.
2. PASCAL News #11, p. 48-50.
3. Weinberg, Gerald M. The Psychology of Computer Programming. Van Nostrand-Reinhold, 1970.
4. See, for example,
   Dijkstra, E.J. A Discipline of Programming. Prentice-Hall, Englewood Cliffs, 1976.
   Good, D.I., "Towards a Man-Machine System for Proving Program Correctness", Report TSN-11, The University of Texas at Austin, Computation Center, June 1970.
   Hoare, C.A.R., "An Axiomatic Basis for Computer Programming", Comm. ACM, vol. 12, no. 10, October 1969, p. 567-580, 183.
   Marmier, E., "A Program Verifier for PASCAL", IFIP Congress 1974.
   Naur, P., "Proof of Algorithms by General Snapshots", BIT, vol. 6, no. 4, 1966, p. 310-316.
5. This solution to the table search problem is taken from the excellent article:
   Knuth, Donald E., "Structured Programming with GOTO Statements" in Current Trends in Programming Methodology: Volume I, Software Specification and Design, ed. Raymond T. Yeh. Prentice-Hall, Englewood Cliffs, 1977.
6. See, for example,
   Pagan, Frank G. A Practical Guide to Algol 68. John Wiley & Sons, 1976.
7. Kernighan, B.W., and Plauger, P.J. The Elements of Programming Style. McGraw-Hill, 1974.
8. Knuth, Donald E., "An Empirical Study of FORTRAN Programs", Software - Practice and Experience, vol. I, no. 2 (April-June, 1971), p. 105-133.
   (See also the work cited in reference 5.)
9. See, for example, the work cited in reference 7, and
   Yourdan, Edward. Techniques of Program Structure and Design. Prentice-Hall, 1975.

(* Received 78/05/11 *)

MASSACHUSETTS STATE COLLEGE COMPUTER NETWORK

A RESOLUTION OF THE BOOLEAN EXPRESSION-EVALUATION QUESTION

or

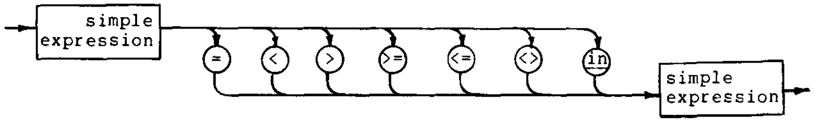
IF NOT PARTIAL EVALUATION THEN CONDITIONAL EXPRESSIONS

Morris W. Roberts
Robert N. Macdonald
Department of Information Systems
Georgia State University
Atlanta, Georgia 30303

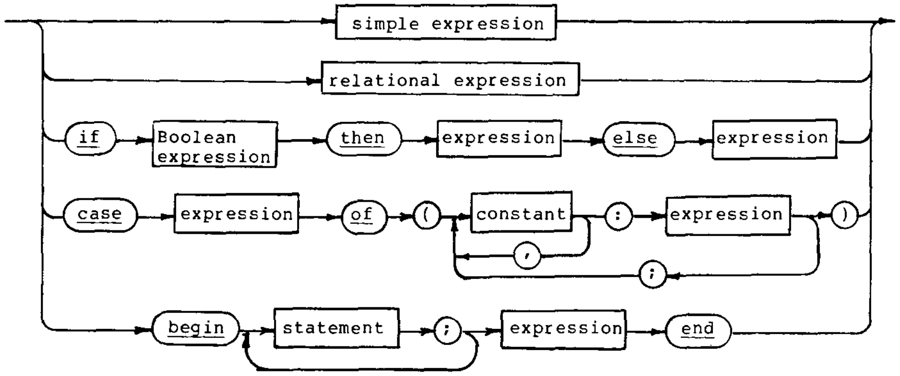
Introduction

The programming languages ALGOL-60[1] and ALGOL-W[2], which contain the precursors of many of the elegant features of PASCAL, are richer than PASCAL in the variety of ways that an expression may be formed. Both ALGOL-60 and ALGOL-W contain the conditional expression and ALGOL-W contains, in addition, the case expression and the value block. A "PASCALized" summary of these constructs is shown in the syntax diagrams below.

relational expression



expression



Although the effects of the conditional expression, case expression, and value block may be had in PASCAL (or in FORTRAN, for that matter), the resulting constructs require multiple statements and the declaration and use of temporary variables that are not otherwise needed if these forms of expression are used. We recommend the incorporation of these forms of expression into PASCAL on the following grounds:

1. The increased programming facility that they offer more than compensates for the increased syntatic complexity which their adoption would entail.
2. The conditional expression, in particular, promotes rigor by removing semantic ambiguities that exist in the evaluation of Boolean expressions.
3. None of these extensions conflicts with the PASCAL design goals cited by Vavra[3].

We shall restrict the scope of this paper to the case for the conditional expression, showing first some examples of its use, and second the way in which it avoids the current arguments concerning the proper evaluation of Boolean expressions.

The Conditional Expression

In BNF notation, the <conditional expression> may be defined to be

```
if <Boolean expression> then <expression>
    else <expression>.
```

(ALGOL-60 restricts the <expression> following the then to be a <simple expression>.)

This construct permits such statements as:

```
a.  x := (if n<100 then a+b else a-b) * (c+d);

b.  a := sqrt(b[if i in s then i else 0]);

c.  while if x<=10 then a[x]<>b else false do x:=x+1;

d.  append := if null(x) then copy(y)
      else if x^.class=list then
        cons(copy(x^.car),append(x^.cdr,y))
      else referenceerror('append: 1st arg invalid structure');
```

Expressing these constructs in PASCAL is straightforward. Example c could be written as

```
found := false;
while (x<=10) and not found do
  if a[x]=b then found:=true else x:=x+1;
```

Thus, the PASCAL version of c requires two statements and the extra Boolean variable, "found".

Resolution of a Semantic Problem

Another possibility is that the previous example could be written as

```
while (x<=10) and (a[x]<>b) do x:=x+1;
```

provided the evaluation of the Boolean expression is terminated as soon as x<=10 becomes false. This avoids errors when a[11] does not exist or when it is undefined. This approach is currently the subject of some debate. In two recent articles in Pascal News, the authors Sale[4] and Barron and Mullins[5] have taken opposite positions regarding standards for the evaluation of Boolean expressions. Sale recommends the "Boolean operator" approach which forces full evaluation of the complete expression, whereas Barron and Mullins prefer "sequential conjunction" which permits the compiler to terminate evaluation of an expression as soon as its truth or falsity is unequivocally determined.

The reasons that have been given for partial evaluation seem to be:
1. efficiency
2. the resolution of cases in which one or more of the terms and factors of the expression are undefined.

Although Barron and Mullins have described three syntactically correct ways of avoiding the problem of point 2 by segmenting the expression, still they advocate the use of partial evaluation. Their position is understandable, for the techniques required are contrived. Unfortunately, they are the only reasonable ones available with the present language, and partial evaluation makes the code appear to be simpler.

The User Manual[6], as noted by Sale, interprets the Report[6] neither to require nor to forbid the full evaluation of Boolean expressions. However, the syntax of the <expression> given in the Report clearly implies that all operators are to be applied in the evaluation of an expression. Thus, it seems reasonable to expect that any action which appears explicitly in the flow of control must be evaluated. If this is not the case there will always be an uncertainty as to what portions of the program have been executed. For example, the statement

```
while A and B do ... ;
```

means that the statement following do is to be executed if A and B are both true. According to Barron and Mullins, this would be reinterpreted to mean "don't evaluate B and don't execute if A is false." There is a subtle difference between these two notions. The difference is important because B might involve a Boolean function which performs necessary operations on global variables or var parameters.

Full evaluation of the expression is in keeping with the syntax described in the Report and with intuition. From the language-design standpoint, there seems to be no justification for performing a partial evaluation of an expression. This is particularly true since the most compelling reason advanced for the partial evaluation is to avoid an awkward temporary variable. The conditional expression is a complete solution to this specific problem in that it permits the selection, by the programmer, of the terms and/or factors that are to be evaluated.

Example c, above, solves problem 2 by explicitly directing the flow of control around impossible cases. It does not depend on implicit conventions of partially evaluating expressions.

In our opinion, the only reason for not fully evaluating an expression is efficiency of time and memory utilization. While the use of partial evaluation does have an advantage over the standard PASCAL construction, the advantage is insignificant when it is compared against the conditional-expression approach. The following shows the code segments that might be generated for example c, above, if the target machine were a PDP-11.

Conditional Expression

```
1$:     evaluate the condition
        put result of x<=10 on stack

     CMP    (SP)+,#TRUE
     BEQ    2$

        evaluate the else expression
        put false on the stack

     BR     3$

2$:     evaluate the then expression
        put a[x]<>b on stack

3$:  CMP    (SP)+,#TRUE
     BNE    4$

        perform do

     BR     1$
4$:
```

Partial Evaluation

```
1$:     evaluate expression
        put x<=10 on stack

     CMP    (SP)+,#TRUE
     BNE    4$

        evaluate expression
        put a[x]<>b on stack

     CMP    (SP)+,#TRUE
     BNE 4$

        perform do

     BR     1$
4$:
```

We see from these examples that the space advantage gained from partial evaluation is that for evaluating the else part of the if and a branch instruction. In this case it amounts to two instructions. It is more interesting to note that the execution time advantage is zero as long as the else condition is not evaluated. We feel that there is insufficient justification for adopting partial evaluation as a standard feature of the language. It might, however, be a desirable implementation-dependent feature activated by a compiler directive.

Conclusions

PASCAL is not yet a complete language in that inclusion of several desirable features of other languages has not yet been openly debated. We recommend the case expression, the value block, and particularly the conditional expression as additions to the language. The basis for this recommendation is that these features will promote semantic rigor, will not conflict with any language-design goals, will provide the programmer with new and useful tools, and will improve the efficiency of the generated code over standard PASCAL.

References

1. Naur, Peter (Editor), et al., "Revised Report on the Algorithmic Language ALGOL 60," Communications of the ACM 6 (January, 1963), 1-17.

2. Wirth, Niklaus and C. A. R. Hoare, "A Contribution to the Development of ALGOL," Communications of the ACM 9 (June, 1966), 413-433.

3. Vavra, Robert D., "What Are Pascal's Design Goals?", Pascal News 12 (June, 1978), 33-35.

4. Sale, Arthur H. J., "Compiling Boolean Expressions," Pascal News 11 (February, 1978) 76-78.

5. Barron, D. W. and J. M. Mullins, "What to do After a While," Pascal News, 11 (February, 1978) 43-50.

6. Jensen, Kathleen and Niklaus Wirth, PASCAL User Manual and Report, Springer-Verlag (1975).

(* Received 78/08/07 *)

What to do after a while .. longer

Chepstow (029 12) 4850

T.M.N. Irish
5 Norse Way
CHEPSTOW
UNITED KINGDOM

Sedbury
Gwent
NP6 7BB

78 Se 20 W

References
(1.) Barron, D.W. & Mullins, J.M. WHAT TO DO AFTER A WHILE
PN 11 p. 48, 1977
(2.) Jensen, K. & Wirth, N. PASCAL USER MANUAL AND REPORT
2nd cor. reprint of the 2nd ed. Springer-Verlag, 1978
(3.) Sale, A.H.J. COMPILING BOOLEAN EXPRESSIONS —
PN 11 p. 76, 1977

I was brought up to regard B&M (1.) appendix example 1. as the normal way of searching a table. I deny that it is a distortion, and, on the contrary, claim that it bears both a simple and an obvious relationship to the problem. When I study it, I see only the falsity of their assertion.

User Manual, page 22 :- "The while statement" ... "The expression controlling the repetition must be of type Boolean. It is evaluated before each iteration, so care must be taken to keep the expression as simple as possible."

User Manual, pages 20-21, quoted by Sale (3.), beginning :- "Boolean expressions have the property ... " I take this to mean that J&W (2.) have as little sympathy for those who rely on not well-defined factors as B&M have for those who rely on side-effects of functions.

User Manual, page 12 :- "Hence, it is possible to define each of the 16 Boolean operations using the above logical and relational operators." There follows a table showing the 16 Boolean operations, in which p and q are Boolean operands, r is a Boolean variable and "." , "1" , XOR , EQV & IMP are abbreviations for false , true (except "case 1"), exclusive OR , equivalence & implication.

case 1 2 3 4
p = . . 1 1
q = . 1 . 1
	expression	equivalent
		Boolean operator
		expression	

r:=	false		
. . . 1	p and q	not (not p or not q)	
. . 1 .	p > q	p and not q	
. . 1 1	p		
. 1 . .	p < q	not p and q	
. 1 . 1	q		
. 1 1 .	p <> q	not p and q or p and not q	XOR
. 1 1 1	p or q	not (not p and not q)	
1 . . .	not (p or q)	not p and not q	
1 . . 1	p = q	p and q or not p and not q	EQV
1 . 1 .	not q		
1 . 1 1	p >= q	p or not q	IMP
1 1 . .	not p		
1 1 . 1	p <= q	not p or q	IMP
1 1 1 .	not (p and q)	not p or not q	
1 1 1 1	true		

If we were starting the language design again and
we wanted to include a facility for telling implementations how to evaluate expressions (though, in view of B&M's own remark about architectures, that seems of dubious value),
so that we were looking for "sequential conjunction" versions of
and > < or >= <=
then I, for one, would oppose the use of "and" & "or" themselves, on the ground of their old and strong Boolean algebra connotation.
If POP-2 and RTL/2 have already "adopted" (it should be "adapted") them for such a purpose, that is their problem.

Not that I care, but B&M's function andop seems unnecessarily complicated to me.

```
        function  andop ( p , q : Boolean ) : Boolean ;
        begin     andop := p and q  end ;
```

is sufficient — because the arguments are both evaluated when the function is called — surely ?

What the spirit of Pascal says to me is that we ought not to
(i) write programs that rely on not well-defined factors side-effects of functions or undefined values,
(ii) depend on implementors to let us get away with them,
(iii) tell implementors to let us get away with them,
or (iv) complain if implementors use any means they can devise to prevent us getting away with them.

The spirit of Pascal also says that it rather fancies itself as a two-edged sword !

78 Se 20 W Michael Irish

+ - * / := . , ; : ' = <> < <= > >= ( ) [ ] ↑ .. (* *)

copies to :- D.W.Barron J.M.Bishop K.Jensen G.H.Richmond
A.H.J.Sale N.Wirth

(* Received 78/09/26 *)

KNOW THE STATE YOU ARE IN

Laurence V. Atkinson
University of Sheffield
England

In a nutshell

A number of recent articles have highlighted problems with multiple exit loops in Pascal. Many of these problems disappear when a loop is controlled by a user-defined scalar.

Introduction

Multiple exit loops and problems with their implementation have featured prominently in four recent articles: Barron and Mullins [2], Bishop [3], Bishop [4] and Horton [5]. Many of these problems do not occur if user-defined scalars are introduced as 'state indicators' to control the loop. A multiple exit loop constitutes a multi-state process. Pascal's ordinal types provide a natural means of identifying multiple states. This state transition approach is introduced by first considering the Barron and Mullins paper [2] and then taking the other articles in turn.

Barron and Mullins

Their example is linear search for a specified item within a vector (assumed full) but considering the possibility that the desired item may be absent. The program they produce is

```
const maxsize = ...;  succmaxsize = ...;
var  table : array [1..maxsize] of whatever;
     index : 1..succmaxsize;
 . . .
index := 1;
while (index <= maxsize) and (table [index] <> item)
   do index := index + 1;
if index > maxsize then {item absent} . . . else {item found} . . .
```

Barron and Mullins claim that "this is a natural way of expressing the operation to be carried out" and is inkeeping with "the spirit of Pascal". The point of their paper is that this program is viable only if boolean expressions are evaluated by sequential conjunction on a strict left-to-right basis. The Report [6] leaves this issue open but the User Manual [6] states that all operands in a boolean expression will be evaluated. Jensen and Wirth [6] (Chapter 10) produce an equivalent example to illustrate the problem. A state transition approach to their solutions is presented by the present author in [1].

I suggest that a programming style both more natural and more in the spirit of Pascal is achieved when user-defined scalars are introduced and used as state indicators.

State Indicators

In a simple search environment there are three distinct states of interest:

(i) I haven't found it yet but I'm still looking,
(ii) got it,
(iii) I've looked everywhere but it's not here.

This leads us to a solution using a three-state scalar (figure 1).

```
const endoftable = ...;
type toendoftable = 1 .. endoftable;
     searchstates = (searching, thingabsent, thingfound);
 . . .
var item : array [toendoftable] of things;
    here : toendoftable;
    outcome : searchstates;
 . . .
here := 1;  outcome := searching;

repeat
   if item [here] = thingwanted then outcome := thingfound else
      if here = endoftable then outcome := thingabsent else
         here := succ (here)
until outcome <> searching;

case outcome of
   thingfound  :. . . ;
   thingabsent  :. . .
end {case}
```

Figure 1. Linear search with state transition

We now comment upon the program of figure 1.

(i) The intent of the program is more readily apparent.
(ii) The program is now more easily extended to include other cases of interest (eg. figure 2).
(iii) Subsequent processing, upon exit from the loop, is more transparent:- determination of whether or not the desired item has been located is cleaner.
(iv) The compound boolean expression has disappeared and so issues of 'boolean operator' or 'sequential conjunction' approach are avoided.
(v) The order of making the tests is not implementation dependent:- the desired order is unambiguously expressed.
(vi) The subscript cannot go out of the bounds of the array. In Barron and Mullins' version the range of the subscript must be one greater than the index range of the array. We return to this point when discussing Bishop's paper [3].
(vii) No redundant tests are made. Barron and Mullins incur a test (index <= maxsize) which is always true upon entry to the while loop. My objection to redundant testing is based on considerations of logic rather than efficiency.
(viii) Efficient implementation of the loop termination test is possible (jump on zero).

```
 . . .
searchstates = (searching, absent, foundinfirsthalf, foundinsecondhalf);
 . . .
repeat
   if item [here] = thingwanted then
      case here <= (endoftable div 2) of
         true  : outcome := foundinfirsthalf;
         false : outcome := foundinsecondhalf
      end {case}
   else . . .
until outcome <> searching;

case outcome of
   foundinfirsthalf  : . . . ;
   foundinsecondhalf : . . . ;
   absent  : . . .
end  {case}
```

Figure 2. Extended linear search with state transition.

Bishop

Judy Bishop [3] addresses the general problem of subrange exhaustion in a loop of the form

```
i := min;
while (i <= max) and condition do
begin
  { something }
  i := succ (i)
end
```

in conjuction with

```
type index = min .. max;
var  i : index;
```

When considering Barron and Mullins it was noted that one natural consequence of the state transition approach was that the subrange variable could not exceed its bounds. The present problem is therefore solved by this same approach and as before, produces a more transparent program (figure 3). The point raised by John Strait, and discussed by Judy Bishop in [4] is also covered by this approach.

Horton

Mark Horton [5] considers two examples each involving a double exit loop and uses them as a basis for suggesting a modification to the Pascal language. He encourages the use of a deterministic loop which, without any indication of the fact at the loop control level,can jump completely out of itself and far away. I do not claim that state indicators can remove the need for all gotos but they can provide a pleasing solution to both Horton's examples. We consider them in turn.

```
type index = min .. max;
var i : index;
    state : (looping, rangeexhausted, otherexitcondition);
 . . .
i := min; state := looping;

repeat
   {something}
   if i = max then state := rangeexhausted else
      if ... then state := otherexitcondition else
           i := succ (i)
until state <> looping;

case state of
   rangeexhausted      : . . . ;
   otherexitcondition : . . .
end {case}
```

Figure 3. Bishop's loop with state transition.

1. Binary search

Horton's program is

```
const maxsize = ...;  succmaxsize = ...;
var a : array [1..maxsize] of ...;
    ℓ : 0 .. maxsize;
    u : 1 .. succmaxsize;
    found : boolean;
    . . .
ℓ := 1;  u := n;
loop while (ℓ <= u) flag found do
   mid := (ℓ+u) div 2;
   if x < a[mid] then u := mid-1 else
      if x > a[mid] then ℓ := mid+1 else
         exit found
end;
if found then ... else ...
```

which is a syntactic sugaring of the following true Pascal fragment.

```
    label 1;
      . . .
    ℓ:= 1;  u := n;  found := false;
    while ℓ <= u do
    begin
       mid := (ℓ+u) div 2;
       if x < a[mid] then u := mid-1 else
          if x > a[mid] then ℓ := mid+1 else
             begin
                found := true;  goto 1
             end
    end;
1 : if found then ... else ...
```

Again we find our familiar three-state process.

Although we should not worry unduly about minor points of efficiency we must still bear overall efficiency considerations in mind when designing an algorithm. To be most aesthetically pleasing one of the first tests a program should make in a search loop is 'is what I'm looking at what I want?'. However, for binary search, we suffer if we test for equality before we test relative magnitude. This is because, in general, we will hit elements we don't want far more often than we hit an element we do want. Consequently, for about half of our probes, we should know which pointer to move after making only one comparison. Accordingly we follow Horton's order of comparisons (figure 4). Again we comment on the new program.

(i) Program intent is more transparent.

(ii) No modification to the language is necessary to permit a clean solution.

(iii) The subscripts cannot go out of the bounds of the array. Horton's program suffers from a variant of Bishop's problem: if the sought entry falls outside the table Horton's version terminates with ℓ-u=1 (ie u = ℓ-1 or ℓ = u+1).

(iv) The new program is more easily extended to include other cases of interest. In particular we may be interested to know if we found an item on the final probe available (ie when (top = bottom) and (itemwanted = itemat [top])) or earlier (in which case itemwanted = itemat [middle]).

The computation in both programs is the same but for the extra test 'state <> stillchopping' now at the end of each iteration. This test can be implemented (by any compiler anticipating this form of loop control) as a single jump (jump on zero) so this overhead should be of little concern to us.

2. Prime numbers

Horton's program is

```
const n = ...;
var p, d : 2 .. n;
    potential_prime : boolean;
  . . .
loop for p := 2 to n flag potential_prime do
   loop for d := 2 to trunc(sqrt(p)) do
      if p mod d = 0 then next potential_prime
   end;
   write (p)
end
```

```
const endoftable = ...;
type span = 1 .. endoftable;
var itemat : array [span] of ...;
    bottom, middle, top : span;
    state : (stillchopping, found, absent);
 . . .
bottom := 1;  top := endoftable;  state := stillchopping;

repeat
   if top = bottom then
      case itemat [top] = itemwanted of
         true : state := found;
         false : state := absent
      end {case}
   else
      begin
         middle := (top + bottom) div 2;
         if itemwanted < itemat [middle]
           then top := middle-1 else
             if itemwanted > itemat [middle]
               then bottom := middle+1 else
                 state := found
      end
until state <> stillchopping;


case state of
   found  : . . . ;
   absent : . . .
end {case}
```

Figure 4. Binary search with state transition

which, without the syntactic sugar, is

```
    label 1;
    const n = ...;
    var p, d : 2 .. n;
        potentialprime : boolean;
     . . .
    potentialprime := false;
    for p := 2 to n do
    begin
       for d := 2 to trunc(sqrt(p)) do
          if p mod d = 0 then
             begin
                potentialprime := true;  goto 1
             end;
       write(p);
1 : end
```

Horton mentions that only odd numbers and divisors need be tested. In the finite state approach we still sweep through contiguous numbers (although we could avoid it - as could Horton) but this time start at 5 and test only odd divisors. Since divisors start at 3 it is sensible to make the loop deal with primes > 3 (hence >= 5). Accordingly primes <=3 are best dealt with separately. Apart from these modifications we stick to Horton's algorithm (figure 5). There should be no need to reiterate previous comments.

```
 const n = ...;
 var p, potfactor, rootofp : 2 .. n;
     state : (moredivisors, factorfound, pisprime);
  . . .
 if  n <= 3 then primesupto (n) else
 begin
    primesupto (3);
    for p := 5 to n do
      if p mod 2 <> 0 then
        begin { p is odd }
          rootofp := trunc(sqrt(p));
          potfactor := 3;  state := moredivisors;

          repeat
            if p mod potfactor = 0 then state := factorfound else
              if potfactor >= rootofp then state := pisprime else
                potfactor := potfactor + 2
          until state <> moredivisors;

          if state = pisprime then write (p)
        end { p is odd }
end
```

Figure 5. Prime numbers with state transition.

Conclusions

We have seen some illustrations of a particular style of programming. The state transition technique is applicable to a number of programming situations and to multi-exit loops in particular. I (and my students) have adopted this approach for a number of years and have rarely suffered from Barron's, Horton's or Mullins/Bishop's complaints. My response to Barron and Mullins' query "What to do after a while?" is "Know the state you are in!"

References

[1] L.V. Atkinson, "Pascal scalars as state indicators", 1978. (under review).

[2] D.W. Barron and J.M. Mullins, "What to do after a while", Pascal News, #11, 48-50, 1978.

[3] Judy M. Bishop, "Subranges and conditional loops", Pascal News #12, 37-38, 1978.

[4] Judy M. Bishop, Letter to John Strait, Pascal News, #12, p51, 1978.

[5] Mark D. Horton, Letter to the editor, Pascal News, #12, 48-50, 1978.

[6] Kathleen Jensen and Niklaus Wirth, Pascal - User Manual and Report, Springer-Verlag, 1978.

(* Received 78/09/15 *)

****

Open Forum for Members

DEPARTMENT OF DEFENSE
DEPENDENTS SCHOOLS

DARMSTADT CAREER CENTER
APO New York 09175

EUROPE

25 May 1978

SUBJECT: Pascal News

TO: Pascal User's Group
c/o Andy Mickel
University Computer Center 227 Ex
208 S.E. Union Street
University of Minnesota
Minneapolis, Minnesota 55455

Dear Andy,

Our school computer group reads with great interest the developments you have presented in Pascal News and additional papers obtained from UCSD written by Kenneth L. Bowles concerning Micro Computer Based Mass Education and the Personalized System of Instruction (PSI). Until such time when we can pilot and implement microbased systems, we would like, as a first step, to obtain a Pascal implementation for our installed equipment. We have in our overseas schools 32 Interdata 7/16's and 3 Univac 90/30's with some 200 terminals supporting BASIC. This office represents the European region and has special interest in the Interdata 7/16 implementation. (Our Pacific region operates the Univac 90/30's.) These systems are devoted to our instructional program where one of our goals is to generate computer literate youth.

Many of our graduates become employed with DOD in some capacity. If DOD, with its "Ironman" project is moving toward a Pascal based standard system, it is only natural for its own school system to move in step with the same programing language system.

We would certainly be interested in the experience of other K-12 school systems which are using Pascal in their computer education programs. Andy, you speak of jobs for Pascal people. We are a large school system with over 120,000 students. We invision a real demand for Pascal instructors (teachers) in the next few years, but we need help in getting started.

Thank you,

Samuel W. Calvin
SAMUEL W. CALVIN
Coordinator Computer Education

OUR GERMAN ADRESS

DARMSTADT CAREER CENTER
ATTN: SAM CALVIN
ESCHOLLBRUCCKERSTR. ENDE
61 DARMSTADT DEUTSCHLAND

78/6/5

JOHNSON CONTROLS
Systems & Services
Division

Johnson Controls, Inc.
507 East Michigan Street
Post Office Box 423
Milwaukee, WI 53201
Tel. 414/276 9200

Mr. Andy Mickel
University Computer Center
227 Experimental Engineering
University of Minnesota
Minneapolis, MN 55455

June 8, 1978

Dear Andy:

It was nice talking with you after having been away from the University of Minnesota for so long. As per your request, I am documenting in writing the discussion that we had, in the hope that you will be able to communicate my request to your readers.

We are currently designing a process control language for use in our Building Automation Systems. The language will be similar to (possibly a subset of) PASCAL. In the course of our system design we have developed a need to produce a decompiler which will generate a program in this process control language given an internal Polish representation. We would like to know if any literature has been produced on the subject of decompilation from an internal version (such as Polish or PASCAL P-Code) to a block structured higher level language such as PASCAL. If any of your readers have information on this topic, we would be interested in corresponding with them.

Thank you very much for your assistance in helping me with this problem. Also, I would like to congratulate you on the excellent job that you have been doing with the newsletter.

Sincerely,

Dave Rasmussen
Dave Rasmussen

DR:ph

P.S. Enclosed is my application for membership for the next academic year along with the membership fee.

State of Florida

Department Of
HIGHWAY SAFETY AND MOTOR VEHICLES
NEIL KIRKMAN BUILDING TALLAHASSEE 32304

RALPH DAVIS
EXECUTIVE DIRECTOR

COL. J. ELDRIGE BEACH, DIRECTOR
DIVISION OF FLORIDA HIGHWAY PATROL
JOHN D. CALVIN, DIRECTOR
DIVISION OF MOTOR VEHICLES
CLAY W. KEITH, DIRECTOR
DIVISION OF DRIVER LICENSES
AUDRY CARTER, JR., DIRECTOR
DIVISION OF ADMINISTRATIVE SERVICES

April 24, 1978

Dear Andy:

I fear that in my hurry to meet the March 20 deadline for my letter I allowed two errors to slip by. They are unimportant, but here are the corrections if anyone wants them.

Paragraph 4 line 5: delete "of".

Paragraph 4 calculations: should be

```
 1000 transactions/day x 30 separators/transaction
≒300000 keystrokes/day
≅4 key entry stations & operators
≅$4,000/month
≐$48,000/year
```

I believe the argument is still valid.

Sincerely,

C. EDWARD REID
Kirkman Data Center

CER:jem

✱ ✱ ✱ ✱ ✱ ✱ ✱

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

78/12/01

Open Letter to all PUG members
from Andy Mickel

The Party is Over

I'm tired and want to quit coordinating PUG and editing Pascal News (effective any time after 1979 July 1). (* Besides, I turn 30 (base 10) on May 4 and you'll no longer be able to trust me! *)

As I said in this issue's Editor's Contribution, my ability to "coordinate" PUG and edit Pascal News may be the best I can do but doesn't seem to be enough. One reaso is the one I mentioned in PUGN #12: PUG is getting too big for me to handle.

I can't continue; I've done all I can, and my endurance, optimism, good humor, lifestyle, physical and mental health are all stretched past the limit. The people closest to me remind me every day.

I'm also a little upset at the seemingly unnecessary growth of politics about standards, extensions, and the future of PUG and Pascal itself. The politics from my point of view seems simply a waste of time. If you reply "it's inevitable," I would answer that it would have happened a lot sooner had PUG been operated and organized in a conventional and ordinary manner.

What About PUG?

What should happen to PUG and Pascal News? I don't know exactly, but there are several possibilities:

1. Disband the organization.
2. Affiliate with a professional society.
3. Institutionalize ourselves and remain independent.
4. Keep PUG the same, but decentralize the work.

Most PUG members I've talked to would like PUG and Pascal News to operate informally, factually, clearly, and in a friendly manner as it has in the past. One person pointed out that by not being formalized, PUG was not susceptible to corrupting influences such as political, social, or economic gain for personal benefit.

As a fifth alternative, I somewhat doubt that anyone of you will be crazy enough to step forward and volunteer to take on all the responsibility like I did from George Richmond three years ago. Therefore I didn't include it in the list above.

Disband the Organization

I occasionally have entertained the idea that perhaps the proper ending for an unconventional organization like PUG would be to simply shut it down. ("For one brief shining moment there was Camelot.") I hinted at this in the Editor's Contribution in PUGN #12.

Shutting PUG down might not be such a bad idea if you realize that entities can outlive their usefulness. In the long term such a decision could be considered brilliant. You must realize that we have largely accomplished two important goals:

1. Making Pascal programming a respectable activity.
2. Getting an officially-accepted Pascal standard.

However, my friend Steve Legenhausen told me that when the Whole Earth Catalog project was stopped, 14-odd cheap imitations appeared, and wouldn't this also happen to Pascal News?

Actually, one of the reasons I've reprinted the roster is to provide insurance against PUG collapsing--any member had all the information necessary to restart the organization.

Affiliate With a Professional Society

Please realize that with close to 3000 members in 41 countries, PUG functions as one of the 10 or 15 largest computing organizations in the world. We're certainly one of the most international.

That is why we are a very desirable "plum" to be annexed by the Computer Society of the Institute of Electrical and Electronic Engineers (IEEE) or by the Association for Computing Machinery (ACM) Special Interest Group on Programming Languages (SIGPLAN).

They have made overtures to us recently, and I asked that they put their offers in writing for publication in a future issue of Pascal News.

Although I think we in PUG do far more to promote good programming ideas and practice than do those organizations (and indeed, in spite of them!) they would offer us subscription and publication services and a guarantee of continuity. The closest example is STAPL (Special Technical Committee on APL) within SIGPLAN. But simply looking at STAPL and its publication Quote Quad might make you forget the idea altogether.

The rates for membership would surely go up, and the "membership services" would not be much better, and the publication and the group would no longer be independent. In fact you would have to join the parent organization (at absurdly high rates) or else pay nearly $10 more than you now pay for a PUG-only membership.

The simple fact that Pascal News would lose editorial freedom (manifested for example by our printing paper clips and crab claws on the cover--or--accepting advertizing, etc.) is a major concession.

Personally, I've always been against this idea, because I never liked the way I was (and am) treated as a member and subscriber to SIGPLAN.

Unfortunately, by being late on issues such as this one, I'm not doing much better!

Institutionalize Ourselves and Remain Independent

The most familiar refrain recently among PUG members besides "Keep up the good work!" and "Hang in there!" is "Keep PUG independent!".

Institutionalization would solve one big headache for me: I was never formally selected to manage PUG, because I've always considered my work "volunteer." But some people have demanded that I "represent PUG" at special conferences and make "official statements." I've always hesitated, because at best I realized my authority was by default, not democratically chosen.

These people always got angry at my hesitation. (Another example of politics that's making me depressed.) I would always point to the simple mechanism of using the Open Forum section in Pascal News to air their ideas.

PUG is in good shape financially (a specific report updating the last report in PUGN 9/10 will appear in PUGN 15). Recently, we have been using the extra $2 of the new $6 membership rate here at PUG(USA) to hire some clerical help (about ¼ of a ½-time secretary).

But, you must recognize that technically Pascal User's Group is a non-profit activity of the University of Minnesota (Computer Center) and that Pascal News is a University of Minnesota publication. I have taken steps all along to ensure that PUG could be transferred somewhere else within one week (really!).

The tremendous benefit we've derived from our warm University of Minnesota home should not be taken for granted. Besides paying my salary, the U of M has provided good production, publication, and mailing facilities. These are the major reasons the cost of PUG and Pascal News in my opinion remains reasonable. And we've done what we have without prostituting ourselves by selling advertizing, without selling the mailing list, and without accepting subsidies from special interests (such as computer companies).

Institutionalization requires a constitution and bylaws, officers, elections, and more of the same old thing (SOT). PUG member Richard Cichelli wrote a proposed constitution and sent the following note to me on 78/08/30:

> "I hope this is a good enough start. Please work over the bylaws dues section to reflect the international situation."
>
> - Rich (Gone Fishing!)

The proposed constitution follows this letter.

In my opinion, this is the best alternative if you want to see PUG continue. However I do not want to serve as any of the officers or as the editor under a constitution and bylaws (I will have my hands full simply affecting the transition over the next one or two years!). The constitution would enable PUG to use authority in standards discussions and in organizing projects promoting rational programming methods. The constitution would also give us the independence we would need to sell advertizing etc., in order to keep the cost of membership low. I don't want to waste my time making money for PUG. Count me out. I'll be the first person to step aside and not be an obstacle to the greater interest.

Keep PUG the Same, but Decentralize the Work

This is not an alternative as far as I'm concerned. It seems that decentralization incurs the horrible tradeoff of high overhead and communications problems. If you say "nonsense" then you may be right, but then I'm the wrong person to coordinate activities. My involvement in a highly decentralized scheme would be less direct work on PUG and Pascal News and more a role of an administrator.

I'm not an administrator or editor; I'm a systems programmer!

People have asked me if there was any chance that I want to do PUG full-time. The answer is "no."

Summary

I (with the generous help from many dedicated people) have had fun organizing PUG and putting together Pascal News. We've done so very informally.

I assumed the editorship after issue #4 when George Richmond (who had edited for 2 years) gave up because of lack of time (his management was not as far-sighted regarding Pascal!). PUG was founded by about 35 persons who attended an ad hoc session at the ACM '75 conference in Minneapolis. I was there and was "volunteered" by everyone to coordinate. So if George is Phase I, then I am Phase II.

I became involved with PUG because I wanted to see Pascal succeed, and I knew that something had to be done urgently to make that happen.

I have had fun in trying to produce a creative and refreshingly different and unconventional publication to promote a programming language. At times, it has been discouraging, and the "bright ideas" offered by "helpful people" have always tended to point back to the ordinary--the same old thing (SOT).

I assure you that the SOT approach to PUG and Pascal News would not have succeeded as well. The conventional wisdom would have doomed Pascal to the role of "just another language." But if Pascal hadn't been an extraordinary language, even unconventional tactics would have failed. The combination was irresistable.

Sometimes I've done things differently just to be different: such as printing paper clips and a screwdriver on the cover of Pascal News. But other differences I implemented as "improvements" are those I had alwayswanted to see in the magazines I had subscribed to. Examples: page numbers on the left in the table of contents; a single, self-explanatory POLICY; enough room on the All-Purpose Coupon to write a 4-5 line address and comments about anything; easy-to-obtain and publicized backissues; and "all the news that fits, we print."

As for price, I would never want to be a PUG member myself if the cost of membership went over $10/year (in 1977 dollars). By keeping things simple and excluding special rates, services, etc., we have also kept the price lower for a longer time, much to the benefit of students (who show the way to the future).

Well, if you are confused, so am I! It has been sheer agony to write this letter, not because I don't want to quit, but because the ideas needed to be stated carefully.

In late October I wrote to ten or so active Pascalers for advice, and I'm grateful to Jim Miner, Rich Stevens, Rick Shaw, Tony Addyman, Bob Johnson, Rich Cichelli, and Jeff Tobias for the advice they offered.

It has been really disappointing to be without the advice of Judy Mullins Bishop, David Barron, and Arthur Sale. They are three persons I would consider to be among the closest to PUG And Pascal News since its beginning. I just know it would have been easier for me if they had responded. Maybe they were too busy.

But, then, that's the problem! Something else must happen. I think it's time for Phase III.

Sincerely,

PROPOSED CONSTITUTION

The following are submitted as a proposed Constitution and initial set of Bylaws for the Pascal Users Group. The Constitution and Bylaws will be accepted or rejected by a simple majority of the ballots (enclosed with this copy of Pascal News) returned to Rick Shaw before April 15, 1979.

A few notes about some of the wording in the documents. First, concerning the choice of an "official" version of the organization's name--apostrophes are bad news in organizational names. The American Newspaper Publishers Association dropped one from their name--let's drop it from PUG. Secondly the term "Chair" is intended to be equivalent to the term "Chairperson". It's just shorter and sounds a little less clumsy.

- Richard J. Cichelli, August 1978

PASCAL USERS GROUP

Official Ballot - October 1978

I believe that the PASCAL USERS GROUP:

____ should institutionalize itself and remain independent.

If so, then I:

do ____ / do NOT ____ approve the submitted Pascal Users Group Constitution;

do ____ / do NOT ____ approve the submitted preliminary Pascal Users Group Bylaws.

My reasons for rejection of either document are:

____ should NOT institutionalize itself, but instead should:

____ disband, or

____ affiliate with a professional society:
____ACM SIGPLAN
____IEEE
____other (____________________), or

____ other:

Return this completed ballot by April 15, 1979, to:

Rick Shaw - PUG
Systems Engineering Labs
6901 West Sunrise Blvd.
Ft. Lauderdale, FL 33313 USA

Your signature need only be on the envelope enclosing the ballot. Rick will certify that voting will be by members only.

Constitution of the Pascal Users Group

Article I Name of the organization

The name of this organization shall be the Pascal Users Group (PUG).

Article II Purpose of the organization

A. The primary objective of PUG is to promote the use of the programming language Pascal as well as the ideas behind Pascal.

B. Specific objectives shall be:

1. to provide channels of communication among members of the international Pascal community (through Pascal News, etc.).
2. to coordinate the efforts of individuals in forming special interest groups within PUG concerned with standards, implementations, etc.
3. to coordinate sponsored research into implementations, uses, etc. of Pascal.
4. to facilitate distribution of Pascal software among PUG members.

Article III Membership

A. General PUG membership requirement

Any person who is interested in the objectives of the Pascal Users Group may become a member upon paying the current annual dues.

B. Voting rights

Formal voting privileges consist of the right to vote at PUG meetings and through mailed ballots on proposed amendments to the PUG Constitution, Bylaws, and standing rules, and on all motions made to and by the Chair. All members are entitled to vote.

Article IV The Officers

A. The government of PUG shall be vested in the Executive Committee which shall consist of:

The Chair

The Vice-Chair

The Secretary/Treasurer

The Editor of the Pascal News

The most recent previous Chair

Three members-at-large

B. The Executive Committee members (excepting the most recent previous Chair) shall be elected for a term of two years by members of PUG.

C. Any member of PUG shall be eligible for any office. The office of Chair may not be held for more than two consecutive terms by the same individual.

D. Vacancies of Office

If any office (excepting members-at-large) shall become vacant, the Chair shall at the earliest possible date thereafter order a special election for the purpose of filling such office. The member thus elected shall take office immediately and shall hold office until the next regular election.

E. Duties of the officers

1. The Chair shall

   a. preside at all PUG meetings
   b. call special meetings at her or his discretion subject to the limitations of Article V, Section E
   c. appoint all committees not otherwise provided for
   d. make provision for the discharge <u>pro tempore</u> of necessary duties of absent members
   e. sign all warrants on the treasury of PUG
   f. see that PUG's regulations are enforced
   g. carry out assignments and instructions dictated by vote of the membership
   h. perform other duties as customarily pertain to the office of Chair

2. The Vice-Chair shall be an aid to the Chair and in case of absence of the Chair shall <u>pro tempore</u> assume and perform the duties of the office of Chair.

3. The Secretary/Treasurer shall

   a. keep a record of all meetings
   b. issue timely notices of meetings and agenda after consultation with the Chair
   c. conduct correspondence of PUG
   d. collect all fees and dues
   e. maintain a list of current (paid-up) members
   f. render an account at least yearly, or more often if required, of all receipts and expenditures
   g. pay the bills of PUG only after approval by vote of the Executive Committee and upon orders or warrants signed by the Chair.

4. Members-at-large and the previous Chair shall attend Executive Committee meetings and vote on issues raised there.

5. The Editor of the <u>Pascal News</u> shall coordinate the publication and distribution of the journal, edit articles, and write editorials.

Article V Meetings

A. Time and Place

At least one regular general membership meeting shall be held each year, the place and time to be determined by the Executive Committee.

B. Voting

A simple majority shall be required to pass all motions. Members present shall constitute a quorum.

C. Meeting procedure

The procedure at all meetings of PUG shall be governed by this Constitution and its Bylaws and by Robert's Rules of Order.

D. Motions

Any member may make a motion to the Chair. This motion must be accompanied by at least one second to the motion by another member.

E. Special meetings

Special meetings may be called when the Chair, after consulting with other Executive Committee members, is convinced that the need is sufficiently urgent. A special meeting shall be called upon the demand of any five Executive Committee members regardless of the wish of the Chair.

Article VI Amendments

A. This Constitution may be amended at any regular business meeting of PUG by a 2/3 vote of those present and voting, provided that written or printed notice of the proposed amendment has been given to all members in sufficient time for it to have been received by them at least one month before the meeting.

B. Bylaws of PUG may be adopted or modified at any regular meeting by majority vote provided that notice has been given as described above.

<u>Bylaws of the Pascal Users Group</u>

Article I Fees and dues

A. The annual dues shall be:

L4.00 (U.K.) per year when joining from Europe, Western Asia, or Northern Africa;

$A8.00 (AUS) when joining from Australia or Eastern Asia;

$6.00 (U.S.) when joining from elsewhere.

These dues are payable in advance during July.

B. Members will receive all <u>Pascal News</u> issues of the July-June year during which they are members, except possibly new members joining after back issues are not available.

C. <u>Pascal News</u> subscriptions are available to libraries and other organizations at $25.00 (U.S.), L15.00 (U.K.), or $A25.00 (AUS) per year.

Article II Meetings

A. Date and time of annual meeting

The annual meeting will be held on the afternoon of the Sunday preceding the Association for Computing Machinery (ACM) annual conference at a location near the conference site.

Article III Sponsoring Affiliates

A. Individuals and organizations wishing to fund colloquia, conferences, research, and other activities of PUG may do so by becoming PUG affiliates, subject to approval by the Executive Committee.

TECHNOLOGISTS

Pipe Line Technologists, Inc.

July 17, 1978

Dear Andy,

In Pascal News #12, J. S. Merritt wrote that he couldn't find the CACM article by Tanenbaum mentioned in PUGN #11, p. 87. I couldn't either. As it turns out, the publication date of December 1977 is wrong. It appeared in the March 1978 issue. Here is the correct reference.

> Tanenbaum, Andrew S. Implications of Structured Programming for Machine Architecture. Comm. ACM 21 (1978), 237-246.

This is a thought-provoking article which implementors of portable Pascal systems should read. It shows the advantages of designing a computer architecture taking into account not only the formal properties of high level languages, but also impirical knowledge of how those languages will actually be used. The result is a stack machine wherein the vast majority of instructions require only one byte of code. Tanenbaum's design is called the EM-1. It could be built as a hardwired computer, microprogrammed, or--and this interests me--as a software interpreter on byte-oriented microprocessors.

The very compact object code of the EM-1 will go a long way toward getting large compilers into small memories and external storage devices. Here are some code space benchmarks (complete programs) for the EM-1 contrasted with carefully handcrafted assembly language programs for the PDP-11, which is normally considered an efficient machine in code space usage:

	EM-1	PDP-11	PDP-11/EM-1
Towers of Hanoi	352 bytes	992 bytes	2.82
Sort integer array	562 "	1,248 "	2.22
Dot product	552 "	832 "	1.51
Find Primes	306 "	704 "	2.30

To produce an assembler and interpreter for the EM-1 machine for all the popular microprocessors would be a worthwile project. I would be happy to talk to anyone interested in the idea.

Sincerely yours,

Charles Hethcoat

Charles L. Hethcoat III

State of Florida

Department Of
HIGHWAY SAFETY AND MOTOR VEHICLES
NEIL KIRKMAN BUILDING TALLAHASSEE 32304

RALPH DAVIS
EXECUTIVE DIRECTOR

COL. J. ELDRIGE BEACH, DIRECTOR
DIVISION OF FLORIDA HIGHWAY PATROL
JOHN D. CALVIN, DIRECTOR
DIVISION OF MOTOR VEHICLES
CLAY W. KEITH, DIRECTOR
DIVISION OF DRIVER LICENSES
AUDRY CARTER, JR., DIRECTOR
DIVISION OF ADMINISTRATIVE SERVICES

July 28, 1978

Mr. Andy Mickel, Editor Pascal News
Computer Center, 227 Exp-Engr
University of Minnesota
Minneapolis, MN 55455

Dear Andy:

Pascal-ers should take note of Edsger W. Dijkstra's article "DoD-I: The Summing Up" in the July 1978 SIGPLAN Notices, pp. 21-26. Many have been proud that PASCAL will almost certainly base the DoD's new standard; the results appear likely to prove that pride not fully justified - not because of shortcomings in PASCAL but in the bureaucracy. To quote Dijkstra briefly,

> ...instead of listing the goals to be reached, IRONMAN already starts the design by prescribing "features" from which it is often hard to reconstruct or guess which sensible goal they are supposed to serve.

And his closing,

> Of ALGOL60 C.A.R. Hoare once remarked that it was a significant improvement over almost all of its successors. What can we do to prevent PASCAL from sharing that fate?

Sincerely,

C Edward Reid

C. EDWARD REID
Kirkman Data Center

CER:jem

THE UNIVERSITY OF MISSISSIPPI
SCHOOL OF ENGINEERING
UNIVERSITY, MISSISSIPPI 38677

Computer Science

601-232-7353

29 July, 1978

Andy Mickel
University Computer Center: 227 EX
208 SE Union Street
Univ. of Minnesota
Minneapolis, MN 55455

Dear Sir,

I have been using PASCAL here at Ole Miss for the past two years on the DEC-10. I currently have available two compiler writing tools written entirely in PASCAL:

(a) LEXGEN--An Automatic Lexical Analyzer Generator The generator takes regular expressions for any number of lexical tokens as input and outputs the minimized finite automaton for accepting any of that set of lexical tokens. Intermediate user-controlled output includes diagrams showing how the NFA is constructed, the complete NFA (in graph form), the resulting DFA (in tabular form). These intermediate outputs should be especially useful for teaching the theory and application of this type of lexical analyzer.

(b) LALR1--An LALR(1) Syntax Analyzer. Given the BNF description of a grammar this program outputs the LALR(1) tables for driving a parser. Indication is given whether grammar is SLR(1), LALR(1), or neither.

Either of these programs and their documentation is too large or of a specialized nature to be included in the new algorithms section of Pascal News; however I invite any interested parties to contact me directly.

Sincerely,
Ralph D. Jeffords
Ralph D. Jeffords
Asst. Prof. of Computer
Science
University of Mississippi

AUGUST 23, 1978

PASCAL USERS' GROUP
ATTN ANDY MICKEL
UNIVERSITY COMPUTER CENTER: 227 EX
UNIVERSITY OF MINNESOTA
208 SE UNION STREET
MINNEAPOLIS MN 55455

Andy,

I guess it's about time for me to renew my PUG membership, so I've enclosed an old "ALL PURPOSE COUPON" and a check for $10.00 in devalued American currency (since PUG membership fees have surely increased by now). If even $10.00 isn't enough for two years, let me know how much more is necessary, and I'll send the balance ASAP.

I hope all is going well with PUG -- I have some doubts since I haven't heard a word from you people since March. Was there another 77-78 issue published after #11? If so, I have never received it, and I'd hate to miss anything!

A few weeks after attending the 2nd West Coast Computer Faire, I took a job as "designated internal programmer" for North Star Computers, here in Berkeley. North Star is best known for its mini-floppy diskette subsystem, which is compatible with any 8080- or Z-80 based mainframe incorporating the S-100 bus. To date, the firm has supported only BASIC (albeit a powerful, feature-laden version of the language), but Mr. Thos Sumner (who does software evaluation for NS) and I have convinced the "powers that be" to look seriously into supporting Pascal as both an internal software development tool as well as a marketable software product. It is almost certain that Ken Bowles' group will develop a version of their UCSD Pascal system which will operate on 8080 or Z-80 machines using North Star disk units, but it isn't clear at the moment whether or not North Star will itself support and/or market the system (though I am personally lobbying for such a development).

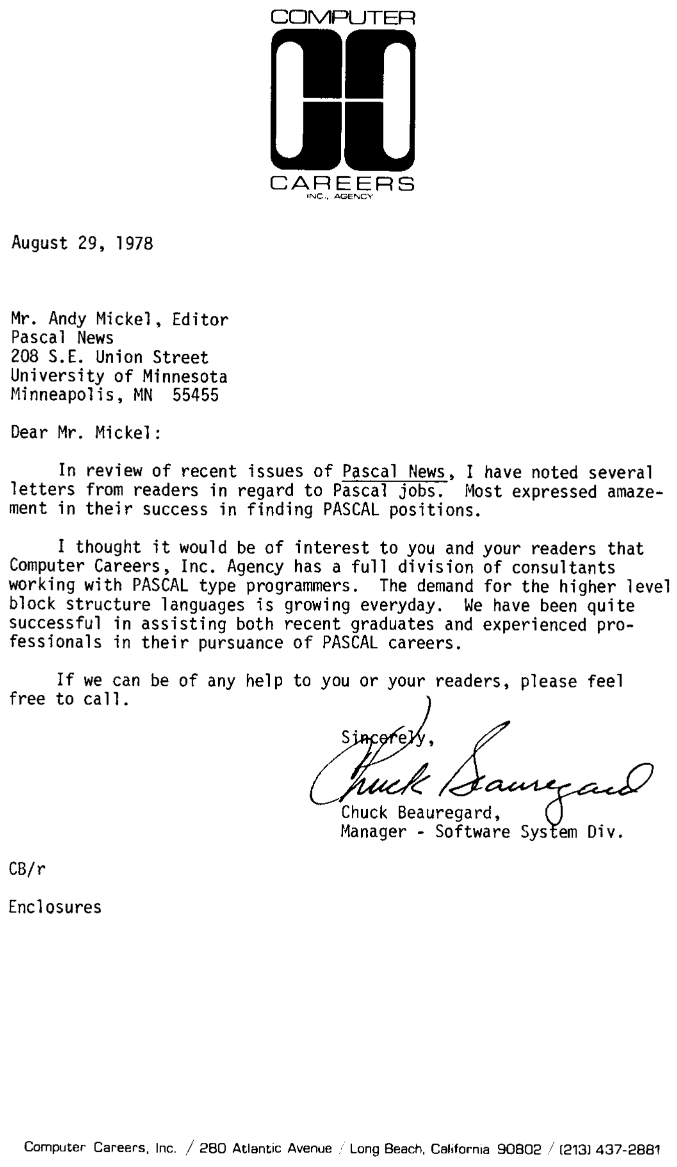
Regarding my somewhat sceptical comments on Pascal in the micro-world as published in PNEWS 9/10, I am pleased to note that UCSD Pascal seems to have "done the trick" and catapulted full-blown Pascal into the marketplace, at an extremely reasonable cost, yet! Finally! In the spirit of "hit 'em again, harder", Part 1 of my own tutorial series on the language, "Pascal, from beginning to end", will appear (after innumerable crazy circumstances and delays) in the September-October issue of Creative Computing magazine. With luck, my own, and other, similar articles will serve to bootstrap the consciousness of personal/micro-computer users into the Pascal era. (Notice the Pascal-oriented August, 1978 issue of Byte, for example.)

From my vantage point, in the midst of the small-systems market, I see Pascal's momentum increasing at an astonishing rate. It appears that we now have the ball. Let's all pull together and run with it -- now that many computerists are accepting Pascal as a "real" language, there must be a concerted effort on the part of we who support the language to provide documentation and software (systems and applications) for it. (Need I add that this can also be quite lucrative?)

JIM MERRITT
PO Box 4655
BERKELEY CA 94704
PHONE 415-845-4866

Keep in touch,

78/08/28



COMPUTER
CAREERS
INC., AGENCY

August 29, 1978

Mr. Andy Mickel, Editor
Pascal News
208 S.E. Union Street
University of Minnesota
Minneapolis, MN 55455

Dear Mr. Mickel:

In review of recent issues of Pascal News, I have noted several letters from readers in regard to Pascal jobs. Most expressed amazement in their success in finding PASCAL positions.

I thought it would be of interest to you and your readers that Computer Careers, Inc. Agency has a full division of consultants working with PASCAL type programmers. The demand for the higher level block structure languages is growing everyday. We have been quite successful in assisting both recent graduates and experienced professionals in their pursuance of PASCAL careers.

If we can be of any help to you or your readers, please feel free to call.

Sincerely,

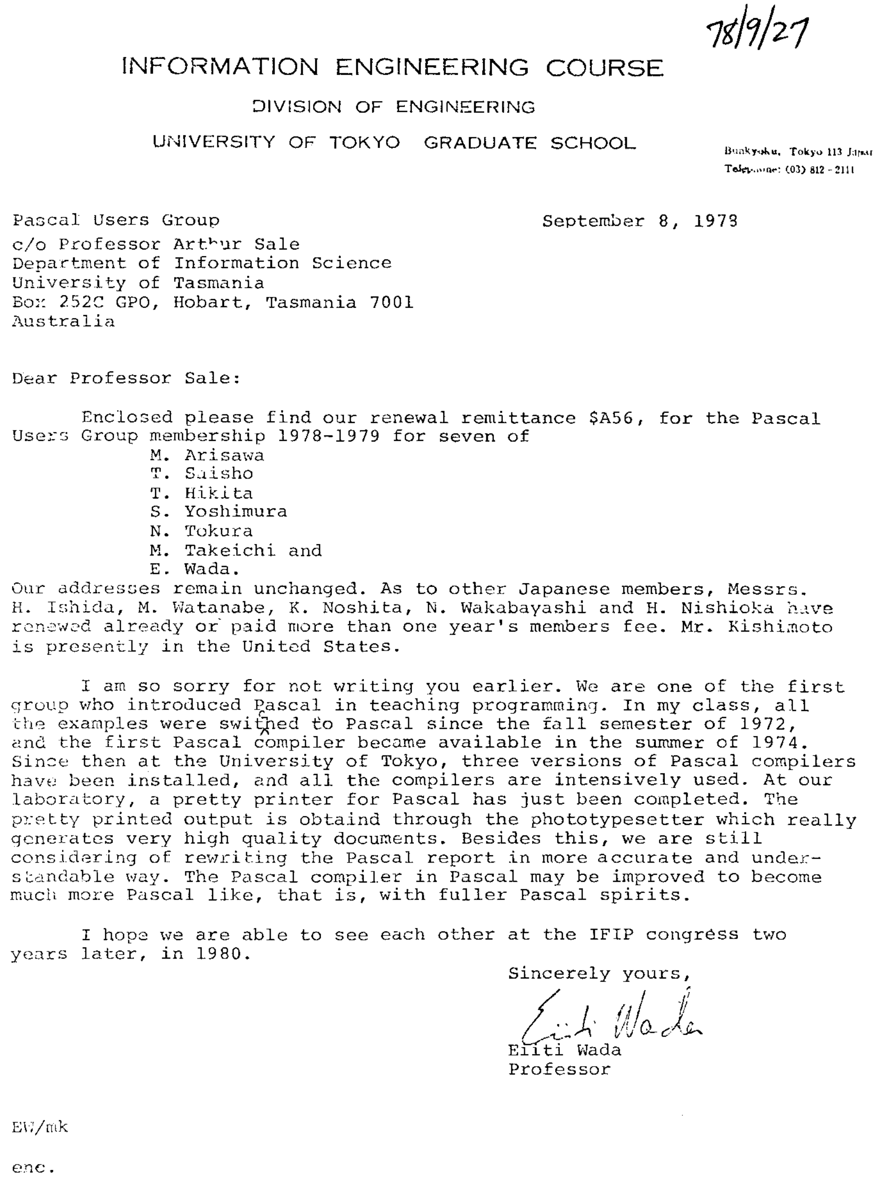
Chuck Beauregard

Chuck Beauregard,
Manager - Software System Div.

CB/r

Enclosures

Computer Careers, Inc. / 280 Atlantic Avenue / Long Beach, California 90802 / (213) 437-2881



78/9/27

INFORMATION ENGINEERING COURSE
DIVISION OF ENGINEERING
UNIVERSITY OF TOKYO GRADUATE SCHOOL

Bunkyo-ku, Tokyo 113 Japan.
Telephone: (03) 812-2111

Pascal Users Group
c/o Professor Arthur Sale
Department of Information Science
University of Tasmania
Box 252C GPO, Hobart, Tasmania 7001
Australia

September 8, 1978

Dear Professor Sale:

Enclosed please find our renewal remittance $A56, for the Pascal Users Group membership 1978-1979 for seven of
M. Arisawa
T. Saisho
T. Hikita
S. Yoshimura
N. Tokura
M. Takeichi and
E. Wada.
Our addresses remain unchanged. As to other Japanese members, Messrs. H. Ishida, M. Watanabe, K. Noshita, N. Wakabayashi and H. Nishioka have renewed already or paid more than one year's members fee. Mr. Kishimoto is presently in the United States.

I am so sorry for not writing you earlier. We are one of the first group who introduced Pascal in teaching programming. In my class, all the examples were switched to Pascal since the fall semester of 1972, and the first Pascal compiler became available in the summer of 1974. Since then at the University of Tokyo, three versions of Pascal compilers have been installed, and all the compilers are intensively used. At our laboratory, a pretty printer for Pascal has just been completed. The pretty printed output is obtaind through the phototypesetter which really generates very high quality documents. Besides this, we are still considering of rewriting the Pascal report in more accurate and understandable way. The Pascal compiler in Pascal may be improved to become much more Pascal like, that is, with fuller Pascal spirits.

I hope we are able to see each other at the IFIP congress two years later, in 1980.

Sincerely yours,

Eiiti Wada

Eiiti Wada
Professor

EW/mk

enc.

Health Products Research, Inc.

3520 U.S.Route 22, Somerville, New Jersey 08876 · (201) 534–4148
Cable Address: "Healthpro"

European Office:
Heuberg 12
4051 Basel, Switzerland
Telex: 63972

23 September 1978

Mr. Andy Mickel, Editor
Pascal News
University Computer Center: 227 EX
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455

Dear Andy:

As 'Pascal Coordinator' for the Amateur Computer Group of New Jersey (ACG-NJ), I am in a position to report some good news about the enthusiasm for Pascal among computer hobbiests in the New Jersey area:

a. The ACG-NJ has taken advantage of the group subscription offer of Ken Bowles' group at the University of California at San Diego. Approximately twenty members have obtained the UCSD implementation through the ACG, and at least five have it "up and running" on their personal systems. Most of these systems are 8080/Z-80 microcomputers, although there are two or three LSI-11s as well.

b. I gave a brief talk on Pascal at this month's ACG-NJ meeting, which was well received; I have also been invited to speak on Pascal to the New York City amateur computer group in December.

c. At least sixty people attended a "Pascal Users' Group" session at the 'Personal Computing '78" show, held in the Philadelphia Civic Center at the end of August. This turnout was mildly astonishing in view of the fact that the session on Pascal was a last-minute addition to the program, not publicized except by posters put up on the first day of the show, and scheduled on the show's last day. A show of hands at the start of the session yielded the following statistics:

Persons who had used UCSD Pascal:	None
Persons who had used another Pascal:	16
Persons who wanted but did not have Pascal:	25
Persons who didn't know whether they wanted Pascal or not:	6
Persons who knew they didn't want Pascal:	None

I wish to thank Mr. Robert Hofkin of UCSD, who happened to be at the show on business, for stopping by and helping field some of the questions.

d. Three other noteworthy presences at "PC '78" were those of three companies selling UCSD Pascal with their computer systems:

(1) Northwest Microcomputer Systems, of Eugene, Orgeon, demonstrated their "Programmer's Workbench", a desk-top system containing an 8085 microprocessor running at 3 (optionally 5) MHz, 56K 8-bit bytes of memory, dual magnetic diskette drives, and a video display, priced at $7495. I understand that two people from Zurich wanted to pay cash and walk away with one of the two systems on display (they didn't because the system wouldn't have run on Swiss electrical power without a modification too extensive to be done at the show), and that Carl Helmers, editor of BYTE magazine, was responsible for the disappearance of one of the systems on the second day of the show, being "unable to resist the desire to take it to his hotel room and play with it".

(2) ALTOS Computer Systems, of Santa Clara, California, demonstrated their "ACS8000" system, featuring a Z-80 microprocessor running at 4 MHz, up to 64K bytes of memory, and one or two magnetic diskette drives. Price for a minimal system with 32K bytes of memory and a single drive: $3,840. (This system does not include a built-in videa display.)

(3) Alpha Microsystems demonstrated UCSD Pascal as a subsystem of their multi-user system, whose CPU is based on the Wester Digital WD-16 chipset. Workspace available to a single Pascal user in this system would be restricted to 48K bytes, but the system supports multi-megabyte hard-surface magnetic discs as well as (or instead of) diskettes. I regret I do not have their prices readily at hand.

e. The August issue of BYTE magazine had a cover portraying "Pascal's Triangle", an area of smooth water with well-marked channels bordered by such less hospitable places as the turbulent "FORTRAN Ocean", the desolate "Isle of BAL", the "JCL Barrier Reef", the "Straights of COBOL" (in which much commercial traffic is seen), the perpetual fog bank wherein lie the "exotic and mysterious jungles of LISP", and the "interactive and weed-filled Sea of BASIC". Several vessels, ranging from warships to tiny rafts, are fleeing to the safety of the Triangle.

On a more serious level, the same issue of BYTE contained five articles on Pascal, including one by Ken Bowles himself entitled "PASCAL VERSUS COBOL: Where Pascal Gets Down to Business". This last article may be especially important, since there seems to be a consensus among those involved in the "personal computer" industry that the big market right now is small business systems, for which the greatest lack is high-quality software.

A less welcome development is the discovery that the UCSD is no more immune than any other vendor to the announce-it-early, deliver-it-late syndrome: I have been waiting since mid-August for their Release I.5, my phone calls every other week being taken by a pleasant but apparently not-too-knowledgeable young person who assures me that the Release will be forthcoming "in another week or two". I guess we should be thankful we get anything at all!

Keep up the good work!

Sincerely,
Rod Montgomery
Rod Montgomery

P.S.: I prepared this letter on my personal system using the screen editor that comes with UCSD Pascal. It works!

The Commonwealth of Massachusetts
Fitchburg State College
Fitchburg 01420

July 10, 1979

Andy Mickel
Pascal Users Group
Univ. of Minnesota

Dear Mr. Mickel:

Enclosed is my renewal for the coming year. I have truly enjoyed receiving PUG newsletter. (I finished #12 in less than 12 hours and still haven't read CACM May!) The new section on APPLICATIONS should be an excellent media for transmittal and evaluation of programming methods.

We at Fitchburg State College have just totally restructured our course structure to put Pascal into the Freshman year where it belongs. Other languages are taught within the courses which require them and assume a knowledge of Pascal.

Re: standardization of Pascal. I vote for Charles Fischer's method (PUG #12, pg. 54), a standardized set of extensions designed by a small group and an all-or-nothing vote by PUG membership. I have a great many changes I'd like to see in Pascal; but, I'drather see a standard. I'm sure a lot of other people feel the same way.

Keep up the excellent work.

Sincerely,

Kenneth R. Wadland
Computer Science Program
Fitchburg State College

* * * * *

Oct 18 '78

DEAR ANDY,

THERE IS A NEED FOR A BOOK TO BE PUBLISHED ABOUT PASCAL. IT WOULD HAVE TO DEFINE A STANDARD SUCH AS WIRTH'S (7), HAVE A COMPLETE LISTING OF THE COMPILER WITH THE GENERATED CODE, AND AIDS FOR BOOTSTRAPPING. HANSEN'S (2) BOOK COMBINED WITH HARTMANN'S (3) COMES CLOSE TO THE IDEAL. IT WOULD BE ON THE ORDER OF MCKEENAN'S (4) FOR XPL, RANDELL'S (5) FOR ALGOL 60, HALSTEAD'S (1) FOR NELIAC, OR WAITE'S (6) FOR STAGE 2.

PERHAPS TWO OR THREE LEVELS COULD BE INCLUDED IN ORDER TO BE IMPLEMENTED ON A MICRO, MINI, OR MAXI-COMPUTER. PASCAL P0 IN WIRTH'S (7) IN NOT COMPLETE. (MISSING GENERATED CODE AND WRITTEN IN A HIGHER LEVEL THAN P0). PASCAL-S IN WIRTH'S (9) ALSO HAS MISSING GENERATED CODE. PASCAL P4 PERHAPS IS COMPLETE, BUT BUT THESE THREE LEVELS ARE APPROXIMATELY WHAT IS NEEDED.

1) HALSTEAD: MACHINE INDEPENDENT COMPUTER PROGRAMMING.
2) HANSEN: ARCHITECTURE OF CONCURRENT PROGRAMS
3) HARTMANNS: A CONCURRENT PASCAL COMPILER FOR MINI-COMPUTERS
4) MCKEEMAN: A COMPILER GENERATOR
5) RANDELL & RUSSELL: ALGOL 60 IMPLEMENTATION
6) WAITE: IMPLEMENTING SOFTWARE FOR NON-NUMERIC APPLICATIONS
7) WIRTH & JENSEN: PASCAL USER MANUAL & REPORT
8) WIRTH: ALGORITHMS + DATA STRUCTURE = PROGRAM
9) WIRTH: PASCAL - S - A SUBSET AND IT'S IMPLEMENTATION

William C. Moore Jr.
WILLIAM C. MOORE, JR.

* * * * *

ComputerAutomation
18651 Von Karman
Irvine, Ca. 92713

October 10, 1978

PASCAL User's Group
C/O Andy Mickel

Dear Andy:

At Computer Automation's NAKED MINI Division, PASCAL is gaining interest and support. Our compiler on DOS4 produces code for a virtual machine. I have recently converted the machine to work under our new operating system OS4 on the NM4 series computer. The same compiler now runs under DOS2, DOS4 and OS4. For marketing information, contact Laura Cvetovich (M/S 1167).

PUG members might be interested to learn that several openings in system software development are available at CA requiring PASCAL and assembly language experience. The inside track can be had by writing Dave Robertson (M/S 1175).

Keep up the good work,

D. J. Maine
D. J. Maine
Research Scientist
M/S 1175

P.S. Bob Hutchins says HI to his friends at PUG!

UNIVERSITÄT HAMBURG

INSTITUT FÜR
INFORMATIK
Prof. Dr. H.-H. Nagel

Institut für Informatik
2 Hamburg 13, Schlüterstraße 66-72

Mr. Andy Mickel
Editor, PASCAL News
University of Minnesota
University Computing Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455
USA

Fernsprecher: 040-4123-4151 } Durchwahl
Behördennetz: 9.09(.)

Telex-Nr.: 2 14732 uni hh d

Datum und Zeichen Ihres Schreibens

Aktenzeichen (bei Antwort bitte angeben.)
Na/Ja

Datum
September 25, 1978

Dear Mr. Mickel,

last week I came around to study PASCAL News no. 12 which had arrived at my office during the time I was in vacation. I would like to congratulate you to the fact that this issue must be the third aniversary of starting this activity by you. You have done - in collaboration - with other people who followed your example a very fine job which I consider to be of great importance not only to the community of people who use PASCAL now. It may be even more important for those who are enabled to obtain information about PASCAL and its implementations in a rapidly expanding environment of small system users.

Regarding our DECSystem-10 implementation we have been busy, too. Currently we are testing an improved PASCAL Compiler Version for this system which employs a register allocation algorithm to generate more effective object code. This algorithm is based on the work of Ammann. However, we had to modify it to accommodate the special code generation for the PDP-10 processors. In addition we have removed bugs which had been brought to our attention and adapted the compiler to the more advanced instruction set of the KI-10 processor (double word move etc.). In addition we removed the special file variable TTY introduced for interactive use of PASCAL. We now direct the standard input/output to the user terminal. According to our agreement we have already modified to the "otherwise" extension recently agreed upon. We are currently implementing special conversion routines for input from the user terminal. Instead of aborting in case of typing errors an error message will be output indicating the error detected and the user will be prompted to retype the desired input.

As I had promised to N. Wirth I intend to incorporate those extensions for which a standard form will be agreed upon.

Before I obtained the PASCAL News no. 12 indicating the new rates I had already mailed a check over $ 8 to you originally intended to cover my dues for two years. As I now understand distribution of PASCAL News to Europe costs 4 £ per year. Therefore I suggest you credit these 8 $ as my dues for the year ending June 1979. I shall send the next dues to UK directly in 1979.

Yours sincerely,
H.H. Nagel

CALIFORNIA INSTITUTE OF TECHNOLOGY

PASADENA, CALIFORNIA 91125

DIVISION OF BIOLOGY 216-76

Andy Mickel
Editor, Pascal News
University Computer Center
227 Exp. Eng. Bldg.
Univ. of Minnesota
Minneapolis, Minnesota 55455

Dear Andy,

Judy Bishop's discussion of subranges and conditional loops (Pascal News #12, pp 37, 38 and 51) clearly states a basic problem in standard Pascal: how to keep index variables within their subrange at all times. However, her solution does not seem entirely satisfactory to me because (1) as she noted, single letter prefixes would hinder readibility; (2) the necessary extra type definitions are a hassle both to write and to read; and (3) the extra allowed value of the index may in some cases degrade the ability of the run-time checks to stop an error at it's source.

Some Pascal compilers, such as the Brinch-Hansen DEC-10 compiler in use here at Caltech, allow the loop .. exit if construction. Loop .. exit if is the most general form of the conditional loop, since it contains a statement block before the test of the exit condition (as does repeat .. until) and another statement block after it (as does while). This generality is necessary for a natural solution to some problems, including this one.

Having defined i on the subrange min..max, we can write:

```
i := min;
loop
    (* something *)
exit if (i=max) or (condition);
    i := succ(i)
end;
```

Thus we always have i <= max. In standard Pascal, a somewhat less elegant but equivalent construction is available:

```
i := min;
while true do begin
    (* something *)
if (i=max) or (condition) then goto 10;
    i := succ(i)
end;
10: (* next statement *)
```

yours sincerely,
Karl Fryxell
Karl Fryxell

CRAY RESEARCH, INC.

7850 Metro Parkway, Suite 213, Minneapolis, MN 55420 • (612) 854-7472

August 16, 1978

Mr. Andy Mickel
Pascal User's Group
University Computer Center, 227 EX
208 SE Church Street
University of Minnesota
Minneapolis, Minnesota 55455

Dear Andy:

I'm enclosing twelve (14_8) dollars for two years PUG dues. If you don't think PASCAL will survive for that long please return some of my dues.

Some general comments. First in response to "What To Do After A While" you need two new operators:

a AND THEN b; C OR ELSE d.

In a more serious vein, many PUG articles contain phrases like "stamp out FORTRAN", or "kill the dinosaur", etc. The articles seem to be written with all of the grace and charm of a stiff necked missionary trying to convert a bunch of ignorant heathens. Why is there such an emotional investment in promoting PASCAL? PASCAL, like most human inventions, has some good points and some bad points. PASCAL was implemented on a CRAY-1 Computer by a group at Los Alamos. There also exists a group of 18 short "kernels", called the Livermore Kernels, which (allegedly) are typical of the bulk of the computer usage at places like LASL. When coded in PASCAL the kernels ran (last January) with an average "speed" of about 3.6 MFLOPS (million floating point operations per second). If a second program is used to optimize the code generated by PASCAL the rate goes to 5.7 MFLOPS. When run using the current CRAY FORTRAN Compiler the rate is about 22 MFLOPS, planned FORTRAN enhancements (for "this year") should bring it to over 30 MFLOPS. Now, there are significant differences in implementation strategies between the LASL PASCAL and CRAY FORTRAN and it would be very wrong to conclude (from this example at least) that PASCAL is not a good language. However, with performance ratios of between 4 and 10 (depending on one's point of view) on a system that costs up to $9 million, it seems just as wrong to conclude that "FORTRAN is obsolete". If PASCAL is to become a universally used language won't implementation become machine dependent with additions (and deletions) to take advantage of particular hardware?

Page #2

This leads to the second point. I understand that there recently was a first (annual?) PASCAL standards meeting. I've heard from two different people that the PUG representatives were adamant (to the point of being obnoxious) that nothing in PASCAL should be changed, Wirth has spoken and not a "," must ever be changed. Is this realistic? As the language is used shouldn't it grow, much like English or FORTRAN when deficiencies are discovered? AFTERALLTHEREAREFEATURESINPASCALWHICHDONOTNECESSARILYMAKEA PROGRAMEASILYUNDERSTANDABLE.

More important, is this a legitimate stand for a "PUG representative" to take. I'm a PUG member and I don't recall ever being asked whether or not PASCAL should be changed. Certainly everyone is entitled to an opinion about the future of PASCAL but shouldn't a "PUG representative" somehow survey his members? It seems to me that most of the articles in PASCAL NEWS deals with proposed additions or deletions and most of the implementations mention deletions.

I hope this gives you something to write an editorial about.

Best -

Richard A. Hendrickson

RAH:al

(* In a phone call to Dick in early October, I (Andy) thanked him for the letter and explained that one reason that Pascal is at a disadvantage when compared to FORTRAN is because of the vast difference in the person-years put into compilers, libraries, etc. However, I appreciated the data he provided and his feelings. I also told him that the "standards meeting" he referred to was instead the UCSD workshop on extensions, and that the so-called obnoxious PUG representative was Richard Cichelli who indeed upset many persons. Since I wasn't there, I can only repeat the reports I have heard. I explained that human languages and programming languages are vastly different, and no, programming languages shouldn't necessarily grow, and in fact Edwin Newman's recent books (one is Strictly Speaking) deplore the unnecessary "growth" in the English language. And if you have a decent Pascal implementation, ThenThereAreFeaturesInPascalWhichAreExtremelyElegantInAidingProgramReadability.*)

University of Sheffield

Department of Applied Mathematics and Computing Science
Professors
D N de G Allen, W D Collins, S C Hunter, J R Ullmann

Sheffield S10 2TN
Tel: Sheffield 78555
STD code: 0742

Andy Mickel,
Editor, Pascal News,
University Computer Center; 227EX ,
208 SE Union Street,
University of Minnesota,
Minneapolis, MN 55455, U.S.A.

4th September, 1978

Dear Andy,

My copy of Pascal News #12, mailed in Minneapolis on June 23rd, arrived on August 31st. In it I read that the publication deadline date for #13/14 was August 15th, 16 days earlier! Can this be true? I hope not. Please try to squeeze in the enclosed paper "Know the state you are in". Although written in great haste it might solve a few problems for a few people or at least shed a new light on them.

I was interested to read Judy Bishop's comment about booleans (Pascal News #12, page 51). Since first teaching Pascal three years ago I have encouraged my students to use two-state scalars and case in preference to booleans and if. The programming style in my enclosed paper is a natural consequence of this. I have also been following recent work by experimental psychologists studying the (detrimental) effects of negation in programming logic and, in particular, the negation implicit in else. One consequence was that I submitted a paper to CACM in March of this year supporting two-state scalars and case in preference to booleans and if and, of course, praising Pascal for encouraging this approach. So please note, Judy, your anticipated paper "Booleans considered harmful" has already been written!

Yours sincerely,

Laurence Atkinson

L.V. Atkinson

*****

University of the Witwatersrand, Johannesburg
DEPARTMENT OF APPLIED MATHEMATICS

1 Jan Smuts Avenue, Johannesburg, 2001, South Africa
Telephone 39-4011, Telegrams 'University', Telex 8-7330 SA

Mr. T.M.N. Irish,
5 Norse Way,
Sudbury,
Chepstow,
Gwent NP6 7BB,
United Kingdom

telephone ext
your reference
our reference JMB/sw
date 27 September 1978

Dear Mr. Irish,

Many thanks for your note on "What to Do After a While". I would just like to clarify our points of agreement and disagreement and then go on to explain why we think the sequential conjunction/ boolean operator contraversy can now die a natural death.

1. We take your point about potentially undefined factors. What you are saying is that the a[i] in (a) below is permissable because it is an expression, but the a[i] in (b) is not, because it is a factor.

```
(a) if i<=n then if a[i] = ....
(b) if (i<=n) and (a[i] = ...)
```

While this may be a valid distinction, it is a hard one to grasp. After all, the a[i] in (a) starts off as a factor!

2. To POP-2 and RTL/2 remember to add Wirth's new language Modula and Euclid. All of these specify that factors in a boolean expression will only be evaluated while necessary. The Boolean algebra connotation may be old, but it is certainly no longer strong in the world of language design.

3. The andop function is "wrong". Moreover, the loops in our Appendix examples 2 and 3 should be repeat's, strictly speaking.

We could argue on and on about this for ever. Fortunately, the problem - that of searching a list to our satisfaction - has been solved in a completely novel way by Laurence Atkinson of Sheffield University. He brought to our notice the following solution which takes account of the fact that there are three states in the loop, represented by

```
i<=n and a[i] <> item   :  scanning
i<=n and a[i] = item    :  found
 i>n                    :  notthere
```

```
Solution 4. USE A STATE VARIABLE
            var table : array [1..maxsize] of whatever;
                state : (scanning,found,notthere);
            :
            :
            index:=1;  state:= scanning;
            repeat
            if index > maxsize then state:= notthere else
            if table[index] <= item then state:=found else
               index:=index+1
            until state <> scanning.
```

It may not be as short and sweet as your favourite solution but it works for all cases and does not need additional elements. Incidentally, this method still requires index to be declared over 1..maxsizeplusone. (See Mullins PN12 (1978)"Subranges and Conditional Loops").

I think we should let this matter rest now. In a sense no-one has won - we can't have undefined factors, you can't have side effects in functions. Pascal is a double edged sword, but it is very sharp for those who care to use it properly, as Atkinson has shown us.

Best wishes,

Judy

Judy Bishop

PASCAL STANDARDS

Please direct all enquiries for this section to Tony Addyman or Rick Shaw

Tony Addyman
Dept. of Comp. Sci.
Univ. of Manchester
Oxford Road
Manchester, England
M13 9PL U. K.

Rick Shaw
Systems Engr. Labs
6901 W. Sunrise
Ft. Lauderdale, FL
33313 USA

Much has happened since issue #12 last June. Rick Shaw is now Tony's "right hand" in the USA. Thanks to Tony and Rick, Standards discussions are placed within the Pascal User's Group where it belongs. Arthur Sale was selected to chair the Australian Standards Assoc. committee on Pascal Standards (MS/20).

On 78/06/18, Niklaus Wirth wrote that there was one error in the EBNF syntax published in Pascal News #12, June, 1978 on page 52. The definition of FieldList should be:

```
FieldList = FixedPart [";" VariantPart] | VariantPart
```

Below are reports from Tony, Rick, and Brian Wichmann. Rich Cichelli reported that when they are ready, he will distribute the Wichmann-Sale Validation Suite and a standards-conforming checking program. (There exists a similar program developed for Pascal programs by North-American Philips Corp. which checks to see if a program conforms to the language accepted by the Pascal-P compiler.) Rich expects that he will be able to distribute this software for a reasonable fee.

Tony's working group produced a third draft of the BSI/ISO standards document which will appear as PUGN #14 (January, 1979). The BSI/ISO standards effort, incidentally was unanimously endorsed by the participants of the UCSD workshop on extensions in July. On October 11, it was reported that the ISO vote on the BSI proposal was 8 in favor (U.K., U.S.S.R., Brazil, Canada, Italy, Germany, USA (with qualification) and The Netherlands (with qualification); 1 opposed (Japan), and 10 abstentions (!?). Also in October, ANSI announced the formation of X3J9, a committee for examining the ISO standard to be adopted as an American standard.

Rick Shaw, Rich Cichelli, and Jim Miner will attend as PUG's official representatives to the December 19 meeting.

News from the International Working Group on Pascal Extensions

In PUGN #12 we announced the formation of this group: a small number of competent implementors of "major" Pascal implementations were chosen by Niklaus Wirth. Why only a few people? As Bob Vavra stated in #12, a PUG committee-of-the-whole is unthinkable; with everyone interacting it won't work. We must rest assured that if someone has a great idea, it will certainly be recognized. Even with a few people, it has been an overwhelming amount of work (forests of paper have been consumed!). So the project is delegated to a small group for good or ill. Here is the invitational letter from Niklaus:

ETH EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE
ZÜRICH

Institut für Informatik

January 30, 1978

Dear Andy,

The "Standardization of Pascal" is a recurrent theme. As you probably know, I have been rather reluctant to get involved in such an effort, being aware of the time-consuming nature of ill-defined and politics infested endeavours. Nevertheless I am also aware of some genuine motivations for obtaining a "standard".

A recent visit of Professor Jorgen Steensgaard-Madsen from Copenhagen, implementor of Pascal for the Univac 1100, has brought up the topic again. We have had some refreshingly productive discussions. The gist of them is that we should try to obtain a consensus among a few implementors of Pascal on major computers on at least some of the pending problems. Their agreement to work on such a consensus and to implement the results on their machine would in our opinion be the most effective way to reach a standard that does not only exist on paper and evokes a lot of discussion and controversy, but will effectively be adhered to.

Jorgen has agreed to work out a draft of a working document within the next two or three months. We are solliciting your suggestions. If a positive response should emerge, we would envisage a meeting, preferrably sometime this summer. I would appreciate to know your reaction to such a plan.

The draft document to be worked out rests on the basic assumption that Pascal as defined by the Revised Report shall essentially remain unchanged. It shall concentrate on three topics:

1. Standard representation of programs in terms of standard character sets, and definition of the set of standard procedures, types, etc., 2. Clarification of issues that are left open by the Report (such as type equality), and 3. Extensions. We agreed that the following topics would be included:

1. Specifications of the types of parameters of formal procedures. This would be the only point involving an actual change of Pascal, since it would require that such types be specified.
2. Array parameters, especially the possibility of omitting the specification of index bounds for formal arrays. This might or might not include dynamic actual arrays.
3. An "otherwise" clause in the case statement.
4. Structured constant definitions.
5. External procedures and "forward" declarations.
6. Standard procedures for reading text files according to the program schemata used for regular files.

I am looking forward to your reply and suggesions and hope that with your dedicated help a contribution towards a much discussed goal may evolve. Please send a copy of your reply directly to Jorgen.

Sincerely yours,

Niklaus

Prof. Niklaus Wirth

cc: O. Lecarme, Université de Nice, France (CII)
A. Mickel, University of Minnesota, USA (CDC)
H.H. Nagel, Universität Hamburg, Germany (DEC)
J. Steensgaard-Madsen, University of Copenhagen, Denmark (Univac)
J. Tobias, Australian Atomic Energy Commission, N.S.W. Australia(IBM)
J. Welsh, Queen's University of Belfast, North Ireland (ICL)

Jim Miner and I suggested in February in our response to this letter that Arthur Sale, Tony Addyman, and Ken Bowles be added to the list because Arthur's Burroughs B6700 and Ken's microprocessor interpreters were major implementations, and Tony had been doing all of the standards work so far. They were added. We promised in issue #12 to report on the results. In the 3 months of activity (from April to June) no one would have predicted the amount of controversy and heap of paper generated by the 10 participants. It is an example of the "frailty of human interaction as opposed to problems caused by individual personalities." Nevertheless the Working Group rebuffed the hack changes done by individual implementors by concentrating on just a very few issues. We finally agreed on some results. Our first result involved a conventional form for the almost universal extension providing an "otherwise clause" to the case statement. Arthur Sale presented the report below for publication:

International Working Group on Pascal Extensions

Consensus Position on Case defaults

1. Background

The International Working Group is a group of implementors of Pascal set up by Niklaus Wirth and the Pascal Users Group to responsibly draft some key extensions to the programming language Pascal. The following report details the first consensus decision by the Group, and is published in *Pascal News* in the interests of other implementors and to achieve rapid dissemination of information.

The term *conventionalized extension* is used here to mean that the feature described is not to be considered as part of the standard language Pascal, but rather that some implementations may include the feature in accordance with the conventions suggested by the Working Group. The purpose of conventionalizing extensions is to

(i) enhance portability of programs which use the extension, and
(ii) ensure a concern for the integrity of Pascal in making extensions.

The following minor extension to the language is the first consensus decision by the Working Group and is to be regarded as a conventionalized extension.

2. Notation

The modifications to the syntax will be described in EBNF notation, as this is likely to be the form used in the draft standard for Pascal, and can be used to avoid repetition or the introduction of new non-terminal symbols.

3. Purpose

The extension described allows a construction to which control is transferred if the selector expression of a case statement fails to match any case constant (label) in the statement. The construct is often used in the writing of lexical analysers so as to ensure robustness against unexpected input.

4. Modifications to the Report

(a) Add to the list of special-symbols in section 3:

```
        | "otherwise"
```

(b) Replace the production for case-statement by:

```
    case-statement =

        "case" expression "of"

        case-list-element { ";" case-list-element }

        [ "otherwise" statement { ";" statement } ] "end"
```

(c) Add the following text to the explanation of the semantics of the case statement in section 9.2.2.2:

> "If there is no constant in the case statement whose value is equal to the current value of the selector, then the group of statements between otherwise and end are executed. If the otherwise part does not occur, then programs which cause this to occur in execution are invalid."

5. Implications for variant records

The Working Group considers that no corresponding change should be made in the syntax of variant records.

6. Considerations taken into account

In recommending this syntax and semantics, the Working Group has considered many alternatives, including

(a) the use of alternative word-symbols, including else,
(b) other syntax constructions,
(c) what the 'undefined' actions might be, and
(d) whether the extension was needed and added to the power of the language.

NOTE

In the creation of a draft standard, the wording of the Revised Report may be altered, with consequent effects on the phrasing of this extension note. The syntax and semantics will not alter.

The full specification of all parameters to procedures and functions which are thenselves parameters was agreed on. Discussion of this topic was very influential and resulted in its inclusion in the third working draft of the BSI/ISO Pascal Standards document. Its description appears in #14, so we won't waste room here.

In July the UCSD Workshop referred important extensions to the Working Group, such as conformant array parameter bounds.

On August 24, Jorgen Steensgaard-Madsen had to resign as coordinator of the Working Group because he began spending a sabbatical year. Charles Fischer of the University of Wisconsin took his place to represent Univac implementations. Jim Miner and I are now coordinating the Working Group. The current topic of discussion is conformant ("dynamic") array parameters, which are important for building practical subprogram libraries for both numeric and non-numeric applications. - Andy and Jim

****

Dear Andy

12 June 1978

PASCAL test suite

Readers of PASCAL News will be aware of the standardization effort that is being undertaken in the UK under the auspices of the British Standards Institute. As part of that effort, I am collecting together (with

Arthur Sale and others) a suite of test programs designed to illustrate trouble spots in the language definition (and potentially in compilers). When the standardization is completed, it should be possible to use the suite to validate compilers, assess their performance or diagnostics as well as giving some indication as to how they match up to the standard.

Anybody who would like a copy of the tests or who would like to contribute to the tests should write to me. I am not publishing the tests at this juncture since they will change rapidly over the next year.

Yours sincerely

Brian Wichmann

B A Wichmann

Department of Industry
NATIONAL PHYSICAL LABORATORY
Teddington Middlesex TW11 0LW U.K.
Telex 262344 Telegrams Bushylab Teddington Telex
Telephone 01-977 3222 ext 3976

THE STANDARD
PROGRESS REPORT NUMBER 1
15-9-78

This report will necessarily be brief, since time spent writing the report is time that cannot be spent on the draft.

At the April meeting of DPS/13/4 it was decided that we should make an attempt at preparing a draft. Up until the April meeting our efforts had been largely directed towards identifying the problems rather than the solutions. Although production of the rough draft was rather behind schedule (largely due to examination marking by the university members of DPS /13/4) I was able to take a copy to LaJolla. This proved to be very valuable. An improved draft was presented at the September meeting of DPS/13/4 at which a number of alterations were agreed. These alterations are currently being made. When completed, this working document will be given to BSI for the necessary editorial and other processing before it is issued as a draft for public comment.

When I was in the USA this summer it was my belief that I could arrange for the draft BS to appear in Pascal News. This will not be possible, unfortunately. However, I will be submitting to Andy Mickel a copy of the working document which DPS/13/4 passes to BSI. The technical contents of this document will be the same as the draft unless any errors are detected and corrected by the BSI machinery.

The decision to prepare a draft for public comment does not mean that we have,or even believe that we have, resolved all the questions that people have concerning Pascal. We have prepared a draft because we believe that many issues have been resolved and that now is an opportune time to receive comments on what has been done.

It is my intention to send a commentary on the working document along with the working document, in an attempt to highlight those areas which currently are causing concern.

In the next issue of Pascal News I should be in a position to report on the situation within ISO.

A M ADDYMAN

AM Addyman

Convenor - DPS/13/4

SYSTEMS
ENGINEERING LABORATORIES

September 12, 1978

Mr. Andy Mickel, Editor
PASCAL News
University Computer Center
208 Southeast Union Street
University of Minnesota
Minneapolis, Minnesota 55455

Dear Andy:

I apologize for writing this letter so late. My only excuse is that I have been quite busy.

As you know by now, Andy, during the conference at the University of California, San Diego, I volunteered (read I was cajoled!) to act as coordinator for standards for the PASCAL User's Group at least until some formal arrangement was made by charter, elected officers and the like.

However, after conversations with Tony Addyman, we both noticed a severe overlap in his informal position with the User's Group and mine. We solved this quite easily. I told him he could have the whole thing! But, Tony was too smart for that. So here is what I offered to do:

(1) Act as North American liason for Tony's efforts in standardization and to generally aid him and the newsletter staff.

(2) Draw up a first-draft proposal for Program Interchange Standards (by January 1, 1978).

(3) To collect and standardize a more extensive set of syntax and semantic test programs for the standard PASCAL language and to propose a uniform way of classifying and organizing these tests (by April 1, 1979).

I volunteered to do the last item because it is one of the things I committed to do for my company. I know that both Arthur Sale and Brian Wichmann are extremely interested in this same effort, and I will work closely with them as well as any other Pascal User's Group members who wish to contribute.

I hope to be able to carry out these tasks in a timely manner.

Sincerely,

Rick

W. F. Shaw

WFS/esl

6901 WEST SUNRISE BLVD., FORT LAUDERDALE, FLORIDA 33313 (305) 587-2900

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455
(612) 373-4360

William F. Hanrahan, X3 Secretary
American National Standards Institute, SPARC
Suite 1200
1828 L Street NW
Washington, DC 20036

Wednesday 78/09/27.

Dear William,

I am writing to you regarding an article appearing in the 28 August issue of Computerworld on page 27 about the proposed ISO Pascal Standard. I only hope that this letter is not too late.

I am the coordinator of the international Pascal User's Group (PUG) which now numbers over 2700 members in 41 countries and 49 states. PUG produces the quarterly Pascal News, of which I am editor. Pascal News has overseen the rapid spread of Pascal simply by disseminating vast quantities of information (please see enclosures). American PUG members number about 2000.

I would like to point out that PUG has been in very close contact with the British Standards Institute DPS/13/4 group and its chairman, Tony Addyman, which are producing a draft Pascal Standard. PUG is fully aware of the activities of this British Pascal group, the Swedish Technical Committee on Pascal, the French AFCET Sub-group on Pascal, and the Pascal group within the German ACM.

The PUG membership (which certainly comprises the majority of Pascal enthusiasts) has consistently held the position that the control of the international effort be left in British hands.

The over 100 persons attending the Third Annual Computer Studies Symposium on Pascal at the University of Southampton, held in March, 1977, unanimously approved a motion that the Pascal Report be standardized with the semantics "tightened up" and with no extensions. Tony thus had the support he needed to undertake his effort.

Niklaus Wirth, the designer of Pascal, has given Tony Addyman his enthusiastic support and is providing technical assistance for the ISO standardization effort.

The recent Pascal Workshop held at the University of California, San Diego, was attended by representatives of over 15 computer companies having used Pascal for more than a year. That workshop unanimously agreed that every attempt be made to conform to the resulting BSI/ISO standard.

We are proud to say that the simplicity of Pascal which separates it from other languages has carried over into the standards activity undertaken so far.

In a couple of issues of Pascal News, people who supported a general standards effort naturally thought of turning to ANSI. It was pointed out, however, that the ANSI Pascal Standard should be one line which reads: "See the ISO Pascal Standard, document number X." just as the ISO standards for FORTRAN and COBOL are one-line entries (with ANSI's cooperation) which say "see the ANSI Standard."

Thus ANSI has the opportunity of reciprocating its respect with ISO. Pascal is a language with European origins, and the major work on standards has appropriately been left to Europeans. The savings in time, expense, and energy to ANSI or any proposed American "technical committee for Pascal" are obvious. We all don't want national variants for Pascal.

Thanks for your support,

Andy

Andy Mickel

american national standards committee

X3—computers and information processing
X4—office machines and supplies

operating under the procedures of the American National Standards Institute

NEWS RELEASE

October 23, 1978

For more information, contact:

C. A. Kachurik
202/466-2288

PASCAL PROGRAMMING LANGUAGE STANDARDS COMMITTEE FORMED BY X3

Washington, D.C. -- "Programming Language PASCAL" is the responsibility of a new committee under American National Standards Committee X3. Identified as Technical Committee X3J9, the initial task of the technical committee is to prepare a proposal for standardization of the PASCAL programming language and obtain approval of the proposal and program of work. Committee work will be aligned closely with the international standards subcommittee on PASCAL, as well as on-going work in the Federal Government, domestic professional societies, equipment manufacturers, and other interested organizations.

The committee is seeking active participation from users of PASCAL, as well as developers of the PASCAL language compilers. Interested persons and organization representatives are invited to contact C. A. Kachurik, 202-466-2288 at CBEMA/Standards, Washington, D.C. for further details.

X3J9 will be a part of the parent committee X3, which has overall responsibility for standards on computers and information processing. X3 currently has 29 technical committees and has completed some 60 standards published by the American National Standards Institute (ANSI), with as many more in various stages of development. The X3 administrative secretariat is the Computer and Business Equipment Manufacturers Association (CBEMA).

secretariat: Computer and Business Equipment Manufacturers Association
1828 L Street NW (Suite 1200), Washington DC 20036 Tel: 202/466-2299 CBEMA

american national standards committee

X3—computers and information processing
X4—office machines and supplies

operating under the procedures of the American National Standards Institute

NEWS RELEASE

November 10, 1978

For more information, contact:

C. A. Kachurik
202/466-2288

PASCAL PROGRAMMING LANGUAGE STANDARDS COMMITTEE MEETING SCHEDULED

Washington, D.C. -- Mr. Justin Walker will convene the inaugural meeting of the newly-formed X3 Technical Committee on Programming PASCAL, X3J9, Tuesday, December 19, 10:00 a.m. at the offices of the Computer and Business Equipment Manufacturers Association. The Association which serves as Secretariat to the X3 parent committee of X3J9 is located at 1828 L Street N.W., Suite 1200, Washington D.C. Mr. Walker, of the National Bureau of Standards, has extensive background in the PASCAL area and has developed several compilers.

The committee is seeking active participation from users of PASCAL, as well as developers of the PASCAL language compilers. The initial task is to prepare a proposal for standardization of the language and obtain approval of the proposal and program of work.

Interested persons and organization representatives are invited to attend, or contact C. A. Kachurik, Secretariat Staff, at 202/466-2288 for further details.

secretariat: Computer and Business Equipment Manufacturers Association
1828 L Street NW (Suite 1200), Washington DC 20036 Tel: 202/466-2299 CBEMA

GENERAL INFORMATION

Special note: We are pleased to print Scott Jameson's announcement (below) of the formation of the PUG Implementors Group. Given the wide variety of previous and current activities in the implementation of Pascal and extensions, this group will fill an important role in coordinating and furthering these efforts. Everyone is encouraged to read and respond to Scott's proposal.

As this is the first issue of Pascal News in this academic year, let us explain how this section is organized:

-- First, Reports of interest from the Implementors Group.

-- A CHECKLIST to be used as a guide to users, distributors, implementors and maintainers for reporting the status of Pascal implementations on various computer systems.

-- A PORTABLE PASCALs section reporting distribution information about kits used to produce Pascal compilers for real computer systems.

-- Information on PASCAL VARIANTS.

-- A FEATURE IMPLEMENTATION NOTES section describing implementation strategies and details of various Pascal features as suggestions to all the compiler implementation efforts underway. This section may be replaced by the Implementors Group's Reports.

-- A list of MACHINE DEPENDENT IMPLEMENTATIONS sorted by name of computer system, giving news of Pascal compilers for real machines.

-- And an INDEX to all the implementation information in current issues and back issues of Pascal News.

Note: It is not economically feasible for us to reprint all of the old information from previous issues. We therefore will provide references to back issues when we have received no new information. (Use the All-Purpose Coupon at the beginning of this issue for ordering back issues.) We will be very happy to print new information, or revisions of previous items, submitted by users, distributors, maintainers, or implementors. When appropriate please use the CHECKLIST form. We prefer dark camera-ready copy, single-spaced, with wide (18.5 cm) lines.

IMPLEMENTORS GROUP REPORT

HEWLETT PACKARD

DATA SYSTEMS • 11000 Wolfe Road, Cupertino, California 95014, Telephone 408-257-7000, TWX 910-338-0221

To: PASCAL NEWS Readers

One of the results of the UCSD Workshop on Extensions to Pascal was the decision that the Pascal User's Group would evolve in order to continue to meet the needs of its 2200+ members. The diverse interests of these members, ranging from first-time programmers to language designers, require PUG to structure itself so that it can better respond to everyone interested in Pascal. One proposal is that PUG form an 'Implementors Group', to provide a medium for communicating items of interest for those involved in developing Pascal compilers, and for those desiring further information regarding specific compilers or machine implementations.

The Implementors Group can serve the Pascal community in many ways. Some of the things we hope to do are:

- Publish a newsletter, aimed primarily toward the compiler developer. It will contain articles of interest to implementors, such as how to implement sets, structures of files, 'core' files, and the standard procedure 'dispose', as well as guidelines to ensure transportability and compatibility with other Pascal implementations. 'PASCAL NEWS' has provided this, but the Implementors Group Newsletter could provide extended and more specialized features.

- Provide a means for implementors to add to a 'validation suite', such as that mentioned in 'PASCAL NEWS' #12 (pages 52 & 54), or other set of Pascal test cases. A compiler writer could contribute a program that gave him fits, and see if other developers had solved that problem.

- Provide an information exchange for all persons interested in Pascal. The PUG offices are inundated with requests for information on a particular implementation, and the Implementors Group could serve as a clearinghouse, to channel these questions to the appropriate persons.

- Provide an organization to evaluate and decide on proposals for Pascal extensions. This may include the experimental language features suggested by the UCSD Workshop or 'conventionalized extensions' proposed by other persons.

- Provide a forum for implementors and users to interact with each other. This includes user's comments about a particular implementation, to give implementors a better feel for what users think of their compiler (many developers share the complaint that they don't get any feedback from users of their compilers, and have no feeling for the number of compilers in use or how successful they have been), as well as the checklists which implementors now provide for 'PASCAL NEWS'.

This list is not complete by any means, and we are looking for suggestions of other areas where the group can serve all Pascal users.

The logistics of this group still have to be worked out. I have volunteered to act as the group's coordinator, at least until a more formal arrangement is established. The membership will be open to all interested Pascal User's Group members, and a mailing list will be maintained so that all known implementors and interested persons receive the newsletters and other mailings. There will be no fees until we have a feeling on the number of people involved, and the cost can be determined.

Everyone who is interested in this group would have to be prepared to contribute in some form. This could include responding to queries regarding their particular implementation, and contributing to the newsletter. I can provide the clearinghouse function and forward inquiries to the right people, but I don't have the time or the knowledge to be able to answer

Implementation Notes

questions on all compilers. The same is true of the newsletter. It is for all implementors, and is an excellent vehicle to show off an elegant solution to a sticky problem, as well as a convenient means to communicate with other Pascal implementors.

Please send any comments,etc. or requests to be on the mailing list to me:

Scott K. Jameson
Hewlett-Packard
11000 Wolfe Road
Cupertino, CA 95014

(408) 257-7000

CHECKLIST

Pascal Implementations Checklist

0. DATE/VERSION
   (* Last checklist changes; version name or number, if any. *)

1. DISTRIBUTOR/IMPLEMENTOR/MAINTAINER
   (* Names, addresses, phone numbers. *)

2. MACHINE
   (* Manufacturer, model/series and equivalents. *)

3. SYSTEM CONFIGURATION
   (* operating system, minimum hardware, etc. *)

4. DISTRIBUTION
   (* cost, magnetic tape formats, etc. *)

5. DOCUMENTATION
   (* In form of supplement to <u>Pascal User Manual and Report</u>? Machine retrievable? *)

6. MAINTENANCE POLICY
   (* How long? Accept bug reports? Future development plans. *)

7. STANDARD
   (* Implements full standard? Why not? What is different? *)

8. MEASUREMENTS
   (* -compilation speed (in characters/sec. please; this is a meaningful measurement for compilation speed);
   -compilation space (memory required at compilation time);
   -execution speed;
   -execution space (the memory required at execution time; compactness of object code produced by the compiler);
   ** Try to compare these measurements to the other language processors on the machine, e.g., FORTRAN. *)

9. RELIABILITY
   (* stability of system (poor, moderate, good, excellent);
   how many sites are using it?
   when was the system first released to these sites? *)

10. DEVELOPMENT METHOD
    (* Compiler or interpreter? Developed from Pascal-P / hand-coded from scratch/bootstrapped/cross-compiled/etc.? What language? Length in source lines? Effort to implement in person-months? Previous experience of implementors? *)

11. LIBRARY SUPPORT
    (* Libraries of subprograms available? Facilities for external and FORTRAN (or other language) procedures available? Easily linked? Separate compilation available? Automatic copy of text from library into source program available? Symbolic dumps available? *)

Return to: Pascal Implementations
c/o Andy Mickel
University Computer Center: 227 EX
University of Minnesota
Minneapolis, MN 55455 USA

PORTABLE PASCALS

Pascal-P.

The most-widely used portable compiler for creating new Pascal Implementations is Pascal-P. Basically Pascal-P is distributed from three places in the form of a kit consisting of a magnetic tape and printed documentation.

Pascal-P is a compiler written in Pascal (almost 4000 lines) which generates symbolic code for a hypothetical stack machine called a "P-machine" because it is a low-level idealized architecture for Pascal. The symbolic code is thus called P-code.

On the magnetic tape are textfiles containing:

- a sample character set collating sequence. This file is also distributed as a listing to simplify character set conversion.
- the Pascal-P compiler in Pascal.
- a P-code assembler/interpreter written in Pascal which is intended to document how to write an interpreter in an existing language on the target computer system.
- a Pascal-P compiler in P-code. In other words, the result of compiling the Pascal-P compiler on itself.

The person implementing Pascal has several choices. If there is no access to a working Pascal compiler on another machine, the implementor orders a Pascal-P kit already configured to the target machine. Configured compilers have constants inserted in them to specify, for example, the size of each simple data type. These configuration parameters are given by the implementor on the Pascal-P order form. (See below.)

After receiving the kit, the implementor can write an interpreter for P-code in another language (usually takes about one person-month), and thus immediately has access to a Pascal compiler running interpretively by using the P-code version of the compiler included in the kit.

To produce a real Pascal compiler for the target machine then requires editing of the Pascal-P compiler written in Pascal to produce code for the target machine (instead of the P-machine). After recompiling, a Pascal compiler exists in the code of the target machine.

If the implementor initially has access to a working Pascal compiler on another machine, the step of writing a P-code interpreter can be omitted.

Facts about the Pascal-P compiler:

- The current version is called Pascal-P4 and is distributed with a copy of Pascal-P3 (which is of interest to previous recipients of Pascal-P2).

- Pascal-P4 represents a major improvement over earlier Pascal-P versions because it removes data-type-alignment restrictions, is more efficient, includes runtime tests, and is a more complete implementation of Pascal.
- Pascal-P2 was developed from a phase in the stepwise refinement of Urs Ammann's Pascal-6000 compiler in 1974 by K. V. Nori, Urs Ammann, K. Jensen, and H. H. Nageli. Subsequent improvements were done by Christian Jacobi.
- Reliability of Pascal-P4 has been fairly good. As of Spring, 1977, it was distributed to 106 sites by George Richmond (from Colorado), to 37 sites by Chris Jacobi (from Switzerland), and to more than a dozen sites by Carroll Morgan (from Australia).
- The is no promise of maintenance for Pascal-P. P4 is the final version produced at Zuerich. We do print reports of bugs (and fixes) in P4. Over 25 fixes were printed last year in Pascal News issues #11 (pp 70-71) and #12 (pp 56-57). More are printed below.
- Documentation for Pascal-P4 consists of a 65-page report entitled The Pascal <P> Compiler: Implementation Notes (Revised Edition) July, 1976. (A 24-page correction list to the original December, 1974, edition is also available.)
- Pascal-P4 does not adhere strictly to Standard Pascal (the User Manual and Report). Among the differences are:

1. nil is implemented 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. The standard comment delimiters { and } are not supported.

3. The following standard predeclared identifiers are not provided: maxint, text, round, page, and dispose. Further, the following standard predeclared identifiers are recognized but are flagged as errors: reset, rewrite, pack, and unpack.

4. The program heading is not required by P4.

5. Non-discriminated variant records are not supported.

6. The compiler does not allow a ";" before the "end" in a record type. (See the P4 bug reports in Pascal News #12 (pp 56-57) for a fix.)

7. None of the following file-related features are supported:
   -- Declaration of file types, variables, and parameters.
   -- The standard predeclared type text, and standard procedures reset, rewrite, and page.
   -- The requirement by the standard that the standard files input and output appear in the program heading if they are used.
   -- Access to non-text files using read and write.
   -- Output of Boolean expressions, or output of real expressions in fixed-point form with write.

8. Formal-procedures and formal-functions are not supported.

9. Set constructors containing the subrange notation (e.g., ['0'..'9']) are not supported.

10. "Non-local" goto statements are not supported.

Pascal-P can be ordered from three places (write for prices and order forms).

In Europe, Asia, and Africa, order from:

Christian Jacobi
Institut fuer Informatik
E.T.H. Zentrum
CH-8092 Zuerich
Switzerland
Phone: 41/1-32 62 11 x2217

In North and South America, order from:

Pascal Distribution
c/o Steve Winograd
Computing Center: 3645 Marine Street
University of Colorado
Boulder, CO 80309
USA
Phone: 303/492-8131

In Australasia order from:

Tony Gerber
Basser Dept. of Computer Science
University of Sydney
Sydney, NSW 2006
Australia
Phone: 61/2-692 3216

Pascal P4 -- Bug Reports

On 78/06/09, Ted C. Park, Systems Development, Medical Data Consultants, 1894 Commercenter West - Suite 302, San Bernardino, CA 92408 (714) 825-2683, reported:

"I just came across two more bugs in the PASCAL-P4 compiler. FUNCTION EQUALBOUNDS contains an obvious error:

```
        replace P.136 with   GETBOUNDS(FSP2,LMIN2,LMAX2);
```

PROCEDURE GEN2T is used for (among other things) generating 'CHK' instructions. The fix causes the width of the 'P' field to be 3 or 8 as needed. Without the fix the lower limits of arrays must be less than four digits long!

```
        replace P.262 with   WRITELN(PRR,FP1:3+ORD(ABS(FP1)>99)*5,FP2:8);
```

(*Thanks Ted!*)

Pascal Trunk Compiler

The trunk compiler is the machine-independent part (e.g., syntax analysis and error recovery) of a Pascal compiler in which the code generation has to be inserted in a certain number of empty procedures. We have received no new information on the Trunk compiler since that which we published last year in Pascal News issue #9-10 (p 62).

Pascal J

Pascal-J is a compiler which translates Pascal to the intermediate language Janus, a totally portable "mobile programming system" -- even to the point of defining its own character set! Janus in turn is macro-processed via Stage2 which is implemented in standard Fortran. We have received no new information on Pascal-J since that which we published last year in Pascal News issue #9-10 (p-62).

PASCAL VARIANTS

Pascal-S

Pascal-S is a subset of Pascal developed by Niklaus Wirth. We have received no new information on Pascal-S since that which we published last year in Pascal News issues #9-10 (p 63) and #11 (p 72).

Concurrent Pascal

A portable pair of Pascal compilers was implemented by Per Brinch Hansen and Al Hartmann at Cal Tech in 1974-1975 for the PDP 11/45. The system consists of a "Sequential Pascal" compiler, a "Concurrent Pascal" compiler (used for writing operating systems and other concurrent programs), and a "kernel" or machine dependent set of run-time routines written in assembler. The project at Cal Tech centered around writing a one-user operating system called SOLO in Concurrent Pascal. Both compilers are written in Sequential Pascal.

In 1975-1976 the system was distributed widely (252 sites) and led to the development of a machine-independent version with a different kernel.

The distribution tapes ($50) and documentation ($10) can be ordered from:

Pascal Distribution
c/o Steve Winograd
Computing Center: 3645 Marine St.
University of Colorado
Boulder, CO 80309
USA
Phone: 303/492-8131

Publications about Concurrent Pascal include:

(1) "The programming language Concurrent Pascal", in the June, 1975, IEEE Transactions on Software Engineering 1:2, by Brinch Hansen.
(2) A guest editorial and four articles by Brinch Hansen in the April-June, 1976, issue of Software - Practice and Experience 6, pp 139-205. The articles are entitled:
"The Solo Operating System: A Concurrent Program"
"The Solo Operating System: Job Interface"
"The Solo Operating System: Procedures, Monitors, and Classes"
"Disk Scheduling at Compile Time"
(3) The book Operating Systems Principles by Per Brinch Hansen, Prentice Hall, 1973.
(4) An article "Experience with Modular Concurrent Programming" in the March, 1977, IEEE Transactions on Software Engineering 3:2, by Brinch Hansen.
(5) A Concurrent Pascal Compiler for Minicomputers by Al Hartmann, Springer-Verlag: Lecture Notes in Computer Science, Volume 50, 1977.
(6) The new book The Architecture of Concurrent Programs by Brinch Hansen, Prentice-Hall, 1977.

COMPUTER SCIENCE DEPARTMENT
SALVATORI COMPUTER SCIENCE CENTER
(213) 741-5501

October 1, 1978

Dear Concurrent Pascal User,

It is now 3 years ago since the Concurrent Pascal compiler and the Solo Operating System were first distributed. Since then the system has been moved to several computers and used for a variety of purposes.

Some users (but not all) have briefly reported on their usage of Concurrent Pascal in the Pascal Newsletter. I am now trying to get a more complete overview of the current use of the system.

If you are using Concurrent Pascal or Solo then please send me a letter. I would like to know which computer you are using, how hard it was to move the system to that machine, how reliable the software has been, what applications the system is being used for, and any other comments you may have. I would also like to know if you have published any papers about your experience.

Andy Mickel and I plan to publish these letters in the Pascal News. If I receive your letter before February 28, 1979 it will be included in the newsletter.

I look forward to hearing from you.

Yours sincerely,

Per Brinch Hansen

UNIVERSITY OF SOUTHERN CALIFORNIA, UNIVERSITY PARK, LOS ANGELES, CALIFORNIA 90007

UMIST
The University of Manchester Institute of Science and Technology
PO Box 88, Manchester M60 1QD
Telephone 061-236 3311

Department of Computation

27th April 1978

Dear Andy,

We have moved Brinch Hansen's SOLO system on to our 40K CTL Modular One computer. We have found the system to be very reliable and the few bugs that have been found have been simple to fix. Our main interest is in Concurrent Pascal which we are using as a tool for our work on the development of programming methods for multiprograms.

Due to the inhospitable architecture of the Modular One our system runs at only a fifth of the speed of the original PDP-11/45 implementation. Work is under way to improve the speed by the utilisation of a second processor and a fixed head disc. A simple multi-access system is also being considered.

The transportation of SOLO was very straight forward and was accomplished in about eight months by two undergraduate students and one lecturer working part-time. Further details of the move are contained in Malcolm Powell's report [1].

We are interested in exchanging information and programs with other users or potential users of SOLO or Concurrent Pascal.

Yours sincerely,

Derek Coleman

Derek Coleman
Lecturer in Computation

[1] M.S. Powell. Experience of Moving and Using the SOLO Operating System, Computation Department, UMIST

Modula

Modula is a small language for dedicated computer systems and process control applications on small machines, developed by Niklaus Wirth and co-workers in 1975-76. It is conceptually cleaner than Concurrent Pascal in many respects. The Modula language definition provides for machine-dependent facilities for interacting with asynchronous devices. Modula is still experimental and the implementors in Zurich have insisted there are no distribution arrangements. Other implementations are complete or underway. See Pascal News #11 (p 74) for details of the University of York PDP-11 compiler. Also, on 78/10/27, Gerd Blanke (Postbox 5107; D-6236 ESCHBORN Germany; phone (06198) 32448) wrote "MODULA will be running on a ZILOG MCS with 64K under RIO near the end of this year!"

Published material on Modula includes:

(1) "Modula: A Language for Modular Multiprogramming", Software - Practice and Experience 7 (1977), pages 3-35, by Niklaus Wirth.
(2) "The Use of Modula", same as (1), pages 37-65, by Niklaus Wirth.
(3) "Design and Implementation of Modula", same as (1), pages 67-84, by Niklaus Wirth.
(4) "Toward a Discipline of Real-Time Programming", Communications of the ACM 20:8 (August, 1977), pages 577-583, by Niklaus Wirth.
(5) "Experience with the programming language MODULA", University of York - Dept. of Computer Science (June, 1977), by J. Holden and I. C. Wand.

References (1) through (3) received very interesting reviews in Computing Reviews 18 (November, 1977), #32217, #32218, and #32219.

FEATURE IMPLEMENTATION NOTES

IMPLEMENTATION NOTE

Implementation of INPUT and OUTPUT

Arthur Sale and Judy Bishop

PROBLEM
It has come to our attention that there is a problem with the implementation of the pre-defined files *input* and *output*. What follows refers only to *output*, as it is easier to demonstrate the effects on an output file, but applies equally to the file *input*.

The problem turns on two of Pascal's Achilles' heels: the elision of a file-name in *read* and *write* and the resulting default, and the singular program parameter part and its interaction with pre-defined names. The situation can be summed up by two questions, to each of which there are two reasonable answers.

QUESTION 1 : Where do default writes go?
Does *write(x)* write on the default file named *output* (and pre-defined), or on the lexically innermost definition of a file named *output*?
Answer A : x is always printed on the pre-defined file, whatever redefinitions of the name *output* may have taken place.
Answer B : the symbol table is searched for *output* and the write is attempted on the innermost occurrence of it.

QUESTION 2 : At what level is *output* defined?
Is the pre-defined file *output* regarded as declared at the level of the program block (level 0) or in a lexically enclosing block (level -1)?
Answer C : the file is regarded as being at the level of the program block, thereby prohibiting a synonymously named file at that level.
Answer D : the file is regarded as being in a block enclosing the program, so that the name can be redefined in the program block.

WHERE DO DEFAULT WRITES GO?
The Tasmania B6700 compiler and the AAEC IBM compiler transmit default information always to the pre-defined file *output*, and it seems likely that the CDC-6000 compiler, the ICL compilers, and most Pascal-P derivatives do the same. These indicate that Answer A is currently predominant.

What does the Report and User Manual say? The Report (#12.3) defines
    write(x)
as equivalent to
    write(output,x)
which makes one think of Answer B: the elision of the file-name is to be handled by a macro-expansion. However, on reading the User Manual (p61) and earlier in the Report (p161) we find that *output* is described as a program parameter which is assumed by default if the filename is omitted. In other words: the pre-defined file, and Answer A.

On balance, therefore, the predominant Answer A seems to be approved by the User Manual. It can be argued that this is abstractly best, for if we have to have any defaults in Pascal (and we've got these few), then they ought to be as simple as possible.

The following is a test program to exercise your compiler and test its performance on this question:

```
program question1(output);
   procedure inner;
      var output : text;
   begin
      writeln('WRITING ON DEFAULT FILE');
      writeln(output,'WRITING ON LOCAL FILE')
   end;
begin
   writeln(output,'TEST OF QUESTION 1');
   inner;
   writeln('RAN')
end.
```

AT WHAT LEVEL IS *OUTPUT* DEFINED?
Output, in common with other pre-defined names, can be regarded as pre-declared in a lexically enclosing scope, thus allowing its redefinition in the program block. This is asserted by the Report (p161), and is the current interpretation given by the Tasmania B6700 compiler: Answer D.

The alternative, sanctioned by the User Manual (p91) in the CDC-specific section, says that these files are *implicitly* declared in the program block (not pre-declared). In CDC-6000 Pascal therefore one may not define any object with the name *output* at the program level. The AAEC compiler is similar, thus giving Answer C.

So both answers find some support, and both are in use. Which is better? Experience of one of us (AHJS) indicates that perhaps Answer C is best: pre-declared at the program level. This experience arises from a number of apparent 'bug reports' received from afar which, when traced, turn out to be derived from a user attempt to redefine the output file by declaring a hiding occurrence of the name *output* at the program level. If Answer C had been adopted in the Tasmania B6700 compiler, these would have been detected as illegal by the compiler, and other name choices would have been forced on the users.

It is also possible, but inconclusive, to argue from analogy. Focussing on the analogy with other pre-defined identifiers, such as *abs* and *true*, then it seems consistent to argue that the definition of *output* should also enclose the program block. But, of course, these two files (*input* and *output*) are the only two var objects which are pre-defined, so perhaps they should be special. This view leading to implicit declaration in the program block, is supported by the analogy with all other file names mentioned in the program parameter part which must have a declaration in the program block (at least in the CDC-6000 implementation; others allow more freedom).

If then Answer C is more attractive, the Tasmania B6700 compiler should be changed. In this case however, we shall wait until the draft standard for Pascal resolves the issue. The following test program will show what your compiler does:

```
program question2(output);
    var output : integer;
    { if this compiles, you've probably got Answer D }
begin
    output := 1;
end.
```

A DEVIANT IMPLEMENTATION

One implementation, which shall remain nameless as a fitting punishment, lies outside the permitted limits of the Report and User Manual by using a subtle change. In this implementation, elision of the file-name causes the write to take place on an un-named pre-defined file. This has the result that

    write(output,x)

fails to compile unless another file is declared with the name *output*, and that the question of the default file's scope does not arise (because you can't rename an un-named file). It requires additional tests to distinguish this case from an implementation that answers A and C, and it may give rise to confusion amongst users.

BRIEF ADVICE

To Pascal users:

(1) Do not use the identifiers *input* or *output* for anything other than the pre-defined files that you don't need to declare.
(2) Preferably do not leave the file-names out of reads and writes, but put them in explicitly as a good programming practice.

To implementors:

Please modify any implementation plans to be consistent with majority opinion in Answer A, and watch for more information on Question 2.

To language designers:

(1) Future languages should make it mandatory for compilers to inform users of any names they hide under scope rules, if such exist. The extended searches are only necessary at declaration points.
(2) Defaults of any kind should be avoided.

1978 June 13
(Revised-1978 August 1)

Arthur Sale
University of Tasmania
Judy Bishop
University of the Witwatersrand

1978 April
A.H.J.Sale

IMPROVED CHECKING OF COMMENTS

As is well-known, comments of the PASCAL kind have a severe disadvantage in that if a closing marker is omitted or mis-keyed, intervening source text will be treated as commentary until a closing marker is found for a later comment. Since such errors do not give rise to syntax errors, they may remain undetected in source text for a long time. This feature is exacerbated in PASCAL by allowing comments to continue over line-boundaries, and highlighted by PASCAL's otherwise good compile-time error-detection.

In Burroughs B6700/B7700 PASCAL (University of Tasmania compiler), the problem this creates for programmers (especially learners) has been alleviated by issuing warnings if a semi-colon is detected within a comment, as this is very likely to be the result of an error. Very few erroneous comments remain undetected, and the change in the lexical analyser is very simple. *This suggestion is commended to other implementors and maintainers.*

People who use the comment facility to suppress source text compilation (debug code; superceded text) may be annoyed by the many warning messages. They can then suppressed by our compiler option WARNINGS; but better still would be to realise that this is a misuse of comments and hardly likely to enhance readability!

Many Algol 60 compilers have included similar checks in their handling of the singularly nasty end-comment in that language; the experience is generalizable to PASCAL too.

It would be possible to issue warnings for other symbols encountered in comments, for example a comment opening marker, and this would marginally improve the detection probability of these errors. We judged such extension as not worth the effort, especially since both {} and (* *) comments are permitted in our PASCAL, which would require quite complex checks.

Sample output for error:

```
            {this comment is unclosed

             count := 0;

W A R N I N G : DISCOVERED ";" IN COMMENT.  DID YOU FORGET TO CLOSE A COMMENT?

            {this closing marker will match the first one}
```

Lazy I/O

(* The "Lazy I/O" scheme has apparently been invented several times. The earliest implementation of which we are aware is in the Berkeley PDP-11 UNIX compiler. This was discussed in some detail at the UCSD Workshop in July. The consensus there seemed to be that Lazy I/O is the best solution anyone has yet proposed, even though it may be somewhat less efficient than other approaches in terms of execution time.
On October 21, James Saxe and Andy Hisgen added a note written to Andy Mickel which said: "By the way, the lazy evaluation idea was not cribbed from Berkeley UNIX Pascal, as you have suggested, but was developed here independently. We are, however, glad to see that there are other people around who do not feel compelled to introduce unnecessary changes to the semantics of Pascal at every opportunity." - Jim Miner *)

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213
August 4, 1978

Subject: LAZY EVALUATION OF THE FILE BUFFER FOR INTERACTIVE I/O

Dear Andy,

A frequently occurring difficulty in Pascal programming, and one which is particularly puzzling to the novice, arises from the effect of the file lookahead buffer on interactive I/O. Specifically, let TTY be a TEXT (FILE OF CHAR) variable associated with the input stream from the user's terminal and let TTYOUTPUT be a TEXT variable associated with the output stream to the terminal. Now, consider the following program fragment:

```
        . . .
1       ReadLn (tty, nplayers);
2       WriteLn (ttyoutput, 'Number of marbles = ?');
3       ReadLn (tty, nmarbles);
        . . .
```

Under many Pascal implementations, this fragment will fail to work as intended because the READLN in line 1 will not complete until the lookahead buffer, TTY↑, has been filled with a character (presumably the first digit of NMARBLES) from the terminal. The user, meanwhile, will not supply this character until he has been prompted by line 2, which of course cannot happen until line 1 has finished execution. Attempted "solutions" to this problem include

- Use of special user-defined procedures for the terminal which read a single real or integer *after* doing a READLN. (This approach, of course, is not very useful for programs that do character input.)

- Altering, in various ways, the *semantics* of file input when the run-time system "knows" that the file being read happens to really be the terminal (*e.g.*, making EOLN(TTY) be FALSE and TTY↑ be ' ' after each READLN(TTY) regardless of the contents of the following line of input. Note that empty input lines will no longer be reliably detected and may "hang" the terminal.).

- Introducing a new file type for interactive devices, with slightly different semantics from those for TEXT files.

We maintain that all these kludges are *completely unnecessary.* A Pascal compiler and run-time system can be made to support interactive I/O in a perfectly natural manner *without any deviation from the semantics laid out in the report.* This can be achieved by "lazy evaluation" of the file lookahead buffer for the terminal, that is, the practice of never filling TTY↑ until it is actually used.

To describe this more precisely, let ACTUALGET be a procedure having the effect that GET has in most implementations. That is,

```
    ActualGet (tty);
```

has the effect of grabbing one character of terminal input from the operating system and sticking that character in TTY↑. We introduce a new Boolean variable, TTYBFULL, visible only to the run-time system, which, as we shall see, shall be TRUE iff the "current" character in the file TTY has actually been read from the terminal. The action of

```
    Get (tty);
```

is now precisely defined as

```
    IF ttybfull
        THEN ttybfull := FALSE
        ELSE ActualGet (tty);
```

Whenever the programmer *explicitly* does something that requires lookahead (assigns to TTY↑, calls EOLN(TTY) or EOF(TTY), uses TTY↑ in an expression, or passes TTY↑ as a value parameter), the run-time system, behind the programmer's back, forces the lookahead buffer full by doing

```
    IF NOT ttybfull THEN
        BEGIN
        ActualGet (tty);
        ttybfull := TRUE;
        END;
```

When TTY is RESET for input form the terminal an ACTUALGET is *not* done, but TTYBFULL is initialized to FALSE. The call

```
    Read (tty, c);   (* where C is a variable of type CHAR *)
```

continues to be equivalent (as specified in the *Report*) to

```
    c := tty↑;
    Get (tty);
```

The procedure ACTUALGET, like the variable TTYBFULL, is directly accessible only to the run-time system and not to the programmer.

Careful consideration of the rules described above will show that they result in *exactly* the semantics described in the *Pascal Report.* The only difference between this and other implementations is that the terminal will not "hang" in the manner described in the opening paragraph. This conformity with the semantics of the *Report* has several advantages:

- Conformity to Standard Pascal improves the prospects for software portability.

- *Any* program which works correctly under a correct implementation of Standard Pascal will continue to work, and will give the same output (given the same input), under the implementation described above.

- Since the semantics of disk file I/O and terminal I/O continue to be identical, programs which use input from one source can be *easily* modified (say, for debugging) to take input from the other. Also programs which postpone until run-time the decision whether to take input from a disk file or from the terminal can be written without needless duplication of code.

Let us emphasize again that even programs which make use of the lookahead buffer will work in the manner defined by the *Report*, because any program action which actually requires knowledge of the lookahead character will demand that character from the terminal before it can continue. Of course it is the programmer's responsibility to prompt the user for this input, but since the programmer knows that this information is required at a particular point in the program, he should have no trouble remembering to prompt for it. Consider, for example, the following program fragment, which prompts the user for an integer but allows him to just type a carriage return if he wants the default value (shown in brackets by the program):

```
        . . .
1       WriteLn (tty, 'Number of runs [10] : ');
2       IF Eoln (tty) THEN
3           BEGIN
4           nruns := 10;
5           ReadLn (tty);
6           END
7       ELSE
8           ReadLn (tty, nruns);
        . . .
```

In line 2, the programmer does an explicit lookahead at the first character on the line to determine whether it is the line delimiter (*i.e.*, whether the line is empty). In this case, the lookahead character will be demanded by the run-time system before the expression EOLN(TTY) can be evaluated. However, the prompt for this input will have already been supplied by line 1.

In closing, we should take note of some tricky aspects of the lazy evaluation technique which might at first escape the notice of the prospective implementor. First, lazy evaluation of the lookahead buffer should be performed on *all* TEXT files, since it is not necessarily possible to determine at compilation which of these will be associated with the terminal (for example, TTY may be passed as an actual procedure parameter). Second, enforcing correct semantics can be very tricky in cases where the *lookahead buffer* (TTY↑) is passed as a VAR parameter [Our approach at CMU is to force the buffer full *once* at the time of function or procedure invocation and to leave the user on his own thereafter. Since passing TTY rather than TTY↑ guarantees the expected semantics, we feel that this approach does not make impositions on the reasonable user. An alternative approach would be to disable lazy evaluation for the duration of the invocation.]. In spite of these difficulties, however, we believe that the lazy evaluation approach to the interactive I/O problem is substantially superior to the other mechanisms we have seen.

Yours truly,

James B. Saxe

James B. Saxe

Andy Hisgen

Andy Hisgen

MACHINE-DEPENDENT IMPLEMENTATIONS

Alpha Microsystems AM-11

See DEC LSI-11 UCSD.

Altair 680b

See Motorola 6800 St. Paul.

Altair 8800

See Intel 8080.

Altos ACS-8000

It has been reported that Altos Computer Systems; 2378b Walsh Ave.; Santa Clara, CA 95050; 408/244-5766 offers a Zilog Z-80 based microcomputer which supports CP/M and Pascal, but we have received no information from Altos.

Amdahl 470

See also IBM 360/370.
It has been reported that the IBM 360/370 AAEC as well as the Vancouver systems are running on an Amdahl 470.

Andromeda 11/B

See DEC LSI-11.

Apple II

See MOS Technology 6502.

BESM-6 Moscow

0. DATE/VERSION. 78/9/21.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. S. Pirin; Moscow Computer Center; USSR Academic Sciences; Moscow, R.S.S.R.; U.S.S.R.; (* No phone number repotted *)

2. MACHINE. BESM-6.

3. SYSTEM CONFIGURATION. (* No information reported. *)

4. DISTRIBUTION. (* No information reported. *)

5. DOCUMENTATION. (* No information reported. *)

6. MAINTENANCE. (* No information reported. *)

7. STANDARD. (* No information provided. *)

8. MEASUREMENTS. (* No information reported. *)

9. RELIABILITY. (* No information reported. *)

10. DEVELOPMENT METHOD. (* Reported that project has been underway (or possibly complete?) for some time. *)

11. LIBRARY SUPPORT. (* No information reported. *)

BTI 8000

It has been reported that the BTI 8000, a 32 bit multiprocessor system offered by BTI Computer Systems; 870 W Maude Ave.; Sunnyvale, CA 94086; 408/733-1122, includes a Pascal compiler bundled with the hardware and that the system software is written in "Pascal-X", an extended version of Pascal; but we have received no information from BTI.

Burroughs B1700 Zurich

We have received no new information on this implementation since that which we published last year in Pascal News issues: #9-10: 73. #12: 57-58.

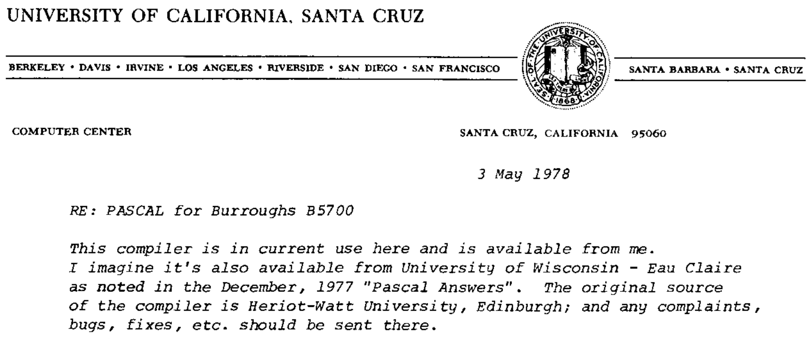
Burroughs B1800

See Burroughs B1700 Zurich.

Burroughs B4700 Fredonia

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 73.

Burroughs B5700 Edinburgh



UNIVERSITY OF CALIFORNIA, SANTA CRUZ

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO | SANTA BARBARA • SANTA CRUZ

COMPUTER CENTER

SANTA CRUZ, CALIFORNIA 95060

3 May 1978

RE: PASCAL for Burroughs B5700

This compiler is in current use here and is available from me.
I imagine it's also available from University of Wisconsin - Eau Claire
as noted in the December, 1977 "Pascal Answers". The original source
of the compiler is Heriot-Watt University, Edinburgh; and any complaints,
bugs, fixes, etc. should be sent there.

No charge if the requester sends a tape.

James H. Haynes
Associate Development Engineer

Burroughs B6700 Helsinki

According to Antti Salava (* 78/10/18 *): "I'm not working with Pascal nowadays. A year ago I left the University of Helsinki, where I was implementing Pascal-HB compiler on the Burroughs B6700. It's been running now a couple of years without any fatal crashes. We wrote a report on our compiler, too. Hasn't anybody noticed it? It's this: Hannu Erkio, Jorma Sajaniemi, Antti Salava; "An Implementation of Pascal on the Burroughs B6700"; Department of Computer Science; University of Helsinki; Report A-1977-1. Copies may be ordered from: Department of Computer Science; University of Helsinki; Toolonkatu 11; SF-00330 Helsinki 10, Finland.

Burroughs B6700 San Diego

We have received no new information on this implementation since that which we published last year in Pascal News issues: #9-10: 74. #11: 81 except that we have received a copy of the multi-page machine-retrievable installation notes that come with the system.

Burroughs B6700/7700 (Tasmania)

0. DATE/VERSION. Checklist has not been updated since 78/03.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. A.H.J. Sale; Pascal Support; Dept. of Information Science; University of Tasmania; Box 252C G.P.O.; Hobart, Tasmania 7001 Australia; STD 002 23-0561 x435.

2. MACHINE. Burroughs Model III B6700, B7700.

3. SYSTEM CONFIGURATION. Burroughs MCP version II.8 (with few (minor) local mods). Minimal system to operate not known, but unlikely to be any B6700 that small--storage demands are low, and little else is critical.

4. DISTRIBUTION. Both 7- and 9- track magnetic tapes available. Annual fee of $100 (Australian) is charged to cover mailing, processing, and maintance costs, payable to "The University of Tasmania".

5. DOCUMENTATION. Available documentation: Report R77-1: Supplement to Pascal User Manual and Report ; Report R77-3: Reference Manual similar to B6700 ALGOL's; A Pascal Language card; and A Pascal System card. (* Not known if this documentation is machine retrievable. *)

6. MAINTENANCE. To be maintained for teaching use within the University as well as larger aims. Reported bugs will be fixed as soon as possible, with patch notices to users. Duration of support not yet determined; several other developments are also pending. Each installation will be issued a supply of FTR-forms similar to those used by Burroughs for use in corresponding with us, and we will attempt to do a professional job in maintenance of the system.

The compiler has been stable in code for some time, reflecting its basic integrity. However, new features are added from time to time, and notified to users as patches or as a new version release. The department accepts FTR notices, and will attempt to fix those which warrant such attention. Some modifications have taken place as a result of user feedback. The compiler was especially designed so as not to generate dangerous code to the MCP, and no system crashes have been attributed to it since the first few months of testing, and then only three.

7. STANDARD.
Restrictions: Program heading: reserved word program is synonymous with procedure; no parameters (files) are permitted after the program heading. Reason: CDC anachronism of no utility in our installation, and likely to be confusing. Set constructor of form A..B not implemented. Reason: future plan. FORTRAN control character on print line not implemented. Reason: a ridiculous feature to standardize. Full Pascal I/O not implemented. Reason: future plans. Present I/O scheme is like Pascal-1.
Extensions: otherwise in case statement. Various reserved words, character set transliterations. Burroughs comment facility. File attributes in declaration. Format declarations. Extensive Burroughs-compatible compiler options. (Pascal control comment option mode not implemented).

8. MEASUREMENTS.
compiles about 20% slower than FORTRAN or ALGOL, but in about 2/3 of their space (for test programs about 4-5 K words on average instead of 8-10K). Elapsed compilation times similar, though Pascal slower. Speed should be improved by eventual tuning.
executes at same speed as FORTRAN and ALGOL (code is very similar and optimal) and takes generally longer elapsed residence time primarily due to MCP intervention to create new segments for record structures (not present in FORTRAN/ALGOL). Elapsed residence times about 20% greater than equivalent ALGOL.

9. RELIABILITY. Excellent. Only one system crash during testing attributed to Pascal. Compiler now in use at 3 sites. True compiler has been in use since 76/10. First released to outside sites in 77/4.

10. DEVELOPMENT METHOD. Compiler which generates B6700 code-files which are directly executed by the B6700 with MCP. Written entirely in B6700 ALGOL. Hand-coded using Pascal-P as a guide/model. All other paths offered much more difficulty due to special nature of machine/system. Person-month details not kept, and project proceeds in fits and starts as teaching intervenes. Project has thus far been limited to two people: Prof. A.H.J. Sale and R.A. Freak (Support programmer).

11. LIBRARY SUPPORT. There is as yet no BINDINFO in the code-file so that it is not possible to link Pascal to modules compiled by other language processors, but the system contains an extended set of predefined mathematical functions.

CDC 2550

See CDC Cyber 18 La Jolla.

CDC 6000, Cyber 70, 170 Bethlethem, PA

We have received no new information on this implementation since that which we published last year in Pascal News issue: #11: 82.

CDC 6000, Cyber 70, Cyber 170 (Zurich)

0. DATE/VERSION. Pascal 6000 Release 3; 78/11/15.

Wally Wedel
Computation Center
University of Texas-Austin
Austin, TX 78712 USA
512/ 472-3242
-(Australia, New Zealand, or Oceania)
Tony Gerber
Basser Dept. of Computer Science
University of Sydney
Sydney, N.S.W. 2006
Australia
61 / 2-692 3216

Switzerland
01/ 32 62 11

Maintainer:
John P. Strait / Andy Mickel
University Computer Center
227 Ex

University of Minnesota
Minneapolis, MN 55455
USA
612/ 376-7290

* Arrangements are underway to have the implementor of the CDC 7600, Cyber 176 run-time system take over distribution for Europe, Asia, and Africa from the original implementor.

2. MACHINE. Control Data 6000 series, Cyber 70 series, and Cyber 170 series.

3. SYSTEM CONFIGURATION. Minimum central memory-49K words. Operates under Scope 3.4, Kronos 2.1, NOS/1.3, and NOS/BE 1.

4. DISTRIBUTION. Tape format is Scope internal binary 7/9track, unlabelled, 800 bpi. Specify: person responsible for maintaining the system, your hardware, operating system, and character set (ASCII or Scientific, 63 or 64). Distribution includes machine-retrievable source and object decks, installation notes, and software tools. Arrangements for distribution (cost, etc.) for the new release have not yet been finalized. Contact the distributor in your area in further information.

5. DOCUMENTATION. Machine-retrievable supplement to Pascal User Manual and Report. Documentation of library-support package is available with Release 3.

6. MAINTENANCE. Will accept bug reports at Minnesota for forseeable future.

7. STANDARD. Nearly full standard. [Restrictions include: standard procedures and functions cannot be passed as actual parameters; file of file is not allowed.] [Extensions include: additional predefined procedures and functions; segmented files, conformant array parameters, otherwise in case statement, variable initialization facility (value), and text-inclusion facility for source libraries.]

8. MEASUREMENTS.
Compilation speed: 10800/5800 characters per second on a Cyber 74/Cyber 172;
Compilation size: 40K (octal) words for small programs; 57K for self-compilation.
Execution speed: self-compiles in 65/120 seconds.
Execution size: binaries can be as small as 2.4K, compared with Fortran minimum of over 10K.

9. RELIABILITY. Unknown, as this is a new release. However, Release 2 was very reliable and was in use at over 300 known sites. First version of this compiler was operational in late 1970. The present version was first released in May 1974. A pre-release version of release 3 was tested at 10 sites for up to 5 months prior to the official release.

10. DEVELOPMENT METHOD. Bootstrapped from the original Pascal-6000 compiler, but developed in a 6-phase stepwise-refinement method. Approximately 1.5 person-years. Runtime system rewritten for Release 3.

11. LIBRARY SUPPORT. Allows calls to external Pascal and assembler subprograms and Fortran (FTN) subroutines. The user library supplied with the system contains many intrinsic procedures and functions in addition to the Standard Pascal ones.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER.
Distributors:
-(Europe, Asia, or Africa)
See Ric Collins
Univ. of Manchester (CDC 7600) *
-(North or South America)

Implementor:
Urs Ammann
Institut fur Informatik
E.T.H. -Zentrum
CH-8092 Zurich

CDC 7600, Cyber 76 (Manchester)

0. DATE/VERSION. Release 3 of the CDC 6000 Zurich compiler (from the Minnesota maintainer) is a common release for the CDC 6000, 7600, Cyber 70, 170 series. See the letter under CDC 6000 Zurich.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. This compiler is essentially the Pascal 6000 compiler modified to fit the 7600 and Cyber 76 machines. A new run-time system is being developed using conditional assemblu in the new Release 3 run-time system by A.P. Hayes; UMRCC; Oxford Road; Manchester M13 9PL; England, U.K.; (061-273 8252).

2. MACHINE. Control Data 7600 & Cyber 76.

3. SYSTEM CONFIGURATION. SCOPE 2.1.3 or 2.14, 32K SCM.

4. DISTRIBUTION. Contact R. J. Collins at address above. A distribution agreement must be signed and the cost is 30 pounds sterling.

5. DOCUMENTATION. Same as Pascal-6000.

6. MAINTENANCE. UMRCC will assist with bugs -- in the 7600 dependant code (runtime system) only. Minnesota will accept bug reports on the compiler itself.

7. STANDARD. Same as Pascal 6000.

8. MEASUREMENTS. None yet for Release 3; [Release 2 was: Compilation speed is about 57,000 characters/sec. Compiler compiles itself in less than 10 sec. Pascal execution speed has been measured by using the obvious encoding in Pascal of Wichmann's Synthetic Benchmark (see Computer Journal Vol. 19, #1). The Units are in kilo Whetstones.

compiler and optimisation level	no runtime checking	array bound checking
ALGOL 4 (OPT=5)	1996	1230
Pascal	6850	6240*
FTN (OPT=2)	945	3174**

* Using T+ option--all run time checks included.
** Forces OPT=0.

Compiler will recompile itself on a 'half-size'(32K SCM) machine. Execution space-- Core requirements (octal): 42,402 SCM, or 36,045 if segment loaded (using a simple segment structure). Self compiles in less than 60,000. (* No information provided on size of compiler or object code produced. *)]

9. RELIABILITY. Same situation as Pascal 6000 (Zurich).

10. DEVELOPMENT METHOD. Cross compiled from Cyber 72 compiler. Based on Zurich 6000 compiler with necessary additions for this machine. (* Person-hours to develop system not reported. *)

11. LIBRARY SUPPORT. Same as Pascal 6000.

CDC Cyber 18 La Jolla

We have received no new information on this implementation since that which we published last year in Pascal News issues: #9-10: 75. #11: 81.

CDC Cyber 18 Berlin

We have received no new information on this implementation since that which we published last year in Pascal News issue: #11: 81-82.

CDC Omega 480-I, 480-II

See IBM 360/370.

CDC STAR-100 (Cyber 203) Virginia

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 77.

CII 10070 France

We have received no new information on this implementation since that which we published last year in Pascal News issues: #9-10: 77-78. #12: 59-60. (see also Xerox Sigma 7 Tokyo.)

CII IRIS 50 Nice

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 77.

CII IRIS 80 Paris, France

We have received no new information on this implementation since that which we published last year in Pascal News issues: #9-10: 77-78. #12: 59-60. (see also Xerox Sigma 7 Tokyo.)

Commodore Pet 2001

See MOS Technology 6502.

Computer Automation LSI-2 and LSI-4 Irvine

We have received no new information on this implementation since that which we published last year in Pascal News issues: #9-10: 78. #12: 60.

CRAY-1 Los Alamos

We have received no new information on this implementation since that which we published last year in Pascal News issues: #9-10: 78-79.

Cromemco Z-2

See Zilog Z-80.

Data General -- Introduction

Data General
Route 9, Westboro, Massachusetts 01581
Telephone: 617-366-8911

27 April 1978

Dear Mr. Mickel:

I am writing to you because of the article that appeared in Computerworld, April 24, on the growth of Pascal and Pascal User's Groups.

The User's Group at Data General will soon have a Pascal Special Interest Group. It is being organized by a member of PUG, Rodney Thayer. He has agreed to serve as an interim co-chairman until the group can elect officers. There will be a Pascal session at the 1978 Annual User's Group Meeting.

The version of Pascal that we are using is one that has been supplied by R.E. Berry at the University of Lancaster. If you have any question about the Pascal User's Group at Data General please feel free to contact either Rodney Thayer or myself.

Sincerely,

Kenneth A. Roy

Kenneth A. Roy
D.G. User's Group

Richard E. Adams; 967 Atlantic Ave.; Apt. 634; Columbus OH 43229; 614/436-3206 asked (* 78/7/31 *): "I have not seen any references to a Pascal compiler running under Data General's Advanced Operating System (AOS). Is anyone out there working on it?"

Data General Eclipse/Nova Columbia

0. DATE/VERSION. 78/3/8.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Rhintek, Inc.; Box 220; Columbia, MD 21045 (301).

2. MACHINE. Data General Nova or Eclipse minicomputers or equivalents. We are using the compiler on a Nova 3/D running Rev. 6.10 mapped RDOS. However, we are cleaning up the code and expect the compiler to be able to run under unmapped RDOS on a 32k Nova within a few weeks.

3. SYSTEM CONFIGURATION. Mapped RDOS system or 32k unmapped RDOS with minimum operating system. The current revision of Data General RDOS will be supported but the compiler should work with older versions.

4. DISTRIBUTION. 9 track magnetic tape, 800 bpi, 7.5 inch tape in the RDOS dump format. Price for a single user license is $975. Multi-use, OEM's, and educational use licenses will be handled on a separate basis.

5. DOCUMENTATION. The package includes source code, binary code, and ready-to-run demo programs. Instructions for executing the compiler are included; the operational information can be obtained from books by Per Brinch Hansen or Al Hartman.

6. MAINTENANCE. Updates for 1 year and notification of substantial enhancements as long as interest is shown. We will maintain a users group and encourage bug reports and suggestions. This compiler is used by Rhintek as an application and system programing language and will continue to receive support and enhancements by us.

7. STANDARD. Based on Sequential Pascal which varies from Standard Pascal. The current version lacks: file, goto, label, and packed reserved words and sqr, sin, cos, arctan, ln, exp, sqrt, eof, eoln, odd, and round built in functions.

8. MEASUREMENTS. The compiler compiles source code at the rate of 200 lines/min. This is about 1/2 the rate of the PDP-11/45 but about 5 to 10 times the speed of other compilers on the Nova. The compiler will compile itself in about 30 minutes total. (* Compilation and execution space requirements not reported. *)

9. RELIABILITY. Good. (* date first released, number of sites using system not reported. *)

10. DEVELOPMENT METHOD. The virtal machine was coded in Nova assembler language and then the compiler was modified along with the interpreter into its present form. (* Person-months to develop system not reported. *)

11. LIBRARY SUPPORT. There is no library support as yet. The operating programs support program swapping or chaining with only minimal effort as this is used with the compiler.

Data General Eclipse San Bernadino

MD
MEDICAL DATA CONSULTANTS
(714) 825-2683

MDC ECLIPSE RDOS PASCAL
Version 3

0. PRODUCT DESCRIPTION. MDC PASCAL Version 3 is an efficient PASCAL compiler and run-time support system designed for the execution of small PASCAL programs in a mini-computer environment. The development criteria are as follows:
   A. To support interactive I/O in a reasonable way.
   B. To be compatible with, as far as possible, the existing MDC ECLIPSE RDOS PASCAL Version 2.
   C. Close agreement with the P4 'standard'.
   D. A reasonable integration into RDOS. (We support background/foreground, subdirectories, and a simple command-line form of activation).
   E. Speed of execution is a primary concern in Version 3. The size of the object program is secondary to this speed criterion.
   F. Although written in assembly language, much effort has been made to preserve the modularity and intelligibility of the code.

The magnetic tape we distribute contains executable object code, source code, and machine readable documentation. It is assumed that the user has an existing MDC ECLIPSE RDOS PASCAL Version 2 operating at his site.

1. DISTRIBUTOR/IMPLEMENTOR/MAINTAINER. Ted C. Park; Director, Systems Development; Medical Data Consultants; 1894 Commercenter West, Suite 302; San Bernardino, CA 92408.

2. MACHINE. Data General - any ECLIPSE-line computer.

3. SYSTEM CONFIGURATION. ECLIPSE must have FPU or EAU, minimum of 16K words user memory, RDOS REV 6.1 or greater, FORTRAN 5 (any recent revision).

4. DISTRIBUTION. System supplied on 9-track 800 BPI tape in RDOS 'dump' format. The cost is $100.00 to cover our mailing and duplicating costs.

5. DOCUMENTATION. User must obtain his own copy of the Pascal Users Manual and Report. It is recommended that the user obtain an implementation kit from the University of Colorado. Documentation and operating procedures are supplied on the tape.

6. MAINTENANCE POLICY. Bug reports are welcome but no formal commitment for support can be made at this time. Extensive testing of the product has been done and all known bugs have been eliminated.

7. STANDARD. PASCAL P4 subset.

8. MEASUREMENTS.

Compilation Speed:	40 chars/sec (including blanks and comments)
Word Size:	16 bits
Real Arithmetic:	Uses 32 bits
Integer Arithmetic:	Uses 16 bits
Set Size:	64 bits
Execution Speed:	Approximately the same as the code produced by Data General FORTRAN V compiler
Minimum Memory Needed:	16K words

9. RELIABILITY. Version 1 exists in at least 10 sites, we believe no bugs exits. Version 2 is primarily the same as Version 1 except with improved operating procedures, faster compiles and executions, and increased capability; it also exists in at least 10 sites, we believe no bugs exist here either. Version 3 is a new product and has had thorough in-house testing. From our past experience, we have every reason to expect good performance in the field.

10. DEVELOPMENT METHOD. Developed from PASCAL-P4. Version 3 consists of a small program which rearranges the PCODE output by the compiler into a form syntactically acceptable to the Data General macro-assembler. A macro-library is supplied which will convert each PCODE instruction into one or more ECLIPSE instructions. The output from the assembler may then be submitted to the normal Data General relocating load procedure to produce an executable core image file. A runtime support library which includes some initialization routines, an error routine, I/O routines, and transcendental function routines is also included. All programs are written in assembly language and are extremely modular and well documented so that any changes wished by the user should be easy to incorporate.

11. LIBRARY SUPPORT. No Data General libraries are needed to run the system nor is it possible to use any if desired.

Data General Nova Austin, TX

THE UNIVERSITY OF TEXAS AT AUSTIN
COLLEGE OF NATURAL SCIENCES
AUSTIN, TEXAS 78712

Department of Computer Sciences
Painter Hall 3.28

14 May 1978

Dear Andy,

I am enclosing three reports on work which I have been doing (did) on implementing Pascal (or a Pascal-like language at least) on a Nova 3/D. This work differs from the University of Lancaster Version by directly compiling assembly code, not hypothetical stack code which must then be interpreted.

Sincerely,

James Peterson
Assistant Professor

(* See Abstracts, above right. *)

"Using Pascal on the Novas"

Abstract: This note describes the procedure for using the Pascal compiler on the Nova computer system at the Department of Computer Science at the University of Texas at Austin. It also indicates the limitations of the system and how they can be overcome.

"A Compiler for a Pascal-like Language"

Abstract: The development of major software systems for the Nova computer system can benefit greatly from the existance of a systems programming language. The development of such a language, and its supporting compiler is currently underway. This note reports on the language definition ant the mechanics of the compiler.

"Code Generation for a Pascal Compiler for a Nova Computer"

Abstract: A compiler is being written to translate a Pascal-like language into assembly code for the Data General Nova 3/D computer. A previous note has described the language and the basic structure of the compiler. In this note, we describe the code-generation problems encountered and their solution.

Data General Nova 840 Barcelona, Spain

We have received no new information on this implepentation since that which we published last year in Pascal News issue: #9-10: 81-82.

Data General Nova (Lancaster)

July 27, 1978

Dear Andy,

Enclosed is my renewal and here is some up-to-date information on our PASCAL distribution effort:

We are currently distributing Revision 2.01 of the Lancaster compiler for the NOVA. This revision has eliminated some of the minor problems found in the first release and has added some enhancements, such as separately compiled procedures and support for random I/O. The source code and binaries are available on magnetic tape for $140. The binaries only are $70.

We have had a tremendous response to our press releases about the compiler, and have shipped some 59 copies so far, including copies sent to 7 foreign countries, even though we are only soliciting U.S. business. The number of reader response bingo-card inquiries is approaching 1000, indicating a high degree of interest in the language, particularly from the commercial and industrial community. In fact, many of the inquiries have come from England, where this version was originally developed. Our customers have had very few problems with the Lancaster software, and we now have several applications programs running in PASCAL on the NOVA.

Sincerely,

Hank

H. S. Magnuski
Gamma Technology, Inc.

GAMMA TECHNOLOGY

800 Welch Road ▪ Palo Alto ▪ California 94304 ▪ 415-326-1661 ▪ TWX: 910-373-1296

0. DATE/VERSION. Checklist last updated 77/10/27.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER.

Distributors:

(Europe, Asia, Africa):
R. E. Berry and A. Foster;
Dept. of Computer Studies;
University of Lancaster;
Bailrigg, Lancaster LA1 4YX, U.K.;
65201 (STD 0524).

(Western America):
H. S. Magnuski
Gamma Technology
800 Welch Rd.
Palo Alto, CA 94304
415/326-1161
TWX: 910-373-1296

(Eastern America):
Jim Herbert
51 Thomas Rd.
Swampscott, MA 01907
(* No phone number
provided. *)

Implementors: R. E. Berry and A. Foster.

2. MACHINE. Data General Nova series (2/10, 820).

3. SYSTEM CONFIGURATION. RDOS 4.02/5.00 operating system; 32K core, disk backing store. No hardware multiply/divide or floating point needed. One user reports using system with RDOS without any trouble.

4. DISTRIBUTION. From Lancaster: Cassette tape or 2.5 Mbyte cartridge disk (* cost not reported *) From Palo Alto: 800 bpi 9 track tape, binary only-$70,with source-$140; From Swampscott: (* format, cost not reported *).

5. DOCUMENTATION. A 82-page user manual is provided. (* Not known if this is machine retrievable. *)

6. MAINTENANCE. No formal commitment to provide support can be given, however, bug reports are welcome. To date all known bugs have been fixed and this policy will continue as long as is practicable.

7. STANDARD. Pascal P4 subset accepted. Extensions for random I/O provided.

8. MEASUREMENTS. Typical runtimes compare favorably with those of other languages generally available on the Nova. P-code is generated, assembled and then interpreted.

	Release 1	Release 2	
Compiler NMAX (decimal)	14,055	15,505	
additional fixed table space	1,092	1,197	(in words)

The workspace remaining depends upon size of the RDOS system used. The size of program which can be compiled depends on the number of user defined symbols (dyunamic area used) and 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 some 1,100 lines of Pascal source generating 7,400 P-code instructions and we can compile this on our 32 k system. We cannot compile the compiler but would expect to do so with more than 32 k core.

Timing information for Nova Pascal Lancaster Release 2: We have not yet compiled the compiler with our system so we cannot give figures for that. Instead to provide the basis for our statement that the performance of our Pascal "compares favorably" with DG ALGOL a list of times obtained by running some well known small, and often uninteresting program are given. The timings are taken from a Nova 2/10 running under RDOS 4.02 with 32 k of core an no hardware multiply/divide or floating point. 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 source program.

Programs:

1) A program consisting simply of begin end.
2) Matrix Mutiply of two 50 x 50 integer matrices (no I/O).
3) Matrix Mutiply Of two 50 x 50 real matrices (no I/O).
4) Sort an array of 1000 integers from ascending order into descending (no I/O).
5) Ackermans function (3,6) (no I/O).
6) Write 10,001 integers onto a file.
7) Read 10,001 integers from a file.
8) Generate 5000 random integers (printing only the last).
9) Generate 5000 random integers and write to a file.

	ALGOL		Pascal	
	compile	run	compile	run
#1	:55	:06	1:31	: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	1:36	4:46	1:57	4:30

Timings such as these offer much scope for debate. It is safer to let others draw what conclusions they will from these figures (and from any other figures which may be produced). I simply wish to observe that interpretive Pascal "compares favorably" with the code produced by DG ALGOL. In the programs used above the ALGOL and the Pascal look 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 contributre to such timing comparisons I would be glad to hear about them.

9. RELIABILITY. Release 2.01 has been distributed to 50 known sites. No significant bugs have been reported from external users. First released 77/01; Latest release 78/7/27.

10. DEVELOPMENT METHOD. 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. (* Person-months to create system not reported. *)

11. LIBRARY SUPPORT. 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.

DEC -- Introduction

University of Montana
Missoula, Montana 59812

DEPARTMENT OF COMPUTER SCIENCE
Phone: (406) 243-2883

October 12, 1978

Dear Andy:

The DECUS PASCAL SIG is alive and well even though I am now in the Big Sky Country (Montana). My steering committee now resides in the four corners of the United States, but we are actively working on several PASCAL related projects. We are keeping in touch with Seved Torstendahl (Sweden) as a US focal point for his PDP-11 PASCAL Compiler. In addition, we are actively pursuing the implementation of the NBS (National Bureau of Standards) PASCAL Compiler on the following PDP-11 operating systems: UNIX, RSX-11, IAS, RSTS, and RT-11. In addition to PDP-11's a small portion of our group is working on a version of the NBS PASCAL Compiler for the VAX-11/780. We are very interested in all of the standardization efforts currently under way. I attended part of Ken Bowles' meeting at UCSD this summer and Justin Walker (NBS) is interested in implementing some of the agreed upon extensions for externally compiled modules. Please publish as much of the UCSD summer meeting report as possible in future issues of the PUG newsletter.

Dr. Roy Touzeau, also of the Computer Science Department here at the University of Montana, is also working on a DECSYSTEM-20 version of Charles Hedrick's DEC-10 (KL10) PASCAL Compiler from Rutgers University. He has modified the run-time system to remove the dynamic page management code as the DEC-20 does its own paging. He is presently changing the run-time support to use TOPS-20 system calls rather than depending on the DEC-10 compatibility code. Future plans are to produce a one-step compiler/linker for student use in introductory programming courses. Any comments or suggestions regarding this effort may be sent directly to Roy.

Sincerely yours,

John R. Barr
Assistant Professor

DEC LSI-11 UCSD

We have received copies of two papers on the UCSD Pascal system; the titles are: "A Brief Description of the UCSD Pascal Software System' (* 78/6/1 *), and "Newsletter #2--UCSD Pascal Project" (* 78/5/30 *).

Jim McCord; 330 Vereda Leyenda; Goleta, CA 93017; 805/968-6681 reports: "I am acting as the distributor for UCSD Pascal for hobby users of the LSI-11. Cost is $50, of which $35 goes to UCSD for continued work. Other $15 pays for documentaion and postage, if user sends me 4 floppies. (Else I will provide for $3 each.) This includes all source code for everything, including the interpreter. Anybody interested should get in touch with me (we already have 7 users).

Following checklist submitted by George Gonzalez, Special Interactive Computing Laboratory; 134 Space Science Center; University of Minnesota; Minneapolis, MN 55455 on 78/10/01.

0. DATE/VERSION. I.4, released about May, 1978.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. UCSD Pascal Project; Institute for Information Studies; University of California-San Diego; Mail Code C-021; La Jolla, CA 92093; 715/452-4526.

2. MACHINE. PDP-11, LSI-11 series with 16-28 kwords memory; and various 8 and 16-bit micros: Intel 8080, Zilog Z-80, etc.

3. SYSTEM CONFIGURATION. Has own operating system. Does not run under any other system (but can be brought up under CP/M). Requires 16-28 kwords (unmapped).

4. DISTRIBUTION. Source & object programs available on RX01 diskettes. Contact UCSD for more information. Cost - $50 for binaries; $200 for source, maintenance and binaries.

5. DOCUMENTATION. User Manual. Gives overview of operating system and differences with/extensions to Standard Pascal. Not machine retrievable.

6. MAINTENANCE. One-year maintenance (optional at higher cost).

7. STANDARD. Not implemented: Program header with file parameters; procedures dispose, pack, unpack; no procedures or functions as parameters; no boolean conversion in write procedure;

Differences: input^ is initially undefined; read(input,ch) is defined as begin get(input); ch := input^ end, instead of the Standard Pascal definition; rewrite requires a second parameter which specifies the system file name; files are not automatically closed at block exit; gotos cannot cross block boundries.

Extensions: Numerous (but ill-defined) extensions: character strings as an intrinsic type; string-manipulation facilities; random access to files; dynamic file opening/closing; shared variables for system communication; I/O error detection capability; segmentation (overlay) scheme.

8. MEASUREMENTS. Compiles a 3400-line program in 28k words, at 400-600 lines /minute. (* How this compares with FORTRAN, other languages not reported. *) (* Execution speed, space not reported. *)

9. RELIABILITY.

The reliability of the Standard Pascal constructs is good.

Large (3000 line) programs, plus several 'portable' Pascal programs (XREF, COMPARE, PRETTYPRINT) have been run with no problems attributable to the Standard Pascal constructs.

The reliability of the UCSD "extensions" is generally poor.

The string-manipulation intrinsics (COPY, POS, CONCAT) do insufficient error checking. The graphics intrinsics do not check for out-of-range arguments (which usually crash the program). Writing on a reset'ed file can destroy other files. The compiler allows literal strings to be passed as var parameters to string intrinsics. This can change the value of the literal. Writing a file which overflows available space does not cause an error.

10. DEVELOPMENT METHOD. P-code compiler/interpreter system. Based on P2. First released 77/8/1. About 300 sites using system.

11. LIBRARY SUPPORT. Compiler can read external source files. Predefined procedures are provided for text-string manipulation, memory-mapped graphics, and system level input/output. These intrinsics are generally ill-defined and unreliable. No symbolic dump is available. The object-code level debugger supplied requires extensive knowledge of the object code layout.

DEC PDP-8 (Minnesota)

0. DATE/VERSION. Checklist updated 78/10/5.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. John T. Easton, 612/373-7525; James F. Miner, 612/373-9916; Address correspondence to: Pascal Group; SSRFC; 25 Blegen Hall; University of Minnesota; 269 19th Ave. South; Minneapolis, MN 55455; 612/373-5599.

2. MACHINE. Digital Equipment Corp. PDP-8/e.

3. SYSTEM CONFIGURATION.
OS/8 version 3. Hardware required:
-RK8-E disk, or other direct access mass storage.
-12 K minimum of core/RAM. 32 K is required for compilation. Can use up to 128K.

4. DISTRIBUTION. Release scheduled for second quarter, 1979.

5. DOCUMENTATION. Machine-retrievable supplement to Pascal User Manual and Report (about 25 pages), in preparation.

6. MAINTENANCE. A policy has not yet been determined.

7. STANDARD. Emphasis has been on close adherance to the Pascal User Manual and Report. There are two major restrictions: a) Procedures and functions may not be passed as parameters. This restriction will not be lifted without full type checking (which requires a change in the Pascal Standard). b) Files may be declared only in the main program, and files may not be components of arrays, records, or files; nor may files be allocated with the procedure NEW. Minor restrictions: set size=96 elements; maxint=8,388,607 (2**23-1). Full-ASCII character set is supported. Major extensions supported: a) direct-access files; b) default case; c) run-time file binding; d) overlays.

8. MEASUREMENTS.
Execution speed--roughly comparable to FORTRAN IV (F4). I/O tends to be faster than FORTRAN, while computation tends to be slower.
Execution space--Interpreter takes 8K, space needed for P-code and run-time storage depends on program.

9. RELIABILITY. Fair to good and improving. An earlier implementation has been in use at 1 site since 76/11.

10. DEVELOPMENT METHOD. As with most languages on the PDP-8, Pascal makes use of an interpreter (a modification of P-code) written in MACREL. The compiler (about 5000 lines, based on Pascal-P4) is written in Pascal. All standard procedures are written in MACREL. The implementation is not suitable for real-time applications.

11. LIBRARY SUPPORT. Currently (78/11/15), none planned for the first release.

DEC PDP-11 (Amsterdam)

0. DATE/VERSION. Checklist not updated since 78/02.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Sources, binaries, and documentation are part of the third UNIX software distribution. Implementor: Johan Stevenson, Vrije Universiteit. Maintainer: Andrew S. Tanenbaum; Vakgroep Informatica; Wiskundig Seminarium, Vrije Universiteit; De Boelelaan 1081; Amsterdam, The Netherlands; 020/ 548-2410.

2. MACHINE. Any PDP-11 on which UNIX version 6 will run.

3. SYSTEM CONFIGURATION. See 2.

4. DISTRIBUTION. Through the UNIX software distribution center. (* No information on cost reported. *)

5. DOCUMENTATION. Short manuals for the compiler and interpreter in UNIX MAN format and a 12 page description giving details about the implementation.

6. MAINTENANCE. Bug reports are welcome. There will be an improved release of the current system. However, we are working on a totally new one. Main differences from the old one are:

-a new hypothetical stack computer named EM1 (see Tanenbaum, A. S., "Implications of structured programming for machine architecture", CACM, Dec. 1977). This intermediate machine allows very compact code (only 15,000 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.
-an new interpreter with all kinds of run-time checks and debugging aids.
-expansion on EM1 codes into PDP-11 instructions.
-less restrictions on the language Pascal.

7. STANDARD. Main differances with Standard Pascal are:
-no gotos 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.
-at most 8 files(all text), including input and output.
-An explicit get or readln is needed to initialize the file window
-empty fields and fieldlists are not allowed in record declarations.
-procedure unpack not available, packed ignored; all records are automatically packed.
Maximum length of string constant is 80 characters. Ordinal value of a set element must be between 0 and 63 inclusive. maxint = 32,767 (2**15 -1). Setsize = 0..63. Full ASCII accepted (parity ignored). Keywords and standard names are recognized in lower case.

8. MEASUREMENTS.
compilation speed--40,000 char/min on a 11/45 with cache.
compilation space--48k bytes to compile the compiler. Very big programs can be compiled.
execution speed--you lose a factor of 8 by interpretation. However, I/O is relatively fast. Compared to interpreted Pascal on a big machine (CDC Cyber 73) it is 10 times slower.
execution space--the size of the complete interpreter is 5300 bytes. The binary code of the compiler is 23,000 bytes.

9. RELIABILITY. The compiler and interpreter are good. However, the run-time checking of the interpreter is poor. Preliminary version first ran in 1977. (* Date system first released to users, number of sites using system not reported. *)

10. DEVELOPMENT METHOD. The compiler is based on the Pascal-P2 compiler. A Cyber 73 was used for bootstrapping. The time needed by one inexperienced implementor was about 6 months.

11. LIBRARY SUPPORT. No library support at all. There are some hidden library routines used by the system.

DEC PDP-11 Berkeley

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

29 April 78

Dear Andy,

I was suprised that there wasn't anything in the PN last time about the Berkeley UNIX (PDP-11) Pascal. I thought I'd let you know it exists, since the implementors apparently haven't told you anything.

It is an interpretive system written for support of computer science instruction, so it is very fast at generating (intermediate) code, but slow at execution. The syntax scan is the best I've seen (of any compiler for any language); it is very informative for unexperienced users, comments on suspicious (but syntactly correct) code, and corrects some trivial syntax errors such as semicolon before ELSE. Such corrections show on the listing but the correct intermediate code is generated -- the note will continue to appear on subsequent listings until the source file is changed by the user, of course. Definately accepts Standard Pascal: I swap very large programs back and forth between the PDP-11 and the CDC 6400 with only changes required in first and last character constants (MINCHAR and MAXCHAR).

The development was supported at least in part by US ERDA, and the authors seem willing to distribute it for instructional use. A fifty-one page user's manual, titled "UNIX Pascal User's Manual, Version 1.0 -- September 1977" is available from the Computer Science Library for a couple of bucks. The authors of the manual are William N. Joy, Susan L. Graham and Charles B. Haley. Joy and Graham can be reached at the UCB Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA 94720. Graham's office phone # is 415-642-2059. I think Haley has left, I have a vague recollection that he is at Bell Labs now.

This is an excellent Pascal system, which I would recommend highly to anyone running under UNIX. Of course, since it is an interpretive system there would be execution time problems for some production applications.

Sincerely,
Willett Kempton
Willett Kempton

p.s. Runs on 11/45 and 11/70. Doesn't accept procedure and function names as parameters. I'll send you some documentation if I get time.

DEC PDP-11 Los Altos

We have received no new information on this UNIX, RT-11, DOS, and RSX-11 implementation since that which we published last year in Pascal News issue: #9-10: 83.

DEC PDP-11 Missoula, MT

We have received no new information on this RSX-11 implementation since that which we published last year in Pascal News issue: #11: 91.

DEC PDP-11 (OMSI) (formerly ESI)

Maurice R. Munsie; Network Computer Services p/l; 69 Clarence St.; Sydney 2000 Australia reports: "We are distributing in Australia OMSI Pascal-1. A number of sales have already been made and plans are being made for the OMSI implementors to hold workshops in Australia later this year." (* 78/8/28 *)

0. DATE/VERSION. 77/12/76.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Oregon Minicomputer Software, Inc. (OMSI); 4015 SW Canyon Road; Portland, OR 97221; 503/226-7760. Implementors: John Ankcorn, Don Baccus, and Dave Rowlar .

2. MACHINE. Any model Digital Equipment Corp. PDP-11 or LSI-11.

3. SYSTEM CONFIGURATION. Minimum of 16K words. Operates under RT-11, RSTS/E, or RSX.

4. DISTRIBUTION. Compiler, support module, cross referencer, text editor and instruction manual available for $1500 ($995 for educational use). Available on 9 track 800 bpi magnetic tape, or DEC cartridge disk.

5. DOCUMENTATION. Over 70-page machine-retrievable instruction manual. Currently (76/11/02) working on more.

6. MAINTENANCE. One year of unlimited fixes and updates, followed by annual subscription service. (* Reported by users that "vendor seems to be responsive in terms of support". *)

7. STANDARD. Full standard plus extensions: additional features for real-time hardware control; separate compilation of procedures; Macro (assembler) code in-line insertion; actual core addresses of variables can be fixed (giving access to external page I/O addresses at the Pascal level.

8. MEASUREMENTS.
compilation speed--About 3500 characters /second, on the PDP-11 model 05.
compilation space--very economical-it can compile 3000 line programs in 28K on PDP-11/40. No overlays are used in the system.
execution speed--about twice as fast as the DEC FORTRAN IV and many times faster than DEC BASIC. A worst-case 'number-cruncher' example ran at 40% faster than the DEC original FORTRAN.
execution space--very economical-much of the space improvement over DEC FORTRAN is due to the smaller support module for Pascal.

9. RELIABILITY. Excellent--far better than DEC FORTRAN. In use since 75/11. Over 100 installations, and growing steadily.

10. DEVELOPMENT METHOD. Single-pass recursive-descent compiler written in Macro-11. Hand-coded based on University of Illinois bootstrap (with extensive changes) in about two person-years of effort. First compiler written by both implementors. Compiler translates source into Macro-11 which is then assembled and linked to the support module for execution.

11. LIBRARY SUPPORT. Separate compilation of procedures with load-time insertion and linkage is implemented.

DEC PDP-11 Redondo Beach

We have received no new information on this Concurrent Pascal (SOLO) implementation since that which we published last year in Pascal News issues: #11: 89-90.

DEC PDP-11 (Stockholm)

0. DATE/VERSION. 77/12/22.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Seved Torstendahl; Tn/X/Tdg.; Telefon AB LM Ericsson; AL/Ufe; S-125 26 Stockholm, Sweden; 08/719-4909.

2. MACHINE. Digital Equipment Corp.:
DEC-10 (cross-compiler that generates code for all PDP-11's);
PDP-11 model 35 and up (self compiles
and generates code for all PDP-11's);
The compilers generate code for floating point hardware and extended arithemtic instruction sets if option switches are set.

3. SYSTEM CONFIGURATION. DEC-10 cross-compiler: TOPS-10. PDP-11: RSX-11M (Probably it is an easy task to replace the RSX interfacing routines with new ones interfacing to DOS or RT-11; but we do not plan to do that work here. Maybe routines to interface with RSX-11S will be made.) PDP-11 with memory management and a user partition of at least 28k words, preferably 32k words.

4. DISTRIBUTION. The compilers are available at $50, plus $10 if we supply the tape (600 feet). The distribution set includes source and object modules of the compilers and the runtime library, command files for compiler generation and maintenance, user manual and compiler generation instructions. The compiler will be distributed in one or more of the following formats; indicate which you want:
- three DECtapes in PDP-11 DOS format (DEC-10 and PDP-11 users)
- one 9-track magnetic tape in DEC-10 format (DEC-10 users)
- one 9-track magnetic tape in industry standard format
- one 9-track magnetic tape in DOS format (PDP-11 users).

5. DOCUMENTATION. A machine-retrievable user manual, complementing the Pascal User Manual and Report, is included on the distribution tape.

6. MAINTENANCE. No responsibility, but if errors are found reports will be distributed to known users. Error reports and improvement suggestions accepted.

7. STANDARD. With regard to the definition of Pascal in Pascal User Manual and Report, the following restrictions hold:
- packed data structures are only implemented for character arrays (always packed, two chars/word) and for Boolean arrays (packing optional, one Boolean / bit). The standard procedures pack and unpack are not implemented.
- only local jumps are allowed.
- a pair of procedures, "mark" and "release", have been added to allocate and deallocate dynamic storage.

The following extensions have been implemented:
- function results can be of a nonscalar type.
- arrays with unspecified bounds (but specified index-structure) can be used as formal parameters to procedures, allowing differently declared variables or constants to be used as actual parameters.
- a string parameter type has been introduced in which one-dimensional character arrays or substrings thereof may be passed as parameters. Such strings and their constituent characters are considered as "read-only".

- procedures may be compiled separately.
- separately compiled procedures can be accessed through a declaration with the procedure block replaced with "extern".
- most option selectors ((*$M+ *), etc.) are selectable by switches on the MCR command line (version 5, 77/12).

8. MEASUREMENTS.
compilation speed--about 300 characters/second; increases to 3000
characters/second in a 64 k words partion using
PLAS under RSX-11M.
compilation space--The compiler requires a 32k word partion (at least
26 k words for very small programs).
execution speed--(* No information provided. *)
execution space--(* No information provided. *)
(* How this compares to FORTRAN and other languages not reported. *)

9. RELIABILITY. Excellent. The compiler is now in use at over 200 sites. Only minor errors have been found since July, 1977. First version released April, 1977. Latest version: December, 1977.

10. DEVELOPMENT METHOD. The compiler is a modification of the cross compiler from Mr. Bron, et. al. of Twente University of Technology in the Netherlands. The original cross-compiler was written in Pascal and developed from Pascal-P. Two major modifications have been undertaken:

- the compiler generates standard object modules;
- the compiler gives full access to the RSX/IAS file system.

The compilers are written in Pascal, and both have the same source code except for two separately compiled routines. The cross compiler is generated when the DEC-10 Pascal compiler from Hamburg compiles the source. When it then compiles itself the PDP-11 version is created. The cross compiler for PDP-11 running on DEC-10 produced by Bron et al was used as input. This compiler was modified to generate object code linkable under RSX-11M and to give access to the file system of RSX-11M. When the cross compiler was finished it compiled itself and was thus transfered to the PDP-11. The implementation effort until now (77/02/09) has been about five person-months. To make use of floating point hardware another two person-months will be needed. A new version which performs some optimization will probably be developed later.

11. LIBRARY SUPPORT. Separate compilation allowed. Possible to use external procedures written in FORTRAN or assembler. The December 1977 version also gives: Automatic copy of text from library into source program (include); execution frequency measurements; execution trace; option selectors ((*$R- *), etc.), settable by switches in the MCR command line. Next version (Spring, 1978) will also include a symbolic post-mortem dump an an interactive source-level debugging package (mainly copied from the DEC-10 Hamburg-DECUS compiler).

DEC PDP-11 Tampere, Finland

The DEC PDP-11 Stockholm Pascal system (for RSX-11M) was modified slightly during October, 1977 to run under IAS by: Jyrki Tuomi and Matti Karinen; Tampere University of Technology; Computing Center; SF-33100 Tampere 10; Finland; (* No phone number reported *). A 60-page report on this implementation (in Finnish) is available from Tampere.

DEC PDP-11 Twente

We have received no new information on this implementation of a cross-compiler from DEC-10 to any PDP-11 on any operating system since that which we published last year in Pascal News issue: #9-10: 85.

DEC PDP-11 Vienna, Austria

We have received no new information on this RSX-11D implementation since that which we published last year in Pascal News issue: #9-10: 85-86.

DEC VAX-11/780 Seattle

We have received no new information on this implementation since that which we published last year in Pascal News issue: #12: 63.

DEC VAX-11/780 (Redondo Beach)

We have heard rumors that an implementation is underway at TRW corporation at Redondo Beach, CA.

DEC-10 (Hamburg-DECUS)

0. DATE/VERSION. Checklist not updated since 77/08.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Implementor/Maintainer: E. Kisicki; H. -H. Nagel; Universtat Hamburg; Institut fur Informatik; SchluterstraBe 66-72; D-2000 Hamburg 13, Germany; 040-4123-4151; TELEX: 214 732 uni hh d.

Distributor(Western Hemisphere):
DECUS;

Maynard, MA 01754;
USA;
617/ 897-5111;
TELEX: 94 8457;
TWX: 710 347 0212.

(Eastern Hemisphere):
DECUS-Europe;
P. O. Box 340;
CH-1211;
Geneva 26, Switzerland;
022/ 42 79 50;
TELEX 22593.

2. MACHINE. Digital Equipment Corp. DEC-10. (Adapted to the DEC-20 by DEC).

3. SYSTEM CONFIGURATION. DEC TOPS-10 moniter using Concise Command Language (CCL). Uses KA-10 instruction set. Modifications to use KI-10 improved instruction set have been made by Charles Hedrick.

4. DISTRIBUTION. From DECUS (Digital Equipment Corp. User's Society).

5. DOCUMENTATION. Machine-retrievable manual included on distribution tape.

6. MAINTENANCE. No regular maintainance can be given.

7. STANDARD. Extensions: Functions FIRST and LAST for scalars; UPPERBOUND and LOWERBOUND for arrays; MIN and MAX available as standard functions; procedures to determine the value of CCL options available; otherwise in case statement; LOOP...EXIT IF...END statement; Initialization procedure.

8. MEASUREMENTS. (* No information provided. *)

9. RELIABILITY. Very good. First version released in 75/7. Distributed to at least 60 sites. Later version operational in 76/9. Latest version released to DECUS in 77/2.

10. DEVELOPMENT METHOD. Pascal-P2 and subsequent self bootstraps. Latest version dated 76/12/30.

11. LIBRARY SUPPORT. Symbolic post-mortem dump available. Interactive run-time source-level debugging package available. Separate compilation and inclusion in relocatable object code library of Pascal, FORTRAN, COBOL, ALGOL, and MACRO-10 assembler routines.

DEC-10 Systems-Pascal

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 90-91.

DEC-20

See DEC-10 Hamburg-DECUS.

Dietz Mincal 621 Hamburg

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 91-92.

FOXBORO Fox-1

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 92.

FUJITSU Facom 230-30 Tokyo

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 92.

FUJITSU Facom 230-55

See FUJITSU Facom 230-30 Tokyo.

Harris/4 Delft

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 92.

Heathkit H-11

(* This machine is based on the LSI-11 microprocessor from DEC and it is believed that the DEC LSI-11 (UCSD) implementation will run on this machine; though nothing definite has been reported. *)

According to Bill Schiffbauer; Sales Coordinator, Computer Products; Heath Company; Benton Harbor, MI 49022; 616/982-3285; TELEX 72-9421: "At this time (* 77/11/15 *), Heath has no plans to offer a Pascal compiler or interpreter...Since the H-11 uses the LSI-11, the [UCSD Pascal] compiler should be compatible with the H-11."

According to Robert W. Furtaw; Marketing, Heath Company, Benton Harbor, MI 49022: (* 78/1/19 *) "We also have been observing the appeals for Pascal appearing in recent publications. However, we presently have no immediate plans to offer one for our system. With all the interest, I would not be suprised to see one which could easily be reassembled for our system."

Hewlett Packard HP-2100 (Trieste, Italy)

0. DATE/VERSION. 78/10/9.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Implementor: Paolo Sipala; Instituto di Electrotechnica; Universita di Trieste; Via Valerio, 10, 34127; Trieste, Italy; Tel. 040-733033. Distributor: Hewlett-Packard Software Center; Contributors Section; 11000 Wolfe Blvd.; Cupertino, CA 95014; (* No phone number reported. *)

2. MACHINE. Hewlett Packard HP-2100 or 21MX.

3. SYSTEM CONFIGURATION. Old version-DOS IIIb; New version-RTE. There are seperate versions for EAU, non-EAU, and floating point hardware. Requires an 11k main area.

4. DISTRIBUTION. (* No information reported on cost, distribution formats. *)

5. DOCUMENTATION. (* No information provided. *)

6. MAINTENANCE. (* No information provided. *)

7. STANDARD. (* No information provided. *)

8. MEASUREMENTS. Requires an 11k main core area (so it might fit in a 16k system, if the resident operating system modules are kept to a minimum, but 24k is more comfortable). It is not noticably slower than the standard compilers when compiling, and not worse than the standard interpreter(BASIC) when interpreting.

9. RELIABILITY. Has been subjected to rather limited testing (a few dozen programs from the Users Manual) and is now (* 78/3/20 *) being offered to students for their use.

10. DEVELOPMENT METHOD. A P-code interpreter written in HP-Algol.

11. LIBRARY SUPPORT. (* No information provided. *)

Hewlett Packard HP-21 MX Durban

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 93.

Hewlett Packard 3000 Santa Clara

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 94.

Hewlett Packard 3000 Sunnyvale

We have received no new information on this implementation since that which we published last year in Pascal News issue: #12: 63-64.

HITACHI Hitac 8800/8700 Tokyo

See also IBM 360/370. We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 94.

Honeywell 6000, level 66 (Waterloo)

0. DATE/VERSION. Checklist not updated since 77/08.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Implementor: W. Morven Gentleman; Mathematics Faculty Computing Facilty; University of Waterloo; Waterloo, ONT. N2L 3G1; CANADA; 519/ 885-1211. Distributor: Honeywell Information Systems; Waltham, MA (* See local HIS sales office first. *)

2. MACHINE. Honeywell 6000 series; level 60/66. Operates under GCOS (TSS). Currently (* 76/03/08 *) a DRL TASK version is under consideration.

3. SYSTEM CONFIGURATION. Honeywell level 66 or 6000 series with EIS. Minimum of 26k words.

4. DISTRIBUTION. (* No information provided. *) (* Rumor has it that distributor charges extra for maintenance. *)

5. DOCUMENTATION. From Honeywell Information Systems; Publication Dept.; MS-339; 40 Guest St.; Brighton, MA 02135: "A Pascal Product Brief", (#AW66, free), 2 pg. (marketing oriented) and "Pascal User's Guide", (#AW65, $1.30), 30 pg. (reference manual). Machine retrievable supplement to Pascal User Manual and Report; also includes extensions, restrictions, known bugs, etc.--about 45 pages total.

6. MAINTENANCE. Supported by University of Waterloo through agreement with HIS; some users have reported problems in getting Honeywell to pass bug reports on to Waterloo. Extensions planned to allow extern to be GMAP, COBOL, ALGOL, PL/I, B, C, etc.

7. STANDARD. Restrictions:
   - Program statement not accepted, replaced by required procedure 'main'.
   - No files with components of type file.
   - Only files of type char may be read or written (with the standard read, write, get, put).

Extensions:
   - Files may be opened dynamically.
   - Extended file handling is available.
   - External separately compiled Pascal and FORTRAN procedures may be used.
   - Various procedures and functions to provide access to operating system.
   - Optional left-to-right evaluation for Boolean expressions and if statements.
   - 'else' clause in case statement.
   - Alternate Interactive I/O package available.
   - Full upper/lower case capability.

8. MEASUREMENTS.
compilation space--minimum of 26k words. Typical programs require less than 30k words.
compilation speed--(* No information provided. *)
execution space--can be as small as 4-5k words depending on the program and the Pascal support routines required.
execution speed--(* No information provided. *)
(* How this compares to FORTRAN and other languages not reported. *)

9. RELIABILITY. (* No information provided on number of sites using system. *) Some users have reported problems with compiler reliability and responsiveness of distributor. See Pascal News #11: 34-36, 92-93. Distributed since 76/05. Version 6 expected in 77/12.

10. DEVELOPMENT METHOD. Independant implementation (unrelated to Pascal-P or CDC 6000 Zurich compilers); written in "B", an implementation language and successor of BCPL. (* Person-months to create system not reported. *)

11. LIBRARY SUPPORT. Separately compiled Pascal and FORTRAN routines may be saved and called from user specified libraries at run time. A post-mortem debugger is planned, but presently (* 76/10/25 *) far from being implemented.

Honeywell H316 Minnesota

0. DATE/VERSION. 78/7/4.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Robert A. Stryk; Honeywell Corp. Computer Science Center; 10701 Lyndale Ave. S.; Bloomington, MN 55424; 612/ 887-4356.

2. MACHINE. Honeywell H-316.

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. (* cost not reported. *)

5. DOCUMENTATION. Informal comments on 316 kernal implem entation.

6. MAINTENANCE. No known errors, no work planned. Bob reported on 78/7/4: "changing jobs--Distribution of H316 Concurrent Pascal very cloudy".

7. STANDARD. A modified implementation of Concurrent Pascal, which varies from Standard Pascal.

8. MEASUREMENTS. SOLO system needs minimum of 40 k to execute compilers.

9. RELIABILITY. No known errors. (* Date first released, number of sites using system not reported. *)

10. DEVELOPMENT METHOD. The H316 kernal imitates the PDP-11 reversed byte addressing which makes it compatible with the distribution tape but a bit slow in execution. The development was done under BOS 210. The kernal is written in DAP700.

11. LIBRARY SUPPORT. That provided by the SOLO system.

IBM 1130

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 101.

IBM 360/370 AAEC

We have received a copy of a report titled "Implementation of Pascal 8000 on IBM 360 and 370 Computers" (* 78/8/4 *) which is available from the distributor.

0. Date 78/09/12.

1. Implementors:

T. Hikita and K. Ishihata,
Dept. of Information Science,
University of Tokyo,
2-11-16 Yayoi
Bunkyo-ku TOKYO, (HITAC - 8000 Version)
113 JAPAN.

G.W. Cox and J.M. Tobias,
Systems Design Section,
AAEC Research Establishment,
Private Mail Bag,
SUTHERLAND, 2232, (IBM 360/370 Version)
N.S.W. AUSTRALIA

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 CMS interface is currently being developed, soon to be available. 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 very 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 which can 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.
- 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, at times better than FORTRAN H.
- 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 early in 1978. It is currently used at about 90 sites. Reliability reports have been generally good to excellent.

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 overlayed.

A symbolic dump of local variables and traceback of procedures called is provided on detection of execution errors.

12. Future Developments.

Version 2.0 is currently under development.

IBM 360/370 Berlin

We have received no new information on this VM370 (CP + CMS) and OS implementation since that which we published last year in Pascal News issue: #11: 99-100.

IBM 360/370 Grenoble

We have received no new information on this OS/MVT and VS/MFT implementation since that which we published last year in Pascal News issue: #9-10: 100.

IBM 370 London

We have received no new information on this CMS implementation since that which we published last year in Pascal News issue: #11: 96-98.

IBM 360/370 Manitoba

We have received no new information on this MFT, MVT, VS1, VS2, MVS, and CMS implementation since that which we published last year in Pascal News issue: #9-10: 97-98.

IBM 360/370 Stanford

STANFORD UNIVERSITY

STANFORD LINEAR ACCELERATOR CENTER

Sept. 15, 1978

Mail Address
SLAC, P. O. Box 4349
Stanford, California 94305

Dear Andy:

This is to announce the release of a new version of the Stanford PASCAL Compiler. This version provides comprehensive runtime checking as well as provisions for user-requested or post-mortem (symbolic) dump, separate compilation and generation of program profile (i.e. frequency of execution of source program statements). The compiler is now about 5ØØØ lines long and, except for a few restrictions, implements the language described in Jensen & Wirth's "User Manual and Report". There are also some minor extensions to allow timing and clean termination of programs without GOTOs across procedure boundaries.

The postprocessor, which translates the output of the compiler into IBM/370 assembly or object code, has also grown to 3500 source lines but the compilation/postprocessing time for the compiler has remained almost unchanged (i.e. about 10 seconds of compilation followed by 5 seconds of postprocessing on the 370-168, or a compilation rate of ~ 500 lines per second). The combined system is still capable of self compiling in a 128K region, but a larger area improves the I/O efficiency by allocating larger buffers.

Our earlier decision in leaving the compiler as machine independent as possible and writing a separate program to translate our modified P-code into target machine code (as explained in the Pascal Newsletter #8) proved to be very helpful in simplifying the task of bootstraping the compiler on a set of drasticaly different target machines. For example, after analyzing the static and dynamic properties of programs expressed in the intermediate form, we concluded that this form was quite suitable for a very compact encoding.

A postprocessor, intended primarily for microprocessor environments, translates the full compiler into a mere 20K bytes which could be run interpretively, or implemented by a micro-coded emulator on any of the existing bit-slice processors. Another interesting outcome of this implementation was that a very small (3K bytes) 8080/Z80 based interpreter, in conjunctin with the obove postprocessor, resulted in a microprocessor-resident compiler with a compilation speed of about 100 times slower than the 370-168 in terms of the CPU time, but quite comparable in "turn around" or terminal time.

Independent from these justifications, there are also some other projects involved in writing machine independent P-code optimizers which would potentially benefit all the programs which are translated into the common intermediate form before being tied to the final target machine.

In conclusion, the PASCAL P-compiler seems to have helped spread the use of PASCAL far more than the sophisticated (and certainly more refined) 6000 Compiler from which it was derived. The Zurich group should be credited for its farsightedness in developing this compiler as a separate program as well as defining the original 'P' pseudo machine which has since established the common grounds for the portability of PASCAL systems.

Sincerely

S. Hazeghi

Sassan Hazeghi
Computation Research Group

P.S. The new version of the 370 Compiler is available through SHARE Program Library as well as Argonne Code Center, the microprocessor implementation is available only from the Argonne Code Center.

IBM 360/370 Stony Brook

We have received no new information on this OS implementation since that which we published last year in Pascal News issue: #9-10: 98-99.

IBM 360, 370 (Vancouver)

0. DATE/VERSION. Barry Pollack reported (* 78/8/7 *): "Pascal/UBC is almost ready for its next round of distributions--it is an upward compatible superset of the old Pascal/UBC system, which is upwards compatible with Standard Pascal. The system runs on IBM 360/370 and Amdahl 470 machines. We plan to begin this round of distribution in Sept. or Oct.--of course, the old system is still available."

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Barry W. Pollack and Robert A. Fraley, Department of Computer Science, University of British Columbia, Vancouver, British Columbia, Canada V6T 1W5 (604/228-6794 or 604/228-3061).

2. MACHINE. IBM 370/168.

3. SYSTEM CONFIGURATION. The current version runs under the MTS (Michigan Time Sharing) operating system. The monitor may be modified with minimal effort to run under VS, OS, etc. Standard OS object modules are generated. The translator requires about 320K bytes of store. Division of the compiler into overlays for non-VM systems would be possible.

4. DISTRIBUTION. The current version is available for distribution now, via 9 track magnetic tape. Costs will be limited to postage (and tape purchase, if one is not supplied).

5. DOCUMENTATION. A User's Guide describes completely the implementation's departures from the Jensen and Wirth Pascal User Manual and Report. (* Apparently not machine retrievable. *)

6. MAINTENANCE. No policy has been decided. It is anticipated that periodic upgrades and modifications will be distributed at least once a year. Reported bugs will be corrected as quickly as possible with notification to users.

7. STANDARD. The compiler provides numerous extensions and a few restrictions. A compiler option issues error messages when non-standard features are used. A complete description is contained within the documentation provided. A summary of the differences follows.

Extensions:

Strings are padded on the right with blanks.
a case default label: "<>".
Optional ";" allowed before else.
"(...)" may be used instead of "[...]".
The character eol has been retained.
packed is ignored.
Input of character strings using read is allowed.
Support of EBCDIC characters "&", "|", and (logical not sign). (* Sorry, we use ASCII at Pascal News. *)
Use "..." for comments.
value section exists for variable initialization.
Hexadecimal integers are supported.
A return code is available in the pre-declared variable rcode.
FORTRAN subroutines may be called. [<Direct access files are supported.
Additional built-in functions include: min, max, substr (using constant length), position (direct access files), I/O interface functions and extensions to reset and rewrite, and insert for data packing.

Restrictions:

Sets are limited to 32 elements (0..31).

Program heading is not used.
Files may not be components of other structures.
Set constructors may not include <expression>..<expression>.
input^ is initially eol instead of the first character of the file. This is transparent when read is used.

Projected extensions:
McCarthy if.
or and and lower precedence than relations.
"Usual" precedence used throughout.
Sets over the range 0..255.
Better control of input and output formats.

8. MEASUREMENTS. The compiler is written in Pascal and is modeled after the CDC 6000 implementation, but it has been extensively modified and improved. The translator consists of approximately 8000 lines of Pascal code. The run-time library consists of approximately 500 lines of Pascal code. The monitor (which contains the interface to the operating system) consists of approximately 2000 lines of IBM Assembler G code. The translator speed has not been determined, but it seems faster than our Algol-W compiler. The code produced has been timed against Algol-W code and is almost uniformly 10-15% better. This is especially true of any program using a large number of procedure calls. The compiler compiles itself in less than 60 seconds of 370/168 processor time. The compiler requires 320K bytes of core.

9. RELIABILITY. To date has been excellent. A student version of the translator has been running since September, 1976, with only one detected compiler error. The main system version has been in operation since December, 1975. All problems which have been encountered to date have been corrected. (* Number of sites using system not reported. *)

10. DEVELOPMENT METHOD. The original translator was developed by Wirth and several graduate students at Stanford University as a partial re-write of the CDC 6400 version in 1972. The current translator and monitor have been extensively modified, a run-time library has been implemented, and a post-mortem symbolic dump package has been developed. The translator has been under continuous development at UBC since December, 1975, by two faculty members and one (* anonymous? *) graduate student.

11. LIBRARY SUPPORT. Fortran routines can be called. The compiler generates standard OS object modules.

IBM 360/370 Williamsburg

We have received no new information on this OS/VS implementation since that which we published last year in Pascal News issue: #11: 95-96.

IBM Series 1 (East Providence)

It has been reported that SPAN Management Systems; Westminister Industrial Park; East Providence, RI 02914; 401/438-2200 has developed a dialect of Pascal which they call TSS and which will run on the IBM Series 1 computer; but we have received no information from them on their system.

IBM Series 1 (Reston)

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 85.

ICL -- Introduction

PCHICL - the Pascal Clearing House for ICL machines - exists 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;
Hull College of Higher Education
Inglenure Avenue
Hull HU6 7LJ
England, U.K.
(0482) 42157

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.

Pascal Compilers for the ICL 1900 series (& ICL 2903/4)

D.A.Joslin,
May 22nd 1978

1. #PASQ Issue 3

This compiler is the most suitable for ICL 1900s operating under George 4, and for those with a large core store (256K, say) operating under George 3. This is the compiler described in the Implementation Checklist in "Pascal News". It incorporates a Diagnostics Package (written by D.Watts & W.Findlay of Glasgow University) and a Source Library facility. It takes 44K to compile most programs (60K to compile itself). It may be obtained by sending a mag.tape (7-track NRZI 556 bpi or 9-track PE 1600 bpi) to the implementor, viz:

Dr. J.Welsh,
Dept. of Computer Science,
Queen's University,
BELFAST, N.Ireland, BT7 1NN.

2. #PASQ Mark 2A

This compiler is suitable for all ICL 1900s (except 1901, 1901A, 1902, 1903. 1904, 1905) & 2903/4s with at least 48K of core; it is the most suitable compiler for ICL 1900s operating under George 2, and for those operating under George 3 where core is at a premium. The language processed (the language of the revised report) is identical to that processed by #PASQ Issue 3, the compiler described in the Implementation Checklist in "Pascal News", but there is no Diagnostics Package or Source Library facility. The compiler takes 36K to compile many programs, 40K to compile all but the most complex (48K to compile itself). It was implemented originally by Queen's University, Belfast, and has been enhanced to include:

Selective compilation listing and insertion of run-time checks;
Nested comments;
Improved compilation listing layout, and full text of compilation error messages;
Improved execution error handling;
More efficient mathematical standard functions;
Facility to compile 15AM programs;
Specification of object-program card & line lengths;
Correction of various errors.

It may be obtained by sending a mag.tape (7-track NRZI 556 bpi or 9-track PE 1600 bpi) to: D.A.Joslin,

(* address on previous page *)

3. #XPAC Mark 1B

This compiler is suitable for all ICL 1900s & 2903/4s with at least 32K of core. The language processed is Pascal Mark 1, ie the language of the original report. The compiler takes 24K to compile most programs (32K to compile itself). It may be obtained by sending a mag.tape to Sussex (as in para 2 above).

4. Pascal-P

A Pascal to P(4)-code translator, configured for ICL 1900s & 2903/4s, may be obtained by sending a mag.tape to Sussex (as in para 2 above). This is suitable for all ICL 1900s (except 1901, 1901A, 1902, 1903, 1904, 1905) & 2903/4s with at least 32K of core. The language processed is broadly the language of the revised report - see the Pascal-P section of "Pascal News". The translator takes 24K to compile most programs (28K to compile itself). To complete the compilation process, either a P-code interpreter (based on the model interpreter provided) or a P-code to machine-code translator must be written.

5. Future Development

A two-stage Pascal compiler, which will be suitable for all ICL 1900s (except perhaps 1901, 1901A, 1902, 1903, 1904, 1905) & 2903/4s with at least 32K of core, is to be produced by Belfast, possibly by October 1978. The language processed will be identical to that processed by #PASQ, and a Diagnostics Package and Source Library facility (George 3/4 only) may be provided.

ICL 1900 (Belfast)

0. DATE/VERSION. Checklist last updated 77/11/4.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Jim Welsh, Colum Quinn, and Kathleen McShane, Department of Computer Science, Queens University, Belfast BT7 1NN, Northern Ireland, U.K. (* No phone number provided. *) Enhancements by David Watts and Bill Findlay, Computer Science Department, University of Glasgow, Glasgow G12 8QQ, Scotland, U.K. (* No phone number provided. *)

2. MACHINE. ICL 1900 Series.

3. SYSTEM CONFIGURATION. Has been installed under George 3, George 4, Executive, MAXIMOP, and COOP operating systems. Requires 36K; uses CR, DA, LP files. (Source library facility only possible, and diagnostics package only practicable under George 3 or 4.)

4. DISTRIBUTION. Free--send 9-track 1600 bpi PE or 7-track 556 bpi NRZI tape to Belfast.

5. DOCUMENTATION. Belfast's Users' Guide (supplement to Pascal User Manual and Report (Revised edition)) 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, United Kingdom (who also produced the Diagnostics package).

6. MAINTENANCE. No formal committment to maintenance. No plans for development in near future. Send bug evidence to Belfast, and also a note of the bug to PCHICL (see notice under ICL--Introduction) who circulate the bug reports and fixes to their members.

7. STANDARD. The level 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 and functions cannot be passed as actual parameters; The correct execution of programs which include functions with side effect is not guaranteed; Only the first 8 characters of identifiers are significant; sets are limited to x..y where 0<= ord(x) <= ord(y) <= 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 functions and procedures include: DATE, TIME, MILL, HALT, CARD; procedures ICL, ADDRESSOF allow use of inline machine code.

8. MEASUREMENTS. Compares favorably to Fortran, requiring about 32K to compile. Generated code is better than that produced by the old 1900 Pascal compiler. (* Compilation speed not reported. *) Performance is better than most other ICL 1900 language processors (exceptions are incore compile-and-go batch systems of the WATFOR type).

9. RELIABILITY. Reported to be good. The compiler is in use at about 50 sites. (* Date first released not reported. *)

10. DEVELOPMENT METHOD. This compiler resulted from a complete rewrite of the old ICL 1900 compiler, which was bootstrapped from the CDC 6000 Zurich compiler. The new compiler is designed for portability, with a clean separation between semantic analysis and code generation. The compiler is about 14,000 lines of Pascal plus about 3500 lines of assembler code and produces absolute binary machine code. The post-mortem analysis program is about 2500 lines of Pascal.

11. LIBRARY SUPPORT. Allows access to Fortran routines.

ICL 2900 (Southampton)

0. DATE/VERSION. Checklist last updated 77/11/4.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Implementors: Project Supervisor: Dr. M. J. Rees;

Computer Studies Group; Faculty of Mathematical Studies; The University; Southampton, S09 5NH; England, U. K. 0703/559122 x2270. Implementors: J. J. M. Reynolds; Computer Centre; Queen Mary College; University of London; Mile End Rd.; London, E1 4NS; England, U. K.; 01 980 4811 x778 and H. J. Zell (deceased). The Pascal compiler will be distributed as a standard ICL program product. Contact the nearest ICL sales office or the Project Supervisor above.

2. MACHINE. ICL 2960, 2970, 2980 series.

3. SYSTEM CONFIGURATION. VME/B and VME/K.

4. DISTRIBUTION. Contact the nearest ICL sales office or the Project Supervisor above. (* No information provided on cost, tape formats, etc. *)

5. DOCUMENTATION. Standard ICL manuals will be available: a) Pascal Language Manual: operating system independant aspects of the Pascal language. b) running Pascal Programs on VME/B and VME/K: information on how to run Pascal under the operating system.

6. MAINTENANCE. Full maintenance will be provided by the implementation group and/or ICL while the compiler is offered as an ICL product. The usual ICL procedure for bug reports will be adopted.

7. STANDARD. The compiler implements "all" [sic] features of the language as described in Pascal: User Manual and Report.

8. MEASUREMENTS. Code generated is fairly compact, the compiler itself producing 80000 bytes. This is better than the 2900 standard compilers. The (CDC) Pascal 6000 compiler compiles the 2900 compiler on a CDC 6400 in 82 seconds. The ICL compiler self-compiles on the 6400 in 100 secs. Running on a 2900, the 2900 compiler self-compiles in 360 seconds.

John Reynolds tells us, "I've determined that almost all time required for a compilation on the 2900 is just burnt up by the system and that hardly any time at all goes in the actual act of code generation." (* 77/7/8 *) (* Execution speed of generated code not reported. *) The source listing is approximately 10,000 lines of Pascal and produces 80k bytes of code. Approximately 160k bytes of store are required to compile the compiler.

9. RELIABILITY. The compiler has been extensively tested and seems to work fairly well. Current (* 77/12 *) reliability is moderate to good. (* Date of first release and number of sites using system not reported. *)

10. DEVELOPMENT METHOD. 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 compiler was bootstrapped using the 1900 compiler from Queen's University of Belfast as a base. Twenty-four person-months of effort from experienced programmers were required.

11. LIBRARY SUPPORT. 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.

IMSAI VDP-40

See Intel 8080.

Intel 8080 Ann Arbor

We have received no new information on this implementation since that which we published last year in Pascal News issue: #12: 64-66.

Intel 8080 INSITE

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 102.

Intel 8080 (Minneapolis)

A 25-page report on "Tiny Pascal", a cross-compiler for a greatly restricted variant of Standard Pascal which is written in CDC 6000 Zurich Pascal and produces machine code for the Intel 8080 is available from: Tiny Pascal Project; Peter H. Zechmeister; University Computer Center: 227 Exp Eng; University of Minnesota; 208 Church St.; Minneapolis, MN 55455 (612/373-4181).

Intel 8080 Munich

We have received no new information on this implementation since that which we published last year in Pascal News issue: #12: 66.

Intel 8080 Stanford

We have heard reports that there is an implementation of Pascal for the Intel 8080 microprocessor that has been developed at Stanford University (Stanford Linear Accelerator Center), but the only information we have received on it is that in the letter under IBM 360/370 Stanford in this issue.

Interdata 7/16 San Diego

We have received no new information on this implementation since that which we published last year in Pascal News issue: #12: 67.

Interdata 8/32 Manhattan, Kansas

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 103-104.

Interdata 8/32 San Diego

We have received a copy of two reports (* dated 78/5/2 *) on cross-compilers for Sequential and Concurrent Pascal which run on the Univac 1100 series and produce code for the Interdata 8/32. These reports, titled "Pascal-V 1.0" and "Concurrent Pascal--V 1.1", are available from: Mike Ball; Code 632; Naval Ocean Systems Center; San Diego, CA 92152; 714/225-2366.

Intel 8080a UCSD

See DEC LSI-11 UCSD.

ITEL AS/4, AS/5

See IBM 360/370.

Marinchip Systems M9900

Marinchip Systems
computer hardware and software
16 Saint Jude Road
Mill Valley, Ca. 94941
(415) 383-1545

Marinchip 9900 Sequential Pascal

Implementation Checklist

1. Distributor/Implementor/Maintainer.

John Walker
Marinchip Systems
16 St. Jude Road
Mill Valley, CA 94941 (415) 383-1545

2. Machine.

Texas Instruments TMS9900. This system runs on the M9900 CPU, which adapts the TMS9900 to the S-100 (Altair/IMSAI/etc.) bus.

3. System configuration.

Runs under Marinchip Disc Executive. Minimum configuration to compile compiler is 56K bytes main memory and one IBM-compatible floppy disc drive.

4. Distribution.

Pascal is available to purchasers of the M9900 CPU board for $150. The system is distributed on an IBM-compatible floppy disc in Disc Executive format.

5. Documentation.

Documentation supplied is a supplement to Per Brinch Hansen's book, The Architecture of Concurrent Programs, and his Sequential Pascal Report. The documentation is in machine-readable form.

6. Maintenance policy.

Bug reports accepted from purchasers of the system. Fixes are available at reproduction cost. System is brand new: no maintenance track record.

7. Standard.

Based upon Per Brinch Hansen's Sequential Pascal, so all comments in the Pascal Variants section about that compiler apply to this one too. The lexical scanner has been modified to permit identifiers to be upper and lower case (case does not affect matching), to accept curly brackets for comments, and square brackets for subscripts and sets. Sequential Pascal syntax still accepted as before.

8. Measurements.

The M9900 permits use of either 8 bit memories or 16 bit memories. With 8 bit memories, the memory cycle time is 3 us, and with 16 bit memories, the cycle is 1 us. Which kind of memory is used has a radical effect on performance. With 8 bit memories the compile speed is about 44 significant characters per second, and with 16 bit memories, the speed is about 130 characters per second. No good benchmarks have been run to judge execution speed. Based on the performance of the original PDP-11 system and comparison of the PDP-11 and 9900 interpreters, we expect performance to range between 25% and 50% of native machine speed based upon instruction mix.

9. Reliability.

No extensive testing of the system has been done by users. However, since the compiler has been compiled through itself without problem, the system is felt to be quite stable.

10. Development method.

The system was bootstrapped from the PDP11/45 version of Sequential Pascal. The interpretive object code was loaded onto the 9900 system, and an interpreter was written for the interpretive code. Rather than implement the entire Solo operating system with which the compiler is shipped, an interface was developed to convert Solo calls into calls on the Marinchip Disc Executive. The execution environment of a Sequential Pascal program is completely simulated. The compiler root segment and seven passes were then compiled through the compiler. The code interpreter and operating system interface total 3000 lines of 9900 assembly code. The compiler was transported and brought up in less than one man-month. The implementor has previously written and moved numerous compilers, but this was the first work on Pascal.

11. Library support.

Separately-compiled Sequential Pascal programs may call each other, passing up to 9 arguments of type INTEGER, BOOLEAN, POINTER, or IDENTIFIER (12 character array of CHAR). The program is loaded coresident with its caller, executed, and a completion status is returned to the caller (termination type and source line). Program calls may be recursive, and nesting depth is limited only by available memory and a configuration parameter. A utility program may be called either from the user terminal, or from another program.

12. General comments

The Sequential Pascal compiler was found to be excellently documented, very reliable in our tests, and extremely easy to move. The current 9900 system is source and object compatible with the PDP11 version. Efficiency considerations may force divergence from the current object code compatibility.

MITS Altair 680B

See Motorola 6800 St. Paul.

Mitsubishi MELCOM 7700

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 104-105.

MOS Technology 6502 (Parksley, VA)

Stephen P. Smith; P. O. Box 841; Parksley, VA 23421; 804/665-5090 is working on a Pascal system for the MOS Technology 6502 chip (using the Ohio Scientific Industries Challanger I system). The system will originally be the minimum subset of Pascal needed to write its own compiler. The original version will cross-compile on any machine which supports a full standard Pascal compiler. The compiler will then convert itself to 6502 machine code and further revisions will then be written in the Pascal subset resident on the 6502. As of 77/12, the parsing procedures were completed and undergoing testing on a DEC-10.

MOS Technology 6502 UCSD

See DEC LSI-11 UCSD.

Motorola 6800 St. Paul

We have received no new information on this implementation since that which we published last year in Pascal News issues: #9-10: 105. #11: 102.

Motorola 6800 UCSD

See DEC LSI-11 UCSD

Motorola 6809

See Motorola 68000.

Motorola 68000

See also Motorola 6800.

Computer Weekly reported on 78/9/7: "Giving further credence to the view that Pascal could become the dominant high-level language of microcomputing, Motorola Semiconductor has revealed that this software will be the prime language supported by its new processor, MACS, due to be unveiled early next year.

"As an intermediate upgrade to MACS, Motorola will also be offering Pascal on its existing 6809 processor chip. The language is already available for the 6800 family from an independant source.

"MACS, the Motorola Advanced Computer System, is expected to see the light of day early next year, and to show its lineage with the 6800 family, will probably be officially known as the 68000."

Nanodata QM-1 California

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 105.

NCR Century 200

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 105.

Norsk Data NORD-10 CERN

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 106.

Norsk Data NORD-10 Oslo

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 106.

North Star Horizon

0. DATE/VERSION. Summer 1978.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. North Star Computers; 2547 Ninth St.; Berkeley, CA 94710; 415/549-0858.

2. MACHINE. North Star Horizon Z-80 based system.

3. SYSTEM CONFIGURATION. Requires 48K of RAM and the Micro Disk System.

4. DISTRIBUTION. $49 including software on diskette and complete documentation (* not known if this is machine retrievable *).

5. DOCUMENTATION. (* No information reported. *)

6. MAINTENANCE. (* No information reported. *)

7. STANDARD. The system is an implementation of UCSD Pascal, which varies from Standard Pascal.

8. MEASUREMENTS. (* No information provided. *)

9. RELIABILITY. (* No information provided. *)

10. DEVELOPMENT METHOD. (* No information provided. *)

11. LIBRARY SUPPORT. (* No information provided. *)

Northwest Microcomputer Systems 85/P

Northwest Microcomputer Systems; 121 East Eleventh; Eugene, OR 97401; 503/485-0626 offers the Northwest 85/P; a self-contained Intel 8085 based microcomputer which includes 2 double density full size Shugart floppy disks (1 Mbyte online), 54K of 450ns Static RAM (I/O, etc. in PROM), Hall effect typewriter keyboard with numeric pad and 29 user definable function keys, 24 line 80 character 12"(30 cm) Video RAM display, 2 serial ports and 16 parallel ports. The basic system includes with the hardware the CP/M operating system and the Pascal system for $7,495. The Pascal compiler/interpreter runs at 725 lines/min and "provides the full Pascal environment", including random and sequential files, screen-oriented editior, interactive source linked debugger, and full documentation.

Ohio Scientific Industries Challanger I

See MOS Technology 6502.

Prime P-300 and P-400 Hull

THE UNIVERSITY OF HULL
HULL HU6 7RX. ENGLAND
Telephone: Hull 46311

Department of Computer Studies

30th August, 1978

Dear Andy,

We're enclosing a fuller set of notes for our implementation of PASCAL on a PRIME 300. The work is now almost complete and we're very pleased with the result.

We have appended some extra sections to the notes. One of these deals with other implementations on PRIMEs and provides a brief summary of the information we hold on them. Unfortunately we can't do a comparison of all implementations since the Georgia Tech. version only runs on a PRIME 400.

Thanks again for your work with "PASCAL News".

Yours sincerely,

Ian

Barry Cornelius.
Ian Thomas.
Dave Robson.

THE UNIVERSITY OF HULL'S PASCAL COMPILER
FOR PRIME 300 COMPUTERS

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: Developed on a PRIME 300 but will also run in 64R mode on a PRIME 400.

3. SYSTEM CONFIGURATION: The PRIME 300 currently has 64K words running under PRIMOS-3 Revision 10.

4. DISTRIBUTION: Two versions of the compiler have been released to PRIME (U.K.) for evaluation and testing. It is hoped to have a distribution arrangement agreed with PRIME in the near future.

5. DOCUMENTATION: A 30 page manual describing the PASCAL system is available in machine-readable form. It includes instructions on how to build a new compiler

6. MAINTENANCE POLICY: This will depend partly on the agreement with PRIME - nevertheless we intend to correct reported errors for the next few years.

7. STANDARD: The PASCAL-P variant of PASCAL is implemented.. Some of its restrictions have been removed and some extensions have been added. The extensions include external procedures (see 11 below) and an initialisation facility for variables in the outermost block.

8. MEASUREMENTS: When range-checking code is produced the compilation speed is approximately 550 characters/second. When code with no checks is required the speed is approximately 650 characters/second which is the same as FORTRAN's compilation speed (without trace or checking).

PASCAL input/output is considerably superior to FORTRAN's input/output. A text copying program takes about 4 times longer to execute in FORTRAN than PASCAL.

We do not have any comparisons for processor-bound programs since no-one can be persuaded to write a sufficiently large program in FORTRAN! However, we would expect PASCAL to be slower than FORTRAN since little optimisation of the code is currently performed.

9. RELIABILITY: The compiler is very reliable and will reach a stable state by September 1978. It is hoped that the first release will then be available. As stated in 4 above, a preliminary release of the compiler is currently available on PRIME (U.K.)'s demonstration machines.

The Run-time Support and the input/output routines have been designed so that, when an execution time error occurs, an error number is output together with a "wordcount". The wordcount is the address relative to the start of the program of the instruction causing the error. The value of the wordcount appears at the start of each line of the compilation listing and so the error can be traced to the line of the source program at which the error occurred.

10. DEVELOPMENT METHOD: The code generation sections of the PASCAL-P compiler have been extensively rewritten to generate 64R mode PMA. It is a true compiler rather than a compiler/interpreter system or a threaded code interpreter. The compiler is now some 6000 lines and compiles itself (without a compilation listing) in 300 C.P.U. seconds on the configuration described in 3 above. The first version of the compiler was developed from the PASCAL-P compiler on the University's ICL 1904S using the Belfast Mk.2 compiler.

11. LIBRARY SUPPORT: Calls of external procedures are permitted. The parameter-passing protocol is a superset of that used by PRIME's standard system routines.

12. OTHER IMPLEMENTATIONS: There are a number of other implementations of PASCAL on PRIME machines. Some of these are described in more detail in an article we wrote for the Bulletin of the European PRIME Users Association, (see "PASCAL", E.P.U.A. Bulletin, Volume 4, Issue 1 (June 1978)).

(i) Per Brinch Hansen's Sequential PASCAL - very slow.

(ii) University of Brunswick's PASCAL compiler. Translates into modified Pcode which is subsequently optimised and translated into relocatable binary. The code produced contains calls to routines to perform Pcode instructions and it is thus a threaded code system. Compilation takes approximately 3 to 4 times as long as the University of Hull's implementation but the translation into relocatable binary is very much faster than the assembly of the PMA that our implementation produces.

(iii) Georgia Tech's PASCAL compiler. The compiler was developed for a PRIME 400. From "PASCAL News" #12 the current version appears to be a threaded code interpreter.

13. FUTURE PLANS: It is likely that we will implement translation into relocatable binary in the near future. The additional compilation time overheads will probably be offset by the reduction in the amount of character input/output currently necessary to output PMA text.

More of the restrictions of the PASCAL-P subset are also likely to be removed. It is possible that we will implement the post-mortem dump facility written (in PASCAL) by Glasgow University for the ICL Belfast Mk.2 compiler.

Prime P-400 Atlanta

We have received no new information on this implementation since that which we published last year in Pascal News issues: #9-10: 106. #12: 67.

Processor Technology SOL

According to Ralph I. Palsson, Customer Applications Manager: Processor Technology Corp.; 7100 Johnson Industrial Dr.; Pleasonton, CA 94566; 415/829-2600: "We do not currently (* 78/1/11 *) have any intentions of providing Pascal. We will be providing a FORTRAN compiler this spring as well as PILOT... Providing good software support for users of Processor Technology hardware is one of our primary committments [sic]. As of this time, there has been relatively little demand for Pascal. Consequently our software emphasis has been in other areas."

According to S. M. Sokolow, Editor; Solus News; 1690 Woodside Rd. 219; Redwood City, CA 94061 (* 78/10/13 *): "We're in the process of preparing to distribute the Stanford Linear Accelerator Center's implementation of P-code Pascal for the SOL with Helios disk."

Radio Shack TRS-80

See also Zilog Z-80.

According to Hugh Matthias, Radio Shack, 205 NW 7th St., Fort Worth, TX 76101; Radio Shack does not intend to produce a Pascal system for the TRS-80 now or at any time in the future. "It appears to be to [sic] costly--ever!" (* 77/11/19 *).

RCA Spectra 70

See Siemans 4004, 7000 and Univac 90/70.

SEL 8600

Jim Gilbert; Systems Structuring Technology; 30436 N. Hampton Rd.; Laguna Niguel,CA 92677; 714/640-5222 (work); 714/495-6039 (home) reports (* 78/9/30 *): "I am the implementor of the SEL 8600 & SEL 32 P2 Pascal mentioned in Pascal News #4. [Co-implementor Michael] Richmond is with D.G.C. in Carolina last I knew. I am available on a contract basis for language consulting."

SEMS T1600 Nancy, France

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 106.

Siemens 150 and 330

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 107-108.

Siemens 4004, 7000 Munich

SIEMENS PASCAL BS2000 PROGRAMMING SYSTEM

0. DATE/VERSION: 78/10/01 Version 2.0
For version 1 see #9/10 : 108

1. Distributor/Implementor/Maintainer
Dr. M. Sommer
SIEMENS AG Dep: D AP GE 1
Otto-Hahn-Ring 6
D - 8000 München 83
Germany

2. Machine:
SIEMENS series 4004 and series 7000

3. System configuration:
all systems under operating system BS2000 (>= rel.3.0)

4. Distribution
- please contact implementor

5. Documentation
Machine retrievable user manual

6. Maintenance Policy
- please contact implementor -

7. Standard.
Standard PASCAL is accepted.
Extension: Sets of any range (maxelements: 2048)
are implemented by minimal byte-strings, separate
compilation of PASCAL, FORTRAN,-procedures and PASCAL-
Modules many additional standard procs.
- compiler cptions, like optimise, xref, debug, codelist, etc.
- compiler instructions like copy from include-lib, skip.

8. Measurements: (For a SIEMENS 7.755)
Compilation speed: 3200 chars (incl. blanks) /second
140 lines/second
(*speed is depending on options/listings*)
Execution speed and execution space of an
average of 6 test programs including prim, queens,
palindromes, quicksort. etc.

	PASCAL version 2	PASCAL version 1	other language
SPACE (bytes)	326	580	446
TIME (sec)	4.2	7.8	5.2

Telefunken TR440

0. DATE/VERSION
   78/06/01, Version 36

1. DISTRIBUTOR/IMPLEMENTOR/MAINTAINER
   H.D. Petersen
   Institut für Informatik
   Azenbergstrasse 12
   D-7000 Stuttgart 1
   Germany
   Phone: (0711) 2078 376

2. MACHINE
   TR440

3. SYSTEM CONFIGURATION
   BS3, MV ≥ 18

4. DISTRIBUTION
   Send magnetic tape, 9 track, 800 bpi. Object module library; source files for reference only.

5. DOCUMENTATION
   Manual in German (preliminary)

6. MAINTENANCE POLICY
   Bug reports welcome; no commitment for maintenance yet.

7. STANDARD
   Full standard is implemented.
   Extensions:
   - separate compilation of procedures and modules
   - external procedures
   - large sets (max. 624 elements)
   - set of char possible
   - random access files
   - interactive I/O
   - packed structures
   - several machine-oriented facilities
   - dynamic arrays (in preparation)

8. MEASUREMTS
   - compilation space > 160 kbytes.
   - compiles itself in 320 sec using 230 kbytes.
   - execution speed between ALGOL and FORTRAN.
   - I/O faster than ALGOL or FORTRAN.
   - run-time system needs up to 18 kbytes depending on features used.

9. RELIABILITY
   No information yet available; previous version delivered to 15 installations, moderately stable.

10. DEVELOPMENT METHOD
   Compiler derived from P-2; new version has two passes coupled by extended SC-code. Approximately 9000 PASCAL lines total; run-time system in Assembler (TAS).

11. LIBRARY SUPPORT
   Separate compilation and linkage to FORTRAN, ALGOL and Assembler available. Full error messages in source listing. Options: cross-referencing, intermediate code listing. Run-time error messages keyed to source line and call hierarchy. Symbolic post-mortem dump for all data types, including heap objects, scalar types and records. Text inclusion into source in preparation.

9. Reliability
   is hoped to be excellent as the reliability of version 1 is excellent. (Used by ca. 30 sites.)

10. Development method
   Developed from version 1 (developed from PASCAL P).
   New codegenerator - and other extensions.
   Length is ca. 14000 lines of compact PASCAL.
   Effort (Version 1→Version 2) ca. 20 MM.

11. Library support
   a) Standard modules from Assembler, Fortran, Cobol
   b) Procedures and modules written in PASCAL
   Standard Linkage. Copy from libraries in Source.

SOLAR 16-05/40/65

See SEMS T1600.

SouthWest Technical Products

See Motorola 6800.

Telefunken TR-440

===>

TERAK 8510, 8510A UCSD

See DEC LSI-11 UCSD.

Texas Instruments TI-ASC

The only new information we have received on this implementation since that which we published last year in Pascal News issue: #9-10: 109 is the rumor that the implementation is being done by the Advanced Software Technology group and is currently (* 78/2/28 *) in the debugging stage; but that the compiler is probably not intended for outside distribution.

Texas Instruments TI-980a

George Cohn, Wrubel Computer Center, Indiana University/HPER, Bloomington, IN 47401, (812) 337-1911, has had a Pascal version running for quite some time in the Computer Science Department, although no formal distribution arrangements have been made.

Texas Instruments TI-990, 9910 Houston

0. DATE/VERSION

Release 1.4.0, May 1978.

1. DISTRIBUTER/IMPLEMENTATION/MAINTAINER

Implemented by Texas Instruments. Information is available from TI sales offices, or write to:

Texas Instruments
Digital Systems Division, MS 784
P. O. Box 1444
Houston, Texas 77001

or call (512) 258-7305. Problems should be reported to:

Texas Instruments
Software Sustaining, MS 2188
P. O. Box 2909
Austin, Texas 78769

or call (512) 258-7407.

2. MACHINE

TI 990/10

3. SYSTEM CONFIGURATION

Runs under the DX10 operating system (release 3) on a TI DS990 Model 4 or larger system, which includes a 990/10 with 128K bytes of memory and a 10 megabyte disk.

4. DISTRIBUTION

Available on 9-track magnetic tape (either 800 or 1600 bpi) or on a disk pack for a TI model DS10, DS31, DS25, or DS50 disk drive. Contact a TI salesman for a price quotation.

5. DOCUMENTATION

Complete user-level documentation is given in the "TI Pascal User's Manual", TI part number 946290-9701.

6. MAINTENANCE POLICY

TI Pascal is a fully supported product. Bug reports are welcomed and maintainence and further development work are in progress.

7. STANDARD

TI Pascal conforms to "standard" Pascal, with the following principal exceptions:

* Functions cannot alter global variables.
* A GOTO cannot be used to jump out of a procedure.
* The control variable of a FOR statement is local to the loop.
* The precedance of Boolean operators has been modified to be the same as in Algol and Fortran.
* The standard procedures GET and PUT have been replaced by generalized READ and WRITE procedures.

TI Pascal has a number of extensions to standard Pascal, including random access files, dynamic arrays, ESCAPE and ASSERT statements, optional OTHERWISE clause on CASE statements, and formatted READ.

8. MEASUREMENTS

The compiler occupies a 64K byte memory region. Compilation speeds are comparable to the 990 Fortran compiler.

9. RELIABILITY

There are some known problems which are currently being worked on, but none are so serious that they can't be worked around. The system has been used by several different groups within TI since October of 1977, and by a number of outside customers since May of 1978.

10. DEVELOPMENT METHOD

The compiler produces object code which is link-edited with run-time support routines to form a directly executable program. The compiler is written in Pascal and is self-compiling.

11. LIBRARY SUPPORT

TI Pascal supports separate compilation of routines and allows linking with routines written in Fortran or assembly language.

Texas Instruments 9900/4 Vienna

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 109.

Univac 90/30

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 109.

Univac 90/70

See Siemens 4004, 7000 series.

Univac 90/70 Philadelphia

UNIVERSITY of PENNSYLVANIA
PHILADELPHIA 19104

The Moore School of Electrical Engineering **D2**
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

April 20, 1978

Dear Andy,

I just wanted to let you know about the PASCAL 8000 implementation which I recently brought up on our Univac 90/70 (VS/9 operating system).

The system is based on the Australian AEC compiler of Cox, Tobias, Hikita and Ishihata (which is quite an excellent piece of software), and was implemented by modifying the runtime system to interface with VS/9. Only the compile-and-go version has been implemented at this time. All features of the Australian compiler have been retained, and additional support added for some VS/9 features: the system files SYSDTA, SYSLST, SYSOUT, SYSIPT, SYSOPT and * are supported, and a COMMAND function has been added which allows PASCAL programs to issue VS/9 commands, i.e. COMMAND('/ERASE filename'); This implementation will probably not run on Univac Series 70 VMOS without modification, since interrupt handling is done with operating system features that I am told are specific to VS/9.

No formal distribution plans have been made, but anyone who is interested (hopefully with software to trade) should contact me at P.O. Box 8191, Philadelphia PA 19101.

Very truly yours,

G. Kevin Doren

Univac 1100 (Copenhagen)

We have received a copy of a 60 page users manual (* dated 77/8 *) titled "A Pascal Compiler for the Univac 1100 Series" which is available from the implementor.

0. DATE/VERSION. Checklist not updated since 77/08.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. J. Steensgaard-Madsen, DIKU (Datalogisk Institut Kobenhavns Universitet), Sigurdsgade 41, DK-2200 Copenhagen N., Denmark. (* No phone number reported. *)

2. MACHINE. Univac 1100 series.

3. SYSTEM CONFIGURATION. Exec-8 operating system. (* Minimum hardware requirements not reported. *)

4. DISTRIBUTION. The charge for distribution from Datalogisk Institut is Dkr. 200. The distributors are attempting to maintain a distribution tree to reduce costs and hassles. Purchasers must sign a license agreement. The system is released only in relocatable form.

5. DOCUMENTATION. A 19-page machine-retrievable supplement to the Pascal User Manual and Report is available. It is "A Pascal Compiler for the Univac 1100 machines", by J. Steensgaard-Madsen and Henrik Snog of DIKU.

6. MAINTENANCE. There is no promise of maintenance, but bug reports are required under the license aggreement.

7. STANDARD.
Deviations from the standard: Reset(f) on any textfile f will cause eof(f) = false and eoln(f) = true; Parameter types of formal procedures and functions must be specified.
Restrictions: file of file is not allowed; standard procedures cannot be passed as actual parameters.
Extensions: otherwise in case statements; conformant array parameters.
Machine dependencies: Sets may have 72 elements, char is defined as (6-bit) Fieldata, ASCII is an additional type; real is double precision always.

8. MEASUREMENTS. Compilation space is roughly 42K; speed is 100 lines per SUP second. Compiled programs run efficiently compared to other processors.

9. RELIABILITY. Excellent. (* Date first released and number of sites using system not reported. *)

10. DEVELOPMENT METHOD. Pascal-P with a team of 4 persons. (* Person-hours to develop system not reported. *)

11. LIBRARY SUPPORT. External procedures may be written in Pascal or (ASCII) Fortran. Inclusion of assembler code is possible.

Univac 1100 Madison, Wisconsin

We have received no new information on this implementation since that which we published last year in Pascal News issues: #9-10: 110-112. #11: 103.

Univac 1100 (San Diego)

We have received a 33-page report on this implementation titled "Pascal 1100" which is available from the implementor.

0. DATE/VERSION. Checklist not updated since 77/08.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Michael S. Ball; code 632; Naval Ocean Systems Center; San Diego, CA 92152; 714/225-2366.

2. MACHINE. Univac 1100 Series.

3. SYSTEM CONFIGURATION. Exec-8 operating system; can be run in Demand mode.

4. DISTRIBUTION. As a member of USF, you may request a copy from Mike by sending a mag tape and noting any restrictions on it s format.

5. DOCUMENTATION. A machine-retrievable supplement to the Pascal User Manual and Report entitled "Pascal 1100" documents the implementation.

6. MAINTENANCE. (* No information provided. *)

7. STANDARD. Restrictions: entry, processor, and univ are reserved words; standard procedures and functions may not be passed as actual parameters; file of file is not allowed. Sets may have at most 144 elements. The compiler accepts the full ASCII character set. A compiler option allows processing of Brinch Hansen Sequential Pascal programs.

8. MEASUREMENTS. The compiler compiles into 34K words and requires 6K words of library routines. (* Compilation speed not reported. *) Self-compilation requires about 15.5K for stack and heap.

Execution times for code compiled by Pascal was compared with code generated by the NUALG and ASCII FORTRAN processors. Fortran's local optimization was taken as a base value. The programs used for comparison were taken from Wirth's paper on the design of a Pascal compiler (Software - Practice and Experience, Vol. 1 (1971), pages 309-333). The results are summarized in the following table.

		Pascal		NUALG		FORTRAN		FORTRAN
	Pascal	no tests	NUALG	no tests	FORTRAN	local opt.		global opt.
	(rel)	(rel)	(rel)	(rel)	(rel)	(rel)	(time)	(rel)
PART	0.62	0.61	0.85	0.84	1.00	1.00	15.10	0.99
PARTNP	1.18	1.06	3.29	3.17	0.94	1.00	0.93	0.85
SORT	1.37	1.12	1.83	1.49	1.00	1.00	18.01	0.59
MATMUL	1.82	1.43	2.05	1.70	1.00	1.00	10.26	0.39
COUNT	0.30	0.28	0.72	0.66	1.00	1.00	16.83	0.97

9. RELIABILITY. Quite good; it should approach excellent. The system has been in local use since about February, 1976, and it has been installed at 25 sites (11 university, 4 government, 10 industry).

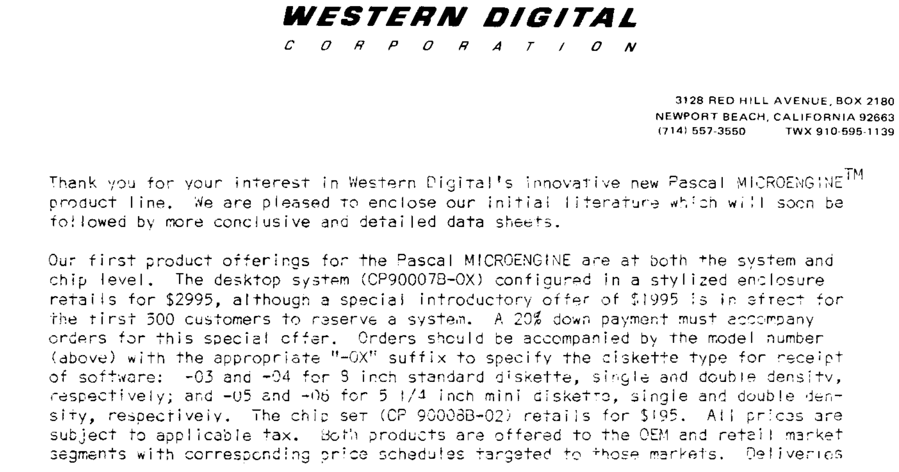
10. DEVELOPMENT METHOD. The compiler was developed from Pascal-P2. (* Person-hours to develop system not reported. *)

11. LIBRARY SUPPORT. Generated code can be linked to subprograms written in Fortran or assembler.

Varian V-70 VOICE

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 112.

Western Digital Newport Beach

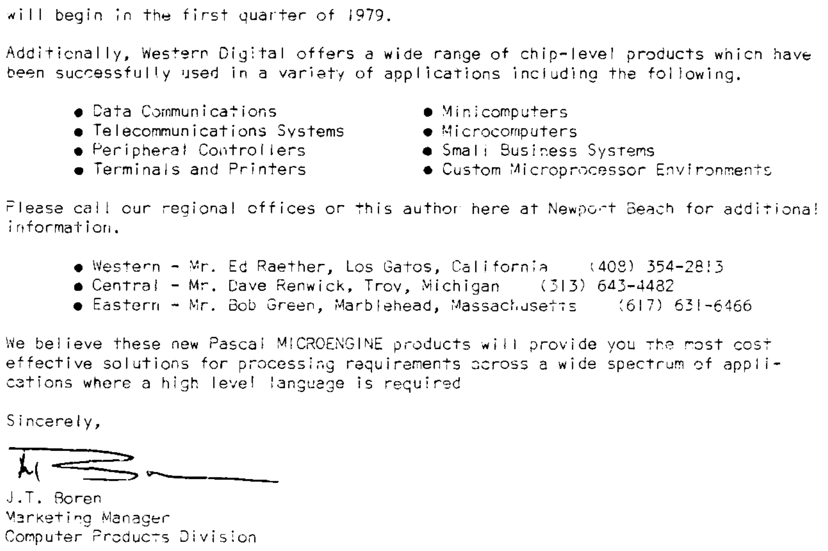


WESTERN DIGITAL
C O R P O R A T I O N

3128 RED HILL AVENUE, BOX 2180
NEWPORT BEACH, CALIFORNIA 92663
(714) 557-3550 TWX 910-595-1139

Thank you for your interest in Western Digital's innovative new Pascal MICROENGINE^{TM} product line. We are pleased to enclose our initial literature which will soon be followed by more conclusive and detailed data sheets.

Our first product offerings for the Pascal MICROENGINE are at both the system and chip level. The desktop system (CP90007B-0X) configured in a stylized enclosure retails for $2995, although a special introductory offer of $1995 is in effect for the first 300 customers to reserve a system. A 20% down payment must accompany orders for this special offer. Orders should be accompanied by the model number (above) with the appropriate "-0X" suffix to specify the diskette type for receipt of software: -03 and -04 for 8 inch standard diskette, single and double density, respectively; and -05 and -06 for 5 1/4 inch mini diskette, single and double density, respectively. The chip set (CP 90006B-02) retails for $195. All prices are subject to applicable tax. Both products are offered to the OEM and retail market segments with corresponding price schedules targeted to those markets. Deliveries



will begin in the first quarter of 1979.

Additionally, Western Digital offers a wide range of chip-level products which have been successfully used in a variety of applications including the following.

- Data Communications
- Telecommunications Systems
- Peripheral Controllers
- Terminals and Printers
- Minicomputers
- Microcomputers
- Small Business Systems
- Custom Microprocessor Environments

Please call our regional offices or this author here at Newport Beach for additional information.

- Western - Mr. Ed Raether, Los Gatos, California (408) 354-2813
- Central - Mr. Dave Renwick, Troy, Michigan (313) 643-4482
- Eastern - Mr. Bob Green, Marblehead, Massachusetts (617) 631-6466

We believe these new Pascal MICROENGINE products will provide you the most cost effective solutions for processing requirements across a wide spectrum of applications where a high level language is required

Sincerely,

J.T. Boren
Marketing Manager
Computer Products Division

Xerox Sigma 6, 9 Quebec

We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 112.

Xerox Sigma 7 Tokyo

See also CII 10070 and CII IRIS 80. We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 112.

Zilog Development System

See Zilog Z-80 UCSD.

Zilog Z-80 Indiana

78/04/07

INDIANA UNIVERSITY
Wrubel Computing Center
MEMORIAL HALL WEST, ROOM
BLOOMINGTON, INDIANA 47401

TEL. NO. 812--337-1911

Andy Mickel
University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

Dear Andy,

Over the past several months I have been working on a PASCAL compiler for the Zilog Z80. The project is now at the point where it generates pretty good code. Any further enhancements will have to be done either at a much slower pace or by somebody else.

I started with the P4 compiler and wrote a PASCAL program that translated the PCODE to Z80 assembly mnemonics. I discovered that for the sake of efficiency of space, many of the PCODE instructions have to generate calls to a runtime support library that was growing at a good clip. My next step was to eliminate the Z80 assembler. I incorporated address resolution and an intermediate file into my postprocessor so that it could generate standard Z80 object code. My final step was to move the logic from the postprocessor up into the main compiler, making it one pass. In order to avoid having an intermediate file, it assembles the code for each procedure body in a chain of arrays allocated from the heap. Thus, the largest procedure determines the amount of memory needed by the compiler. The compiler itself still thinks it is generating PCODE. I have undermined procedures GEN0, GEN1, GEN2, GEN0T, GEN1T, GEN2T, and a couple of others so that instead of printing PCODE mnemonics onto the object file, they call Z80 code generating routines which do the real work. As a bonus, I now have actual Z80 addresses alongside my listings, making breakpointing convenient.

My compiler is probably doomed to remain a cross compiler for the duration of its existence. Its output code cannot hope to compete with interpreter code for efficient utilization of space. However, the runtime support package takes up about 4K bytes, and a good interpreter that used that package extensively could probably be written in a few hundred bytes. The compiler could be remodified to generate a binary interpreter code in Z80 loader format. Then some really big programs could be run on the Z80, perhaps even the compiler itself. But that is another project.

I have made some little white modifications to the compiler to make it more convenient to the microprocessor programmer. These include the following:

1) Files. Not really an extension, since they are part of the language definition. In the current implementation, file names actually serve as device designators.

2) Nondescriminated variants. Also standard PASCAL.

3) ASCII coding. CASE statement constants and SET constants based on characters are translated to ASCII. This is absolutely necessary for cross compilation on non-ASCII machines.

4) External procedures and functions. These must be accompanied by an absolute address, since we have no relocating loader.

5) Hex and octal in the source code. 256, for example, is represented as 100H for hex and 400Q for octal.

6) Hex output on textfiles.
Example: WRITE(OUTPUT,A: 4 HEX);

Under the present configuration, characters range from 0 to 255, sets may contain up to 128 elements, integers are 32 bit two's complement, and reals are not yet implemented. I was once tempted to cut integers down to 16 bits, which would work wonders for efficiency, but Al Towell talked me out of it. What can you do with a 16 bit integer?

PASCAL can provide a convenient medium for applications systems with the help of assembly language procedures for tight spots. We seem to be getting into Z80s more and more at the Wrubel Computing Center, so I am confident that the code I worked on will see plenty of action on the front lines.

I am sending a listing of my runtime support package and listings of a couple of compilations for you to look over. Tell me what you think.

Respectfully,

George

George Cohn III

Zilog Z-80 UCSD

See also DEC LSI-11 UCSD.
We have received no new information on this implementation since that which we published last year in Pascal News issue: #9-10: 112.

Zilog Z-8000

It is believed that many of the present Zilog Z-80 Pascal systems could be easily modified to run on the Z-8000; since Zilog says "using an automatic translator, present users of the Z-80 can easily convert to the Z-8000, since the Z-8000 instruction set is, in effect, a superset of the Z-80 instruction set." We would appreciate hearing from anyone who has made such a conversion; or from anyone who has developed a Pascal system directly for the Z-8000.

INDEX TO IMPLEMENTATION NOTES

General Information

Checklist

Portable Pascals

Pascal Variants

Feature Implementation Notes

Machine Dependent Implementations

POLICY: PASCAL USER'S GROUP (78/10/01)

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 ana 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
1978 (corrected printing), 167 pages, paperback, $6.90.

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 2712 active members in more than 41 countries. This year Pascal News is averaging more than 120 pages per issue.