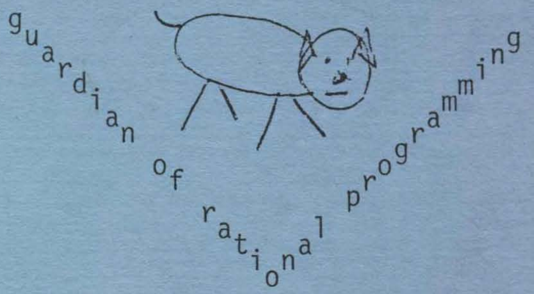
PASCAL USER'S GROUP

Pascal News

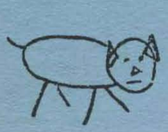
NUMBER 15

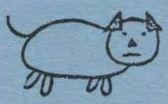
COMMUNICATIONS ABOUT THE PROGRAMMING LANGUAGE PASCAL BY PASCALERS

SEPTEMBER, 1979



P. U. G.


1976, U.S.A.


1977, Europe

1977, Australia

....

POLICY: PASCAL NEWS (79/09/01)

* 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 for this issue (#15) were:

Rick Marcus, Andy Mickel, Jim Miner, Arthur Sale, and Rick Shaw.

(Rick Shaw and Arthur dropped into Minneapolis to save the day!)

Thanks for not giving up hope ...

Pascal News is alive and well !

Well, everyone, it's been a real struggle to get this issue done in spite of the delays over the last 6 months. Unfortunately we've caused some confusion. Please note:

THIS ISSUE (#15) AND NEXT ISSUE (#16) STILL APPLY TO 78-79 SUBSCRIPTIONS!!!

In other words, if your mailing label says "RENEW JUNE 79", your subscription has not expired yet. Further, our policy states that if you join PUG anytime during an academic year ending June 30, we will send you all 4 issues for that year. Well now, I'd like to point out that we are still in the 78-79 academic year (!), and that all new subscriptions are being forced to that period. Why? I expect you new members want the latest information that's available (such as this issue), and this is a 78-79 issue.

Therefore whereas we say in the policy that we attempt to publish September, November, February, and May issues, for 78-79 subscriptions we will have had December, January, September, and October issues. 79-80 subscriptions will start with a November issue (#17). We'll get back on track eventually (I hope!). I'm sorry for the confusion.

Now let me try to explain what happened:

Volunteers do the work on Pascal News. As anyone in computing these days knows, talent (or even mere bodies) are hard to find. With Jim Miner absorbed in standards activities and everyone else hard at work at regular jobs, it's been just Rick Marcus and myself holding things down. In fact from 79/01/22 to 79/04/15, mail piled up unopened, and we were still delinquent in sending out some backissues ordered since 78/11/08! So if you are a new member who joined during this period (nearly 800 of you!), you were the victims of unacceptably bad service. I apologize. By 79/05/15 we had processed the mail and mailed out backissues, which in some cases took 1 more month (79/06/15) to arrive.

However, the next urgent task was to tidy up the PUG files (about 10000 ALL-PURPOSE COUPONS) and update the accounting since we let things go back in May, 1978. It was actually back then that our troubles began, because one article publicizing Pascal and PUG in ComputerWorld generated 500 new members in one month (or a 25% increase in membership in one single month!) We have only recently fully recovered. This summer Rick and I spent one month completely straightening the files. Straightened files (very important) allows us to process new memberships and renewals faster, because we can eliminate duplicates and follow up questions about membership status, lost and uncashed checks, etc.

Finally on 79/08/28, I processed all subscriptions (approximately 450) from 79/05/16 onward and mailed backissues. Only then did we begin looking at Pascal News #15 seriously.

Thanks a lot for your faith and patience--miraculously we've received zero requests for refunds, and only 10 requests regarding what is happening. When I said in #13 that I was quitting effective anytime after July 1, 1979, I was intending to do the 2 issues remaining for 78-79, and #15 and #16 represent the followthrough on that commitment. Some people thought that #13 was my "swansong."

-Andy

Editor's Contribution

About This Issue

As I said on the previous page, it's been a real struggle to get this issue of Pascal News produced. It was a hard task to face, too! Foremost is the fact that we were behind in processing the ever-increasing volumes of mail with fewer and fewer volunteers. Next, event surrounding standards activities effectively sapped all our energy (or so it seems!). Also with the uncertain future of Pascal News and PUG, lots of time was spent discussing "solutions." I found it really depressing to continue to have to cooperate with certain people and performing certain activities (e.g. someone suggesting some grand future for PUG such as a constitution and then requiring me to do all the transition work to implement it) that I don't like nor believe in. I still have my regular job to do here at the comp center

Anyway, good news! With the help of Rick Marcus, and in the last week the air-borne reinforcements of Arthur Sale, Rick Shaw, and a work-liberated Jim Miner, we were able to deal #15 a knockout blow. The next issue (#16) will be a special one on the Validation Suite (see below) and my last one as editor. #16 should appear very shortly after this issue and wrap up the 78-79 academic year.

The Future of Pascal News and PUG

(*Please see related correspondence in the Open Forum section.*)
When we last left you, I had written an editorial and an open letter in #13 saying that I was quitting the editorship of Pascal News and my work informally coordinating Pascal User's Group, and that basically there were 4 alternative futures for consideration. One of these was a proposed constitution provided by Richard Cichelli which included a ballot to be returned by April 15, 1979.

I claimed then that the constitution was probably the best alternative, and that the least likely alternative was to keep PUG the same, but to decentralize the work.

I guess I was really wrong!

Rick Shaw (to whom ballots were to be sent) tabulated 56 votes in favor, 22 votes agains and 2712 abstentions of the 2790 active members. 5 of the yes votes dissented on the by-laws. Some comments written-in included: the constitution effectively shuts out international members; affiliation with IEEE or ACM SIGPLAN was the best alternative. More than a dozen of the "no" votes were in favor of disbanding PUG altogether.

In spite of their promises Steve Zilles (SIGPLAN Chairman) and Bruce Ravenel (on behalf of IEEE) did not send us letters to print for our consideration proposing how we might affiliat with them, much less inviting us to do so. So much for ACM and IEEE.

I happened to go with Jim Miner to my first IEEE P770 / ANSI X3J9 Joint Pascal Standards meeting in Boulder the last week in April, and met many people with whom I discussed PUG's future (besides explaining our terrible workload, etc.!). The feeling by-and-large was that they wanted to see a good thing like an independent PUG continued, and that they had voted for the constitution because they way no other real choice, but ideally they would like to see PUG continued as it is now.

There followed one of those smoke-filled-room meetings in one of the hotel rooms among Jim Miner, Scott Jameson, Rick Shaw, Rich Cichelli, and others (but not myself!) in which a heated (and smoky!) argument raged for over 4 hours. The result was the expansion of David Barron's idea by Jim Miner: the realization that the only important activity of PUG is the publication of Pascal News. Several people responded to Jim's initiative (see Open Forum), and the best news was that Rick Shaw volunteered to take over as editor and informal coordinator of Pascal User's Group for 2 years. Rick is a capable administrator (whereas I am not good at delegating responsibility), and he has the luck of being in a nice work environment at DEC's Atlanta Regional Office with ready access to clerical facilities, etc.

We then realized that PUG could continue informally without a constitution and other politic baggage. The constitution vote could then be thrown safely out--after all, 97% of the membe did not vote! The last step was to actively decentralize the work so that Rick could avoid drowning quickly. We then started to recruit more section editors for Pascal News. The lis of new volunteers now looks like this: Rick Shaw - editor; Bob Dietrich and Greg Marshall - Implementation Notes editors; John Eisenberg - Here and There editor; Rich Stevens - Books and Articles editor; Andy Mickel and Rich Cichelli - Applications editors; and Tony Addyman and Jim Miner - Standards editors. Rick will simply forward material to them which they in turn will convert to camera-ready copy and return to Rick for paste-up. Meanwhile part of the subscription money to Pascal News will go to pay for clerical work (under Rick) for the mailing-label data base, word-processing tasks, printing, mailing, etc. Atlanta is the home of Georgia Tech and Georgia State University with whom Rick has close ties.

We even got offers from the following people and organizations who have expressed the ability to help Pascal News in some material way: John Knight at NASA Langley, Rusty Whitney at Oregon Software, Marius Troost at Sperry Univac Minicomputer Operations, and Don Peckham at Pertec. So the future is bright.

Frankly, at the present time it appears that Pascal News can be viable for only 2 or 3 more years. With the explosion in Pascal interest, the phrase "lingua franca" is often heard in reference to Pascal. The obvious implications of lingua franca are that events surrounding Pascal will be covered thoroughly by every other computing journal and so will take over the role of Pascal News.

In summary, we saved Pascal News and PUG from the near political demise foisted on us in 1978 when the constitution idea was born. We'll have an informal PUG with no constitution by golly, or we'll have a constitution with no PUG! We've just altered the policy pages in Pascal News to protect ourselves from constitutions and politics in the future.

Jottings

Pascal Standards The BSI/ISO standard's progress, with productive and valuable American cooperation, has been remarkable and encouraging, proving those who have claimed such an effort would take at least 5 years dead wrong. See Standards in the Open Forum section.

Pascal Validation Suite A new feather in Pascal's cap is the existence of a professionally produced Validation Suite of test programs to verify the standards-conformance, etc. of a given Pascal compiler. The collection of 300+ programs can be used by implementors and users alike to help enforce standards. See Standards in the Open Forum section. Pascal News #16 will be entirely devoted to the Validations Suite.

Defective copies of Pascal News #14 At least one person has reported that his issue of Pascal News is missing pages 6-14 and has pages 15-22 duplicated. If you are suffering from the same problem, let us know and we'll help.

Eurocheques David Barron sent along this note to European subscribers: "From time to time we are asked why we will not accept "Eurocheques", i.e. sterling cheques drawn on the subscriber's local bank. The answer is simple. A Eurocheque for £4 yields less than £3 to the PUG bank account. The difference, more than 25%, is the charge made by our bank for processing the Eurocheque. So please ask your bank for a draft drawn on a U.K. or Irish bank, or pay by direct transfer into our Post Giro account (28 513 4000)."

Pascal on Micros A large number of people have been complaining to us over the last year about our blind praise and support for Ken Bowles and his group's widespread Pascal interpreter for various micros popularly known as UCSD Pascal. They are expressing reservations about the lack of reliability and speed and the presence of non-standard features in UCSD Pascal. I'd like to make it clear that we don't blindly support Ken or anyone else even though we've printed some highly favorable items about UCSD Pascal in some past issues. (For some contrast see the checklist for UCSD Pascal in Pascal News #13 under DEC LSI-11.) Ken Bowles was one of the people who helped in the middle stages of Pascal's acceptance in this country. I might add that increasingly there is a trend among serious users of Pascal on micros to move away from UCSD Pascal to more standard, reliable, and faster implementations.

An example is Andrew Tanenbaum's Pascal-E (see Implementation Notes), a highly portable Pascal implementation initially developed on PDP-11's. It produces an optimal Pascal intermediate code called EM-1; the EM-1 optimizer on the 11 produces a full compiler in 20K bytes! Other examples are Boston Systems Office Pascal and 2 "native code" compilers for the Z-80 (from Indiana University and Zilog). According to Michael Rooney at BSO, their Pascal is a set of optimizing cross-compilers for use in burning ROM's. George Cohn at Indiana University has a compiler which can now compile itself (see Implementation Notes #13); Zilog seems to have a compiler as well (see Implementation Notes, this issue). Also be sure to watch Motorola's Pascal on the 68000 and National Semiconductor's Pascal on their 2903 and 2910.

-Andy-

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

Tidbits

Peter C. Akwai, SchifferstraBe 88 6000 Frankfurt/M. 70, GERMANY: "Yes, we now have a Northwest Microcomputer Systems 85/P. This is an 8085-based micro with 56k bytes of user-accessible memory, builtin screen and keyboard, and 2 8-inch floppy drives. It is distributed with UCSD Pascal I.4 (a bone of contention and disappointment to us since from the Bowles book Microcomputer Problem Solving Using Pascal we were led to expect the II.3 release with graphics)." (*79/1/11*)

Gerald P. Allredge, Dept. of Physics, Univ. of Missouri-Rolla, 103 Physics, Rolla, MO 65401: "Wilhelm Burger recommended that I contact you concerning Pascal implementations for IBM Systems 370 facilities. (I am particularly interested in getting his Pascal-based parser generator BOBSW running on the University of Missouri Computer Network, which is based on a S/370 168-158 couple.) We presently have the University of Manitoba Version 1 compiler, but Wilhelm thought that the Tobias and Cox version of Pascal 8000 would likely be substantially better. Can you give me an opinion on this? (If you are aware of any better S/370 version, I'd like to know about it also." (*78/7/14*)

James A. Anderson, Dept. of Psychology, Brown University, Providence, RI 02912: "I am trying to find a Pascal program which can find the eigenvectors and eigenvalues of a real, symmetric matrix. An implementation of the Jacobi method is fine, or any alternate way of doing it. This is a very standard type of numerical task, so I suspect somebody must have done it. I would also be interested in finding out about programs for more general eigenvector and eigenvalue calculations if there are any around. I am doing some computer simulations of neural networks." (*79/8/1*)

Floyd O. Arntz, 44 Grove Hill Ave., Newtonville, MA 02160 "I am particularly interested in Pascal implementations available on soon-to-be be available on commercial time sharing services. Also I am considering PDP-11 or CY18(CDC) mini applications." (*78/12/1*)

Arnold Bob, Digitron, 500 Fifth Ave., New York, NY 10036 : "We were wondering if anybody has UCSD Pascal based software for sale. We're especially interested in business and graphics programs, however we're also interested in other applications programs." (*79/1/26*)

Edward W. Bolton, 4253 Moore St., L. A., CA 90066: "My interest is in implementing a subset of Pascal on an 8080 based system (SOL) in less than 44K(bytes)." (*78/10/11*)

Father Mick Burns, St Katherine's Episcopal Church, Martin, SD 57551: "I operate a 24K Heath H8 system and am hot on the trail of a grant to upgrade to a 56K RAM and Heath DOS. As you probably know Heath will shortly make Pascal available to H8 and H11 users. ...Particular interest is in CAI (Christian education)." (78/9/11*)

Richard Brandt, University of Utah, Dept. of Physics, 201 N. Physics Building, Salt Lake City, UT 84112: "I have been running UCSD Pascal on my Terak's since last December. Although it is not a "pure" Pascal, computer science students who have used it have preferred it to the other two Pascal's on campus, specifically the ones on the Burroughs 1700 and DECsystem 20... Our primary emphasis has been in the development of CAI material using both graphics and animation. We have developed the following: (1) a graphics editor; (2) a screen editor; (3) a CAI compiler; (4) a CAI interpreter; and (5) an algebraic answer analyzer." (*78/11/15*)

Robert Cole, GTE Automatic Electric Labs, 11226 N 23rd Ave., Phoenix, AZ 85029, (602) 995-6900: Sent a letter on 78/10/30 soliciting help in finding a commercially produced PDP-11 to Intermediate code to Intel 8086 optimizing compiler written in Pascal.

Lorne Connel, University of Waterloo, Dept. of Computer Science, Waterloo, Ontario, Canada N2L 3G1: "We would like to obtain the SLAC Pascal compiler so that we may compare its performance and usablity to other Pascal compilers we have tried. Could you please direct us to someone in this regard." (*79/4/10*)

Here and There With Pascal

Paul F. Fitts, INNOVATEK MICROSYSTEMS INC., Smithfield Rd., Millerton, NY 12546: "We have an immediate application for preparing an extensive software package and wish to consider Pascal as the program language... We are interested in locating Pascal software, such as compilers and applications programs." (*78/10/12*)

Charles D. Foley, 4 Knollwood Lane, Cold Spring, NY 10516: "To get to the meat of the request, I would like availability information on compilers for [IBM System/3 Model 10]..." (*79/2/26*)

Till Geiger, Falkensteinweg 8, D-7910 Neu Ulm, Germany: "I am just a fan of Pascal. My knowledge of Pascal is rather limited. Last spring I started to do some Pascal programming for about 3 months at New Ulm (Minnesota) High School. The inspiration to use Pascal came from a Pascal News copy a friend lent me. Compared to BASIC, it seemed to offer a totally new field. Those three months I worked with Pascal I got little done, because there were no books or other aids around. But I started to like Pascal and would prefer it over BASIC. In May I left for Germany. And MECC [Minnesota Educational Computing Consortium] is unachieved here. The school I am going has a PDP-11 but only with BASIC. Other schools don't even have computers in their school. So I have to stick with BASIC. Maybe in the near future I will find some system with Pascal in the Ulm area." (*79/4/23*)

Tony Gerber, etc., Basser Dept. of Computer Science, Madsen H08, University of Sydney, N.S.W., 2006 Australia: "Our department has finally switched to teaching Pascal, thus joining every other major Australian university in this regard." (*79/7/18*)

George W. Gerrity, University of New South Wales, Dept. of Mathematics, Australia: "At the moment, we have several PDP-11 machines running RSX-11, RT-11 (and UNIX part-time) and are looking desperately for a Pascal and/or Concurrent Pascal compiler or interpreter which will run under RSX-11D." (*78/7/17*)

J. Daniel Gersten, General Electric Co., Syracuse, NY 13201: "I am running the Swedish Pascal on a PDP-11/60 RSX-11M system. I have succeeded in compiling the compiler on the PDP-11 for version 4 and am presently working on the same for version 5." (*78/11/17*)

Jim Gilbert, Systems Structuring Technology, 30436 N. Hampton Rd., Laguna Niguel, CA 92677: "Get some cooperative soul to donate original copies of issues 1-8 for reproduction at exorbitant rates for the faithful who must have them." (*78/9/30*)

Pete Goodeve, 3012 Deakin St. #D, Berkeley, CA 94705: "We are using the University of Lancaster (P4) Pascal as the basis of a real-time experiment control installation. As you can guess, this needed some extensions to the system! (mainly consisting of an assembly language interface via external procedures, from which we can hang any kludges we like)." (*78/11/27*)

Geoffry R. Grinton, Herman Research Laboratory, Howard St., Richmond, VA: "we are at present using OMSI Pascal-1 under RT-11 on a PDP-11/34 and several LSI-11 systems and AAEC Pascal 8000 on an IBM 370" (*79/4/24*)

James Hargreaves, POB 14734, Cincinnati, OH 45214: "I plan to use Pascal on 990/4 and 990/10 TI computers as well as 9900 and 770 line equipment manufactured by TI that is compatible with the 990/4 and 990/10 cpu's. ... If you know of anyone in the USA who has converted the DEC based Pascal and Concurrent Pascal software on the TI 990 or 980 or 960 cpu's, I would like to get in touch with them." (*78/12/4*)

J. Niel Haynie, North Ridge Data, 971 E. Commercial Blvd., Fort Lauderdale, FL 33334: "We at North Ridge Data have recently committed ourselves to a major software development effort in the Pascal language. Specifically, we will use a micro computer implementation of UCSD Pascal in a real-time, interactive application....One of our primary concerns is the standardization of Pascal. We hope that the problems with Basic and its 50-odd versions does not befall Pascal. This would truly limit the expansion of Pascal into its deserved position as the "Lingua Franca" of computing." (*79/3/16*)

Ed Johnston, 715 6th St., Rochester, MN 55901: "As an IBM employee, I am attempting to generate some interest in Pascal within the company. Few people seem to have heard of it." (*78/12/12*)

Here and There With Pascal

Robert S. Kirk, American Microsytems Inc., 3800 Homestead Rd., Santa Clara, CA 95051: "American Microsystems, Inc. currently has Pascal running on our 6800 MDC's. We have a compiler on order from the University of Tasmania for our large Burroughs B7700 computer, and we are looking for a Pascal compiler for the PRIME 400 computer. Hopefully, your Users Group can aid us in locating Pascal compilers and in making this relatively young language a standard programming tool at American Microsystems, Inc." (*79/1/11*)

Les Kitchen, Comp. Sci. Ctr., Univ. of Maryland, College Park, MD 20742: "Very pleased to see draft standard in #14 especially type-equivalence defining occurrence & for-loop semantics." (*79/3/15*)

David A. Kohler, 1452 Portobelo Dr., San Jose, CA 95118: "I love the PN idea, but find the format a little disconcerting and difficult to read. Keep up the fine effort and emphasize those algorithms and software tools" (*78/12/28*)

Pierre J. Lavelle, Rua Pompeu Loureiro, N 120 APT. 602, 22061-Copacobana, Rio De Janeiro-Brazil: "Traveling PUG members welcome!" (*78/11/17*)

Richard Linton, 3027 N. Shepard Ave., Milwaukee, WI 53211: "Here at the U. W. -Milwaukee we are using both the Navy's and U. W. -Madison Pascals and we are currently running evaluations between the two." (*79/3/3*)

Paul C. Lustgarten, Computer Sciences Dept., U of Wisconsin, 1210 W. Dayton St., Madison, WI 53706: "I am a third year grad. student and teaching assistant at Univ. of Wisc. - Madison, and have been eager to use Pascal to teach introductory programming since I first used it. Although most of our (non-numeric) courses use Pascal whenever possible, almost all of our introductory courses use FORTRAN, COBOL, or BASIC! The only exception to this is the version of the intro. course for potential Computer Science majors, which uses Pascal... Also--my wife is a programmer for a company that produces data base systems on Data General Novas. Apparently, they view the execution speed of their systems as being of primary importance (over such other things as software reliability, cost/time of development, maintenance, etc.), and don't believe that any high-level language could possibly compete in this regard with the several dialects of assembly language they currently use (their comparison is with DG FORTRAN). Does anyone have any statistics or convincing arguments?" (*79/1/9*)

David Matthews, Process Computer Systems, 750 N. Maple Rd., Saline, MI 48176: "Printing actual programs (PUG News #12) was a great help in learning better (easier to read) style." (*78/8/21*)

Jim McCord, 330 Verada Leyenda, Goleta, CA 93017: "I'm a hobbyist using UCSD Pascal. Main interests are graphics, teaching-type programs and sophisticated games (a la Adventure). How many other hobby-Pascal'ers are there?" (*78/11/14*)

Monte Jay Meldman, M. D., 555 Wilson Lane, Des Plaines, IL 60016: "I am interested in knowing about word processers and accounts receivable and things like that on Pascal and would appreciate any information you can give me about applications that have been written for the PDP-11/40, RSTS/E. It really sounds like Pascal is interesting." (*78/11/15*)

Paul Miller, Avera Technology, 1643 Wright Ave., Sunnyvale, CA 94087: "My company has recently determined to use Pascal as the primary implementation language for a new product development. Our current plan is to do program development on a PDP-11 system under RSX-11M and then cross-compile for the microprocessor in our product. Any information you could send me about... DEC Pascal, or available help in starting up a Pascal product would also be appreciated." (*79/5/7*)

Anne Montgomery, POB 30204, Lowry AFB, CO 80230: "McDonnell Douglas has developed a CMI/CAI system here on Lowry Air Force Base called the Advanced Instructional System(AIS). ...This system is basically an extension of the CDC Scope 3.4.3(level 439) operating system. For the development of AIS we have developed a Pascal-like language called CAMIL. The machine coded generater for the CAMIL language is written in Pascal. Camil, while intended primarily for CAI/CMI applications, also happens to be a very good general purpose language but can be run only in the interactive time sharing environment. Until a batch version of CAMIL can be developed, we are also using Pascal as our batch language. It has been used primarily to create batch versions of CAMIL programs because of the similarities between Pascal and CAMIL." (*78/10/12*)

Greg Morris, 297 Turnpike Rd., Westboro, MA 01581: "Much to my surprise, I was able to quickly find a job working with Pascal." (*79/3/28*)

Maurice R. Munsie, Network Computer Services, 69 Clarence St., Sydney, Australia, 2000: "We are distibuting in Australia OMSI Pascal-1. A number of sales have been already made and plans are being made for the OMSI implementors to hold workshops in Australia later this year." (*78/7/27*)

David Nedland-Slater, 1, Buckland Close, Farnborough, Hants. GU14 8DH, United Kingdom: "I am interested in Pascal for micro work as a real alternative to assembler. I hope Pascal keeps us away from nasty bit twiddling." (*78/10/3)

Niel Overton, Computer Systems & Services Inc., Box 31407, Dallas, TX 75231: "Wanted- an accounting package in Pascal. Wish to convert to target machine: TI DS990-2." (*79/9/5*)

G. Dick Rakhorst, Manudax Nederland B. V., 5473 ZG Heeswijk(NB), Holland, PB 25, Meerstraat 7: "As a distributor of Motorola Semiconductors Division in Holland we will introduce within one month a Dutch-written Pascal compiler for the Motorola MC 6800 microprocessor and also will Motorola introduce a Pascal compiler soon for the new MC 6809 and the 16 Bits MC 68000." (*78/11/27*)

F. Eric Roberts, Perkin Elmer Co., Mail Station 284, Main Ave., Norwalk, CT 06856: "I'm introducing the virtues of Pascal to a Fortran, PL/I and assembler community, for applications and small systems work. Full marks for fantastic Pascal News." (*78/10/5*)

Robert E. Rogers, Jr., 18625 Azalea Dr., Derwood, MD 20855: "I have received a copy of the University of Bratislava Pascal-b compiler for CDC 3500 Machines. We have been using it for only a short time and are attempting to compile a list of differences between this implementation and the UCSD Pascal. Hopefully by early spring we'll have something ready." (*79/1/1*)

Antti Salava, Munkkiniemen Puistotie 17A 13, SF-00330 Helsinki 33, Finland: "...University of Helsinkl, where I was implementing Pascal-HB compiler on Burroughs B6700. It's been running now a couple of years without any fatal crashes." (*78/8/28*)

John M. Smart, Smart Communications, Inc., 866 United Nations Plaza, New York, NY 10017: "WANTED - conversion program or part time programmer, capable of converting programs in Burroughs extended ALGOL for B6700 into Pascal for PDP-11 or other systems, including B6700." (*79/8/1*)

Edward R. Teja, EDN, Cahners Publishing Company Inc., 221 Columbus Ave., Boston, MA 02116: "EDN is preparing to write an article dealing with the current interest in Pascal. Our intention is to look at both the historical and contemporary aspects of the situation; we want to put the situation into its proper perspective." (*78/12/15*)

M. Thornbury, Totalisator Agency Board, P. O. Box 3645, Wellington, New Zealand: "The N.Z. TAB are presently designing a large-scale wagering system utilising INTERDATA computers. We originally decided to use the RATFOR preprocessor as a front end to the FORTRAN compiler, but feel that FORTRAN VII does not have a sufficient instruction set to perform certain functions efficiently. We would therefore like to write our software in Pascal if we can locate a compiler presently running on an INTERDATA 8/32." (*79/3/13*)

Bob Wallace, Microsoft, 10800 NE 8th , #819, Bellevue, WA 98004: "Microsoft is developing a microcomputer Pascal compiler." (*79/1/18*)

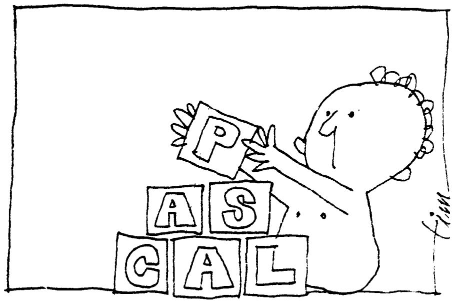
Marie Walter, Scientific-Technical Book and Copy Center, 17801 Main St., Suite-H, Irvine, CA 92714: "...I am also enclosing our current bibliography on Pascal which has proved very popular. CIT has been distributing it with their literature on the Microengine and I get calls from all over the country from people just getting into Pascal. Item 3: I thought you might be interested in our Pascal tee shirts which we just started turning out. They come small, medium, large and can be on any background. $4.95 per." (*79/3/23*)

SCIENTIFIC-TECHNICAL
BOOK CENTER
17801 MAIN STREET
IRVINE, CALIFORNIA 92714
(714) 557-8324

(* letters on blocks can be clear, red, yellow, or blue *)
prices subject to change
by publishers

in California add 6% sales tax

mail orders add $1.50 postage



Copyright 1979 by Scientific-Technical Book Center

Allen A. Watson, The Record, 150 River St., Hackensack, NJ 07602: "The Record (a newspaper) is not currently using Pascal on our 370/138s, but we are considering doing so in view of a possible move in the near future to other mainframes. So what we are looking for is general information about Pascal, advantages vs. other languages--that kind of thing." (*79/3/2*)

Robert Williams, MicroMouse Enterprises, Box 69, Hollywood, CA 90028: "I am building two minicomputers; the first of which was up-n-running earlier this year: a DEC LSI-11 with 20 kwords RAM and two floppy drives. The second is equally powerful (or maybe more so); it is the Alpha Microsystems AM-100. Pascal is to be the main software link between them. I have not yet obtained any code, altho I have the AlphPascal Programming System users reference manual which is a bargain at $7.50. I believe the source was from UC San Diego." (*78/10/6*)

D. J. Yates, Botany Dept., University of Queensland, St. Lucia, Qld, Australia 4067: "I am running two North Star Horizons. Don't yet have Pascal-but it is on order. Very pleased with the Horizons." (*79/3/14*)

Earl M. Yarner, 195 Varick Rd., Newton, MA 02168: "...Hewlett-Packard presently supports FORTRAN and assembler but I hear rumours that they are working on adding Pascal. I am afraid that they will take a long time to get ready, so I would like to put Pascal 'on-line' myself, hopefully within the next year. Any advice or assistance that you or any other member of the group can give me would be appreciated." (*79/3/19*)

Pascal in the News

ACADS Newsletter (The Association for Computer Aided Design Limited, in Australia), No. 19, December 1978: "PASCAL-Everybody's Language?" A short note on the growing popularity of Pascal, the availabilty of compilers, and how to get the Australian Atomic Energy Commission IBM OS/ compatible compiler.

AEDS MONITOR, Apr/May/June 1979: "Basic Thoughts on BASIC", on the use of BASIC as a teaching language. The author sees BASIC as a bad choice, sees hope with possibly Pascal, and would like to see the fundamentally important things involved in teaching programming be brought out.

Australian, July 24, 1979: "Pascal Program" announcing the release of the Pascal Validation Suite by Professor Arthur Sale at the University of Tasmania.

Business Week (industrial edition), April 23, 1979, pg 46: "Computers Rush to Talk to Pascal" covers the growing use of Pascal by major manufacturers. "Pascal is now the odds-on favorite to become the dominant language for microprocessors" says the article along with many other reasons for making the switch to Pascal.

Byte, September 1978, pg.71: An ad for Northwest Microcomputer Systems NMS 85 Series which uses a likeness of Blaise Pascal as its drawing point. Needless to say, Pascal is offered with the machine.

Byte, October 1978, pg.129: An ad for a new book entitled "A Concurrent Pascal Compiler For Microcomputers", by Alfred C. Hartmann.

Byte, November 1978, pg.142: A letter entitled "READER Cs PASCAL ALTERNATIVE", Which is one reader's comparison of C and Pascal.

Byte, December 1978, pg.178: An ad for Cyber-Score Inc, Pontiac, Michigan, offering Pascal softwore, mainly business-oriented.

Byte, February 1979, pg.185: A HELP WANTED ad for Fischer and Porter, Warminster, PA, for software engineers with among other qualifications, a knowledge of Pascal.

Byte, March 1979: A letter critiquing the article "Creating a Chess Player" in the October 1978 issue, which was part of a series of articles on a chess program written in Pascal.
Also an ad for a Pascal Engine, from Cutting Edge of Technology, pg.78.
pg.107: A short note: "More companies jumping on the Pascal bandwagon".
pg.59: an ad for another implementation of Pascal, on Control Systems, Inc. UDS 470. It says that Pascal has been used on their machines to control grain elevator operations.
pg.237: An ad for Oregon Software's OMSI Pascal, and how to get it.

Byte, April, 1979, pg.239: "Pascal versus Basic...", an article comparing Pascal to BASIC.

Byte, May, 1979, pg.20: An ad for Western Digital's 16-bit Pascal Microengine.
pg.57: An ad announcing Pascal for the North Star Horizon.
pg.118: A note that Microsoft plans to announce a Pascal Package plus a note about the U.S. Joint Pascal Standards Committee.
pg.224: A letter which opposes the bundled packaging of Pascal on microcomputers, with UCSD Pascal as its target.

Byte, June 1979, pg.130: 2 short notes, one about Pascal for the 6800 and another about the DOD's Pascal-like language, ADA.
pg.194: An article which mentions an APL interpreter written in Pascal.
pg.202: An ad for 'Tiny Pascal' for TRS-80 and North Star from: Supersoft, POB 1628, Champaign, IL 61820.

Byte, July 1979: In the section NYBBLES, an article about the "TINY Pascal Compiler", which has now been rewritten in 8080 assembly language. The compiler is based on the one published in earlier issues of Byte.
pg.146: An ad for Technology System South's (Loris, SC) Pascal Microengine.
pg.169: An ad for TRS-80 Pascal (a version of UCSD Pascal), available from the FMG Corporation, POB 16020, Fort Worth, TX 76133.
pg.239: An ad for a Pascal compiler for the Zilog Z80. The claim is that it "is often twenty times as fast as UCSD's implementation". Available from: Ithaca Audio, POB 91, Ithaca, NY 14850.
pg.240: An announcement for M6800 Pascal from Central Systems (Williamsburg, VA).

Central Scientific Computing Facility Computer Newsletter(Brookhaven), Volume 18,no. 7, pg.110: A note mentioning a 7600 version of Pascal installed on MFZ, which is essentialy the same as Pascal version 1 on the 6600.

Computer Design, October 1978, pg.188: "CPU Interfaces Processor to S-100 Bus, Providing 16-Bit Minicomputer Power and Pascal", an announcement that there is available to the user of Marinchip Systems M9900 CPU board, which utilizes Texas Instruments TMS9900 processor, both concurrent and sequential Pascal. Both compilers are converted from those developed by Per Brinch Hansen. Marinchip Systems is located at: 16 Saint Jude Rd., Mill Valley, CA 94941.

Computer Design, March, 1979, pg.179: "Pascal Adaptation to Development Center Will Speed Programming", American Microsystems will support Pascal on its MDC-100 product line.

Computer Weekly, November 9, 1978, pg.7: "Now National Opts for Pascal, the People's Language", an article about National Semiconductors decision to support Pascal and what National considers to be the advantages of Pascal.

Computer Weekly, May 24, 1979: "Data General Offers Pascal" Data General's Micron, an operating system for their 16-bit MicroNova, which comes with a Pascal compiler.

Computer Weekly, May 31, 1979: "DEC Pascal for VAX" about a soon-to-be-released native mode Pascal compiler for the VAX-11/780 by DEC and the University of Washington, plus the fact that the University of Adelaide, Australia, ordered 3 VAX machines partly because of the availability of the compiler.

Computer Weekly, (Pacific) August 10-16, 1979: Letter by Arthur Sale in response to a quote from Cobol pioneer Grace Hopper, 'Cobol has knocked PL1 dead and it will do the same to Pascal'. Professor Sale asserts ' that Pascal is not a "fad"'.

Computerworld: (Many issues) ads for Oregon Software (OMSI) PDP-11 Pascal.

Computerworld, February 12, 1979: An ad for Sperry-Univac, Minicomputer Systems, introducing SUMMIT. Pascal is the headlined language that goes with the system although there are other languages available.

Computerworld, February 26, 1979: "Seminar to Consider Pascal Programming" announcing a seminar "Pascal Programming for Mini- and Microcomputers" to be held April 23-27, 1979.

Computerworld, March 12, 1979, pg.99: A want-ad for programmers at Sperry-Univac which mentions of Pascal as parts of the qualifications.

Computerworld, March 19, 1979: "Pascal Now on Level 6 Mini" about the availability of an extended Pascal compiler for the Honeywell, Inc. Level 6 minicomputers. The Pascal has shown programming time reduced by a factor of three on small to medium sized programs and up to 10 times for large programs compared to FORTRAN, COBOL, or assembly language.

Computerworld, March 26, 1979: "Academic-Industrial Union Ends in VAX Pascal" about the University of Washington and DEC's cooperative effort to produce a Pascal compiler for the VAX-11/780.
pg.51: "Pascal Ready for Eclipses under AOS", about the availability of a Pascal compiler from Gamma Technology Inc. , for use on large scale Data General Corp. Eclipse minicomputers running under AOS. Also, on the same page "Package Backs PDP=11 Transaction Processing", about Cytrol's (Edina, MN) CSS-11 package for PDP-11's providing transaction, database and communication processing allowing applications programs written in Pascal.

Computerworld, May 14, 1979: "DOD Stops Work on 'Red' Gives Go Ahead to 'Green'", about the progress of the DOD's study of the 'Red' and 'Green' languages. Green was chosen and is to be called ADA, after Lady Ada Lovelace, who assisted Charles Babbage.

Computerworld, May 28, 1979: "Languages, Operating System Available for DG Micronovas", about Data General Pascal for the MicroNovas, plus a want ad for programmers at Control Data in St. Paul, MN who must know Pascal among other qualifications.

Computerworld, July 16, 1979, pg.41: "Lawsuit Could Set Dangerous Precedent", an editorial which mentions the use of Pascal over FORTRAN.

Computerworld, July 23, 1979: "Apple Offers Users Plug-In Pascal Option", about the "Language System" on Apple computers, a plug in option for the Apple-II that allows users to develop software in Pascal. The package is available at your Apple dealer.

Computerworld, August 6, 1979: "Pascal Now Available for Zilog Z80 Systems", announcing Pascal for Zilog Z80 sytems, available from Zilog at 10340 Bubb Road, Cupertino CA 95014.

Computerworld, August 13, 1979: "Pascal/8002 Development Package Debuts", an announcement of the Pascal/8002 Universal Program Development Package, a software product designed for use with the Tektronix, Inc. 8002 Microprocessor Development Laboratory, by the Pascal Development Co., Suite 205, 10381 S. DeAnza Blvd., Cupertino, CA, 95014.

Computerworld, August 20, 1979: "Pascal Runs on DG Units", announcing the first in a series of five implementations of Pascal for use on Data General Minicomputers, developed by Rational Data Systems, 245 W. 55th St, NY, NY 10019.

Computerworld (Australian), August 3, 1979: Announcement of the availability of the Validation Suite for Pascal, developed in Australia and England. "Validation Suite for Pascal".

Computing News (Computing Services, Northern Illinois University), December 1978: An announcement of the installation of the University of Manitoba Pascal compiler for the IBM 360/370.

Computing Europe, April 5, 1979, pg.1: "Pascal Draft Breaks US Language Grip", describes the British Standards Institutions leadership under Tony Addyman for an International Standard Pascal.

Computing Europe, March 29, 1979: "Pascal is Top of the Class", concerning the use of Pascal for trainee programmers. The results of a study have shown Pascal to be a justified choice for a language to learn programming.

Computing Europe, April 19, 1979: "Floreat Pascal" a letter from C. A. G. Webster referencing the previous article 'Pascal is top of the class', and after 6 years and 500 students agrees wholeheartedly.

Computing Europe, May 3, 1979: An article on the rapid acceptance of Pascal in Australia.

Computing Europe, May 24, 1979: "DG Offers 'Fast Pascal' on two Major Systems",announcement about an across the range compiler for Micronovas to Eclipses, which is according to a spokesman '...not much of a gamble. If you look at high level programming languages available on mini-based machines, there is not much choice'.

Computing Europe, August 6, 1979: "Australia Loves Pascal", a short note about the rise in the use of Pascal in Australia.

Data Communications, March 1979, pg.16: "High-level language attracting new commercial users"An article concerned with using Pascal for data communications, with Sperry Univac's Summit operating system used as an example.

Datamation, July 1979: "Pascal Power", a collection of 4 articles on Pascal, dealing with Pascal's future, its use by the DOD, Pascal's structure, and its uses with micros and minis.

Datamation, August 1979, pp.166-172: Announcements for Apple II Pascal option, Zilog's new Z80 Pascal compiler, and Digicomp Research's new Pascal 100 system.

Diebold Research Program Document Number T23-V1113: Titled "Trends in Systems Software: 1985, 1990, 1995", on page 30 has a short shot at Pascal. The document is marked "Confidential-For Client Use Only", so I did not take the liberty of copying it. (John K. McCandliss)

Dr. Dobb's Journal of Computer Calisthenics and Orthodontia, February 1979, no.32, pg.29: A fairly complete Pascal bibliography by Mike Gabrielson.

Electronic Engineering Times, May 28, 1979, pg.10: An article about Pascal being used on 3 major minicomputers by DEC, Data General, and Texas Instruments.

Electronic Engineering Times, June 25, 1979, pg.30: "Pascal Touted by Engineers As Help For High Software-Development Costs, But Not Seen As Panacea", which discusses the advantages of Pascal to engineers, and also discusses the flaws of Pascal implementations at this point.

Electronic Engineering Times, Aug 20, 1979: "Plethora of PASCAL Possibilities Provided for Data General Users", gives information on how to obtain Pascal for Data General's advanced operating system, developed by Rational Data Systems.

Electronics, December 21, 1978, pg.6: "Obeisance to Pascal Inventor", a letter from Niklaus Wirth, explaining his choice of the name Pascal for the language.

Electronics, June 7, 1979: The cover article "Putting Pascal to Work", is about the adaptation of Pascal to Texas Instruments machines. Part 2 of this article covers the microprocessor version of TI Pascal.

Electronics, August 16, 1979, pg.33: A notice that Softech has acquired control of UCSD Pascal.

Florida State University Computer Center Newsletter: A note that release 2.3 of the E.T.H. Pascal compiler is going up on June 11, 1979.

ICCC (Imperial College, London Computer Center Newsletter), March 1979: "Programming Notes-Pascal", a short note about the increased use of Pascal at ULCC, followed by a few references to Pascal.

Intelligent Machines Journal, February 28, 1979: "New Micro Offers Pascal in ROM for OEM's", another announcement for CSI Microsystem's (Kansas City, KS) UDS 470 computer with Pascal.

Intelligent Machines Journal, April 18, 1979, pg.8: "Pascal Advancement Society of California", an announcment of a group for the exchange of information about Pascal. It should be noted that this group is not PUG California style, but rather a local group that hopes to have its members cooperate to obtain Pascal systems and programs. For information contact Mark Gang, 2262 Fairvalley Ct., San Jose, CA 95125.

Interface Age, June 1979: The first in a series of articles entitled "The Pascal Notebook", the others following in July and August. The article is a tutorial on Pascal and may be of interest to those just learning programming, in particular Pascal, and especially to students who are for the first time learning to program in Pascal.

MACC NEWS #3(University of Wisconsin, Madison Academic Computer center) January 1979: An announcement of a new UW-Pascal release for the Univac 1108.

MICC Digit, (Middle Illinois Computer Cooperative Newsletter) January 1979, pg.3: An answer to the question "How do I format output from a PASCAL program?"

Minicomputer News, November 9, 1978, pg.24: "LSI Chip Set Directly Executes 16-Bit Pascal Application Code", another announcement about Western Digital's Pascal Microengine.

Minicomputer News, February 1, 1979, pg.20, pg.30: "Sperry Opens V77 Minis to Pascal", and "Micro Offers Pascal in Prom", another CSI minicomputer announcement.

Mini-Micro Systems, November 1978, pg.10: "Jumping on the Pascal Bandwagon", an article what many companies are doing with Pascal, in this case all manufacturers of micros.

Mini-Micro Systems, March 1979: "Pentagon to Debut ADA; Commercial Vendors Wary", about commercial vendor reaction to ADA.

Mini-Micro Systems, May 1979, pg.10: A letter entitled "Disenchanted with Pascal", in reaction to the above mentioned article "Jumping on the Pascal Bandwagon", which claims that Computer Automation has a better language (ALAMO) than Pascal, and that Pascal is obsolete.

The OEM Computer Newspaper, November 7, 1978: "Pascal Takes Off", a short article about the success of Pascal.

Sandia Computing Newsletter, No.05/1979, May 1, 1979: "Pascal on NOS", an announcement that Pascal-6000 is available on NOS for for the CDC 6600.

Scientific American, August 1979: Two ads, one for Oregon Software (OMSI) and their use of Pascal, the other an ad for the Apple Computer, which mentions that Pascal is available to users of the Apple.

Silicon Gulch Gazette, March 28, 1979, pg.25: "Pascal: An Aggrressive Young Language the Way Up", announcements for Pascal presentations at the Fourth Annual West Coast Computer Faire in San Francisco, May, 1979: Tom Pittman, a user of Western Digital's Pascal Microengine, Jack Sharp for Varian Research, and Marie Walter on the Midwifing of a Pascal Standard.

Small Systems World, August, 1979, pg.32: An announcement for Pascal accounting software by P.S. Inc, Fargo, ND.

UMD Computer Center Newsletter (U of Minnesota, Duluth), February, 1979, pg.5: An announcement that Pascal-6000 Release 3 has been installed on their Cyber 171.

WSU CCN(Washington State University Computer Center Newsletter), April 3, 1979, pg.4: "Pascal Under the Batch Monitor", a notice that Pascal 8000 is now available on the Amdahl 470.

Pascal and Teaching

We've received good response to this new section; unfortunately, in spite of 3 good contributions for this issue, we decided to postpone them to issue #17 so that we can save space here. Sorry.

Ada (ALIAS DoD-1) (ALIAS Green)

Many Pascal Users are asking about Ada. How good is it? Is it just like Pascal only better? When will we see it? Well, back in the heart of Pascal country we have analysed Ada, and we regret to say that its resemblance to Pascal is so slight that we may not devote any more space in Pascal News to it after this. Ada is a very large and complex language, which should be illustrated by the following statistics. There does not exist as yet any compiler for it, and what such an implementation would look like is not certain. It has the declaration-before-use feature of Pascal which was intended to allow one-pass compilation, but rumour has it that seven passes through the symbol-table may be necessary to resolve potential ambiguities of the overloading. The resolution of overloading ambiguity is too complex to document, so probably programmers will have to leave that to the compiler to resolve. Who wants to go back to languages that can't be understood?

To quote Charles Bass, general manager of Zilog's Microcomputer Systems Division: "Ada will become a millstone around our necks" (Mini-Micro Systems, March 1979).

Edsger Dijkstra prophetically said that he hoped that Pascal was not better than all its successors. He may have been right to worry.

Size of Defining Document
190 pages
(Pascal J&W = 35 pages, ISO draft standard = 43 pages)

Number of Reserved Words
62
(Pascal = 35)

"Features" of Ada
Generic procedures, overloading of identifiers and operators, confusing abstraction and representation for real types,
much syntactic sugar,
too many ways to do the same thing. No sets! No files or sequences in the Pascal sense.
Yet another bizarre set of operator precedence rules. Optional omission of actual parameters (coupled with two sets of parameter association syntax and default values). Ability to freely specify representation of abstract notions without separation of concerns.

Purpose of Ada
Acceptance by DoD as a uniform programming language for real-time and other applications. So far only the US Army have shown interest, even though the very complexity of Ada should appeal to the military mind.

Perhaps the biggest shame is that a beautiful name like Ada, and a woman like Lady Lovelace, should be associated with such an insensitive creation.

Letter to the Editor,
Australian Computer Bulletin. 27th August, 1979

Programming Language Ada

Keen watchers of the U.S. Department of Defence will have been observing the progress of the High Order Language Commonality program. Starting in 1975 and progressing through a series of specifications known as Ironman, Steelman, etc, the U.S. DoD has now arrived at a draft of a new programming language called *Ada* after Ada Augusta, Lady Lovelace, the first programmer.

A copy of the specification, for those interested, is available from

Association for Computing Machinery, Inc.,
P.O. Box 12015,
Church Street Station,
New York, NY 10249 (US $ 22.00)

as Volume 14, Number 6, June 1979, Parts A & B of SIGPLAN Notices.

Ada is stated as being heavily influenced by Pascal. I must say, however, that I found this heavy influence rather hard to detect on reading the documents: to me it seems to clearly and definitely belong to the Algol 68, PL/I or C class of languages in size, features, and basic principles. Apart from a few concepts, the resemblance to Pascal is more like a parody.

The Department of Defence have, of course, solicited comments on the draft. Since it would be very improbable that they would change it substantially, it seems likely that a slightly modified Ada will become a Defence standard in 1980. This means that it will be important in the U.S.: I now have considerable doubts that its influence will be as widespread elsewhere (or in industry) as some people have predicted. However I may be wrong - there is no limit to the extent to which we ignore flaws, and Fortran 77 stands as mute witness to that fact.

Arthur Sale,
Professor of Information Science.

Books and Articles

{Unfortunately I did not collect, forward, or organize materials in time for Rich Stevens to have the slightest chance to produce his regular section. Look for a burgeoning section in #17.}

Publishing success story

The Pascal User Manual and Report by Jensen & Wirth has now sold more than 60,000 copies. We understand that this includes a bulk purchase of 10,000 copies by Apple Computer Inc, and a similarly large quantity by National Semiconductor.

Also in the big selling stakes is Programming in Pascal by Grogono, which has sold over 35,000 copies, with a single order of 10,000 copies going to Motorola.

Book Reviews

We understand that Jan Hext, Basser Department of Computer Science, University of Sydney, New South Wales 2006, Australia, has written a comprehensive review of all the Pascal textbooks now available which is to appear in a special issue of an Australian journal called Microsystems. We hope to get permission to reprint Jan's article in Pascal News, but in the meantime we can only extract the citation and one column of a table of comparisons.

Introductory books:
Bowles, K.L., *Microcomputer Problem Solving using Pascal*, Springer-Verlag, New York, 1977, 563 pages, $A 11.45
Conway, R.W., Gries, D. and Zimmerman, E.C., A *Primer on Pascal*, Winthrop Publishers Inc., Cambridge, Mass., 1976, 433 pages, $A 14.75
Grogono, P., *Programming in Pascal*, Addison-Wesley Publishing Inc., 1978, 359 pages, $A9.95
Jensen, K. and Wirth, N., *Pascal User Manual and Report*, Springer-Verlag, Berlin, 1974, 170 pages, $A 8.70
Kieburtz, R.B., *Structured Programming and Problem-Solving with Pascal*, Prentice-Hall Inc., Englewood Cliffs, 1978, 365 pages, $A 14.75
Rohl, J.S. and Barrett, H.J., *Programming via Pascal*, Cambridge University Press, in press, about 250 pages.
Schneider, G.M., Weingart, S.W. and Perlman, D.M., *An Introduction to Programming and Problem-Solving with Pascal*, Wiley & Sons Inc., New York, 394 pages, $A 21.25 (hard-cover), $A 13.15 (soft cover).
Webster, C.A.G., *Introduction to Pascal*, Heyden, 1976, 129 pages, $A 13.75
Welsh, J. and Elder, J., *Introduction to Pascal*, Prentice-Hall Inc., Englewood Cliffs, in press, about 220 pages, $A 13.95
Wilson, I.P. and Addyman, A.M., A *Practical Introduction to Pascal*, MacMillan Press Ltd., London, 1978, 148 pages, $A 9.95

Advanced books:
Alagic, S. and Arbib, M.A., *The Design of Well-Structured and Correct Programs*, Springer-Verlag, New York, 1978, 292 pages, $A 13.60
Coleman, D., A *Structured Programming Approach to Data*, MacMillan Press Ltd, London, 1978, 222 pages, $A 13.75
Wirth, N., *Systematic Programming: An Introduction*, Prentice-Hall Inc., Englewood Cliffs, 1973, 169 pages, $A 23.75
Wirth, N., *Algorithms + Data Structures = Programs*, Prentice-Hall Inc., Englewood Cliffs, 1976, 366 pages, $A 26.95

Coverage of books, taken from review

First author	Coverage of Pascal
Bowles	fair
Conway	poor
Findlay	good
Grogono	very good
Jensen	good
Kieburtz	poor
Rohl	good
Schneider	fair
Welsh	very good
Wilson	good
Alagic	fair
Coleman	poor
Wirth(1973)	fair
Wirth(1976)	good

Conferences and Seminars

I apologize for the negative impact that tardiness has on this section. John Knight, for example has now been stale-dated twice regarding his PUG-ACM SIGPLAN conference session announcements. Below we have reports from the PUG/SIGPLAN meeting at ACM '78, the DECUS New Orleans meeting, the Australian Computer Science Conference. Next time I'll have the summaries from the French AFCET sub-group meetings on Pascal (belatedly - sorry). First, though we have news of seminars presented to teach Pascal primarily to professionals in the industry, followed by a list of upcoming conferences.

Seminars

The Polytechnic Institute of New York's Institute for Advanced Professional Studies is presenting seminar/workshops on Pascal Programming for mini and microcomputers in Boston on October 22-26, 1979 and in Palo Alto on December 3-7, 1979 for $600. For more information contact George Poonen at (617) 493-3537 or to register write to: Institute for Advanced Professional Studies, One Gateway Center, Newton, MA 02158. Phone: (617) 964-1412 (Donald French)

Vince Giardina by now must have information about a series of IEEE workshops on Pascal. He works out of the IEEE central office in New York City but the phone number I have is (201) 981-0060 x174 or 175 (which is in New Jersey). He was also looking for instructors for this course.

Integrated Computer Systems, Inc. has a "learning tree" (TM) 4-day course on "Pascal: Programming in the Structured Language". The course dates are: October 9-12 in San Diego, October 16-19 in Washington, DC, November 6-9 in New York City, November 13-16 in Boston, and December 4-7 in Los Angeles. A related set of courses are being taught on "Structured Programming - Scientific and Engineering Applications" The Pascal course is $795. To enroll write to: Integrated Computer Systems, Inc., 3304 Pico Blvd. P.O. Box 5339, Santa Monica, CA 90405. Phone: (213) 450-2060 or to 300 N. Washington St. Suite 103, Alexandria, VA 22314. Phone: (703) 548-1333. Ken Bowles is the course instructor.

Software Consulting Services is also offering seminars by Richard and Martha Cichelli:

Software Consulting Services
901 Whittier Drive
Allentown, Pa. 18103
[215] 797-9690

July 12, 1979

Dear Andy:

We have planned the following seminars which may be of interest to your readers.

October 17-19, 1979

A seminar/workshop entitled "An Introduction to Pascal Programming". Taught by Richard J. Cichelli and Martha J. Cichelli. Includes hands-on Pascal programming workshop sessions as well as group and individual instruction. The class will emphasize learning the basics of good programming in Pascal and learning them right! Class size is limited. Three days. For more information contact Software Consulting Services, 901 Whittier Drive, Allentown, PA 18103 (215) 797-9690.

November 14-16, 1979

A seminar/workshop entitled "Advanced Programming Techniques Using Pascal". Taught by Richard J. Cichelli and Martha J. Cichelli. Requires a basic knowledge of the Pascal language. This class will refine the skills of Pascal programmers and teach them how to build a comprehensive and effective Pascal-based software development environment. The emphasis will be on significant programming exercises blended with group and individual instruction. Class size is limited. Three days. For more information contact Software Consulting Services, 901 Whittier Drive, Allentown, PA 18103, (215) 797-9690.

Sincerely,

Martha J. Cichelli

Australian Seminars

Arthur Sale told us of two seminars in Australia that he had been involved with. One was a five-day intensive seminar held by his Department at the University of Tasmania, and the other was a two-day professional development seminar organized by the Australian Computer Society in Melbourne, Victoria. Pascal News acquired about 60 new members from these seminars, and even more people were exposed to Pascal's elegance.

Arthur also said that he had given part of an evening seminar with Michael Rooney of the Boston Systems Office which was attended by around 450 engineers involved in microprocessor applications in Australia. The interest in Pascal was sufficiently great that the University of Tasmania was planning another seminar addressed to professional programmers for February 1980.

Upcoming Conferences

IFIP in 1980 will be held one week in Tokyo and the next week in Melbourne Australia. We don't know of any attempts at a Pascal "interest group" session, but we're sure one will spontaneously occur.

The Fall DECUS meeting should be held in San Diego, and John Barr expects that issues such as compiler performance, Pascal standards, implementation techniques and Modula/Concurrent Pascal will be discussed.

Below is the announcement for ACM '79. If you have a talk, contact John Knight anyway even though you will be reading this late.

Dear Andy:

An informal evening session devoted to PASCAL will be held at the 1979 ACM conference which will take place October 29-31, 1979, in Detroit, Michigan. The session will be sponsored jointly by SIGPLAN and the PASCAL Users Group, and will be very similar to the session held at the 1978 ACM National Conference. 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, implementation, etc. Anybody who is planning to attend ACM '79 and who is interested in making a presentation should send a short description of what they will discuss by September 1 to:

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

Presenters will be informed of their selection by September 15.

The purpose of requesting descriptions is <u>not</u> 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
Programing Techniques Branch
Analysis and Computation Division

NASA
National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia
23665

Conference Reports

The Second Annual Australian Computer Science Conference was held in Hobart, February 1-2, at the University of Tasmania. Pascal was a recurrent theme in several papers.

- Jeff Tobias gave a talk "A Malleable Multiprocessor" about extending Modula for driving 3 Intel 8086 micros.
- Jim Welsh gave a talk on "Pascal Plus" about extending Pascal for current processes.
- Marshall Harris gave a talk on "A Structured Programming Interpretable Instruction Language - or - Against Patriarchal Programming Languages" about SIPSIL, an alternative to Pascal.
- Jeff Rohl gave a talk "On Sets in Programming" about applications with Pascal sets.
- A. M. Lister gave a talk on "Constructive Proofs of Monitors" providing experience with Pascal-Plus.

The text of the invited papers (4) to this conference appeared as Volume 1 Number 1 of a new Australian computer science journal called the <u>Australian Computer Science Communications</u>. Also included were the prepared texts of the Panel Discussion by Arthur Sale, Jeff Rohl, and John Bennett on "What is Computer Science?". A report was included on computer science in China.

This conference demonstrated the vitality of computer science research in Australia and will definitely become a respected institution. - Andy Mickel

The SIGPLAN Compiler Construction Conference was held in Boulder on August 8-10 and papers were presented on some Pascal topics:

- Gilbert J. Hansen, Gerald A. Shoults, and Joe Cointment of Texas Instruments gave a talk on "Construction of a Transportable, Multipass Compiler for Extended Pascal"
- Richard J. LeBlanc of Georgia Tech and Charles N. Fischer of the University of Wisconsin gave a talk "On Implementing Separate Compilation in Block-Structured Languages" which gives examples using the Pascal 1100 compiler.
- Richard L. Sites and Daniel R. Perkins of UC San Diego gave a talk on "Machine-Independent Pascal Code Optimization".
- Philip A. Nelson of Lawrence Livermore Labs gave a talk on "A Comparison of Pascal Intermediate Languages"

The proceedings of this conference appeared as <u>SIGPLAN Notices</u> Vol 14 No 8, August, 1979.

Another rich conference was held in Sydney during September 10-11 being a Symposium on Language Design and Programming Methodology sponsored by the Australian Atomic Energy Commission and the University of New South Wales. The conference was organized by Jeff Tobias and papers covered the whole range of topics from algorithms to data structures, practice and experience. Invited speakers were Niklaus Wirth and Dennis Ritchie.

Report on the DECUS (Digital Equipment Corporation Users Society) Pascal SIG (Special Interest Group)

by Richard J. Cichelli

This is a second hand report of the activities of the Pascal SIG meeting at the Fall, 1978 DECUS symposium. It is based on conversations with John Iobst (also of ANPA/RI) who attended as PUG liaison and chaired a standards workshop.

John Barr (Department of Computer Science, University of Montana, Missoula, Montana 59812) is chairman of the 1200 member Pascal SIG.

The SIG's standards subcommittee reviewed many suggested "enhancements". to Pascal. The commendably short report of the subcommittee is presented here in full.

PROPOSED PASCAL STANDARD

We propose that the DECUS Standard for the language PASCAL be as follows:

PASCAL is that language defined in the "PASCAL USER MANUAL AND REPORT", with the following two modifications:

1) the addition of the reserved word "forward", to allow two or more procedures or functions on the same level to call each other.

2) a method of specifying the parameter list for procedure or function parameters which are passed by name. This will allow the full type checking of parameters at compile time for all procedures and functions which are used as parameters.

In addition to these modifications to the definition of PASCAL, the following additional conventionalized extensions are suggested:

1) a means of defining "flexible" arrays. The method of choice is that which was presented by Ch. Jacobi in the September 1976 Pascal Newsletter.

2) the "otherwise" construct in the case statement.

3) a method of relative record I/O. It will be either a predefined set of procedure(s) and/or function(s) or an extension of the array mechanism, possibly using the key word "slow".

4) the addition of the reserved word "external". This will allow a standard means of accessing separately compiled subprograms and libraries.

5) the expansion of the concept of constant denotation to include the definition of structured constants. This requires a modification to the syntax of PASCAL so that constants may be defined after types are defined. The cyclic nature of this modification may lead to undefined identifiers. It is suggested that each of the constant, type and var groups be self-consistent to control the problem.

6) the predefined procedures of reset and rewrite to associate system file names with the PASCAL file variable.

We also suggest the continued discussion of:

1) the problem of functions being able to return only simple type results.

2) the comparison of structured types other than alfa (packed array of char) on at least the equality/inequality level.

We also suggest that the following not be considered as part of the language PASCAL:

1) strings

2) module type encapsulation

3) concurrency

4) additional standard types (other than complex)

5) real time process control

- - - - - -

The following excerpt from the DECUS U.S. Board Meeting Report which quotes Mark Lewis, DECUS U.S. Special Users Group Coordinator, shows some of the political problems within DEC and DECUS regarding Pascal.

> <u>SIGs By Any Other Name</u>
>
> It appears that DECUS U.S. has SIGs of two very distinctive types: (A) The Sig that organizes into a somewhat powerful force users of a particular subset of Digital products, and (B) the SIG that attempts to service users with common interests that are not represented by a particular subset of Digital products. Among the former are the traditional product-based SIGs such as the 12-BIT, RSTS, RSX-11/IAS, RT-11 and SIG 18. (The DECsystem-10/20 Group is properly speaking a member of this first group). Among the latter are such diverse groups as BIOMEDICAL, PASCAL, TECO, and many others. Only a few SIGs represent the special case where the group attempts to serve areas that represent a global interest and a product interest. (The DBMS SIG is an excellent example of a failure to fit the dichotomized pattern since it attempts to service those users who use some sort of DBMS and also attempts to serve as a representative for the users of DBMS-11).
>
> The SIGs of the first type generally have a more powerful influence on DECUS, since they represent the largest users of DECUS resources (in terms of Symposium space/time and newsletter pages), and they are the groups to which Digital must maintain formal liaison. In fact it is the need for formal liaisons between Digital and the SIG that discriminates between the two types. Thus, DBMS clearly belongs to the first group because Digital must provide (a) formal counterpart(s) to the SIG, while PASCAL clearly belongs to the second group since no purpose is served by having a formal Digital Counterpart to the SIG.
>
> In general this Board has been very liberal in recognizing new SIGs without regard for the potential demands that SIGs might make on DECUS resources. I now believe it is time we recognized formally that not all SIGs are created equal and that the best method of distributing resources must favor those SIGs in which Digital has an investment. The SIGs in the second group are really camp followers that would never have been organized had not DECUS become a convenient way of reaching a large number of users. Thus, to use my favorite example, the PASCAL SIG has no rationale for coming into existence within DECUS, with its access to users of a very popular processor via a relatively inexpensive process. Compare the costs to DECUS members for access to the PASCAL SIG's newsletters with the costs of the (non-DECUS) PASCAL USERS GROUP.

- - - - - -

Of course Pascal is the only popular high level language which runs with any compatability or reasonable efficiency on PDP 8's, 11's, 10's, and 20's. Possibly the fact that it also runs well on PDP 11 UNIX systems and other non-DEC software environments makes DEC somewhat wary of the Pascal SIG. (It is the fastest growing SIG and it is the third largest.) Whatever the reasons for DEC's failure to wholeheartedly support Pascal, the proposal by DEC's representative on ANSI X3J9 that there be a five year delay in Pascal standardization was firmly rejected. Certainly Pascal users on DEC equipment will welcome the earliest standard possible.

A Report on Pascal Activities at the New Orleans 1979 Spring DECUS Symposium

Bill Heidebrecht
TRW DSSG
One Space Park
Redondo Beach, CA 90278

The 1979 Spring Digital Equipment Computer Users Society (DECUS) U.S. Mini/Midi Symposium was held in New Orleans on April 17-20. Following the trend set two years ago when John Barr (Pascal SIG chairman) resurrected the Pascal SIG, we had a number of interesting and very well attended Pascal sessions, including an excellent paper given by Kathleen Jensen.

The first Pascal session was held on Tuesday, April 17th, and consisted of Digital's Education Computer Systems Group product announcement of VAX-11 Pascal. This product is the University of Washington Pascal compiler, developed under the leadership of Dr. Helmut Golde. The speakers at the meeting included Dr. Golde, Dr. Marvin Solomon (U. of Wisconsin, test site for the compiler), Leslie Miller (Digital Central Engineering), and several Digital managers. The compiler, which was bootstrapped from the CDC Pascal compiler, will probably be available in late 1979. Execution time of compiled Pascal programs is roughly 1.6 times longer than Fortran programs using Digital's optimizing Fortran compiler. While the VAX Pascal compiler has a number of extensions, Leslie Miller mentioned her desire to remain compatible with the standard. This compiler represents Digital's entry into commercial support of Pascal.

Tuesday evening, Barry Smith of Oregon Software gave an introductory tutorial on Pascal. Several hundred people attended this very popular session.

On Wednesday morning there was a session on Pascal standards, led by Justin Walker (Interactive Systems), Leslie Miller, and Barry Smith. (Justin was the convener of the first ANSI X3J9 meeting in December 1978, and Leslie and Barry are both members of X3J9.) The speakers expressed their support of the proposed BSI/ISO standard, and stated their expectation that it would succeed as the international standard. Some of the details of the draft were discussed, and there were many questions and comments from the audience.

Wednesday afternoon Leslie Miller gave a more detailed presentation on the University of Washington VAX Pascal compiler. The responsibilities for the project are as follows:

- Digital - project management, documentation, and technical assistance.
- U. of Washington - compiler development.
- U. of Wisconsin - testing.

The emphasis has been on educational use, and keeping down the cost of running the compiler. Leslie also discussed some of the extensions (such as double and single precision reals, exponentiation operator, dynamic arrays, descriptor parameters, otherwise in the case statement, etc.) The extensions can be flagged as such through the use of a compiler option.

A presentation by James Spann, Gordon Smith and Roger Anderson of Lawrence Livermore Labs was scheduled on "LSI-11 Writeable Control Store Enhancements to UCSD Pascal". Unfortunately, I was unable to attend this interesting session because of a session conflict.

The next Pascal session on Wednesday afternoon was Kathleen Jensen's paper, "Why Pascal?", which I though was the highlight of the entire symposium. Kathleen worked for three years with Niklaus Wirth at ETH in the early 1970's as a research and teaching assistant. She also taught Pascal, worked on some of the compiler implementation details, and of course is the coauthor of the Pascal User Manual and Report. Kathleen spoke about the development of Pascal, its motivation and influences, and gave examples of its use. She discussed the advantages of using Pascal, from both a programmer's as well as a project leader's viewpoint. About 400-500 people attended this session, and Kathleen received a rousing applause at the end of her talk. Kathleen has been employed at Digital since leaving ETH.

Thursday morning the Pascal sessions began with an applications panel discussion led by Linda Carlock of Hughes Aircraft. John Collins of 3M described an "include" preprocessor and a text file inspection program he wrote. Thomas Mathieu of Battelle spoke about an 8086 cross assembler and associated software, all written in Pascal. And I spoke briefly about the Pascal SIG library.

After the Applications Panel, David Miller of GTE Sylvania gave a paper entitled "Why We Had to Change Pascal". David described some fairly extensive changes GTE made to a PDP-11 implementation of Pascal for a realtime application.

A Pascal Implementation Workshop has held on Thursday afternoon. John Barr, Justin Walker and Brian Nelson (University of Toledo) spoke about status of the SIG's implementation of NBS Pascal under UNIX, RSTS, RSX-11 and RT-11. NBS Pascal was written by Brian Lucas and Justin Walker, (both) previously of the National Bureau of Standards. The compiler is usable now for some programs, but it does not yet implement all of standard Pascal. We are working on finishing a few details and implementing it on the above systems, as well as on the VAX-11.

Also Thursday afternoon, Don Baccus of Oregon Software gave an interesting presentation on code optimization in Pascal compilers. Much of his talk was based on techniques used in the OMSI Pascal-2 compiler for the PDP-11. Don discussed code improvement techniques such as constant folding, subscript optimization, common subexpression elimination, short circuit boolean evaluation, and machine specific improvements.

Thursday evening Roger Vossler of TRW gave an informal presentation on our (TRW) implementation of Concurrent Pascal on the VAX. We are using Concurrent Pascal on our VAX and four PDP-11's for research in distributed processing.

The last Pascal session was held on Friday. This was the Pascal SIG Business Meeting, in which we started plans for the Fall DECUS Symposium, to be held in San Diego in December 79. One of the other topics discussed was the Pascal SIG library tape copy operation. At the previous symposium we made about 80 copies of the library tape, while at New Orleans we made over 150 copies. We hope to work out better methods of distributing the tape in the future, as we cannot keep up with this growth rate using our present distribution methods.

As the current DECUS Pascal SIG librarian, I have discussed with Rich Cichelli (PN Applications Editor) methods of sharing software between the DECUS Pascal SIG and PUG libraries. Unfortunately, there are a number of problems to consider, such as copyright laws, tape format and character set differences, nonstandard Pascal implementations, cost and method of distribution, etc. For the present we can at least exchange software on a program by program basis between the two libraries.

The New Orleans Pascal SIG tape contains two Pascal compilers for the PDP-11 (Torstendahl's "Swedish" Pascal for RSX 11M, and interim versions of NBS Pascal for RSX 11 and RSTS), and a number of utility programs. Pascal News readers who are interested in obtaining a copy of the DECUS Pascal SIG tape should consult recent editions of the DECUS Pascal SIG Newsletter, or contact an RSX or RSTS Local Users Group.

All in all, I think the New Orleans DECUS Symposium was a success as far as Pascal is concerned. Roughly 25% of the people who preregistered indicated an interest in Pascal. When you consider the size of the Pascal SIG membership (over 1,000), its phenomenal growth rate, and the fact that most of the other DECUS SIGs are organized around Digital products (such as RSX, RSTS, VAX/VMS, etc.) you get some idea of the popularity of Pascal within DECUS.

Pascal Session at ACM '78

by Richard J. Cichelli

An informal evening session devoted to Pascal was held at ACM '78. This excellent meeting was convened by John C. Knight of SIGPLAN and NASA. This was the first joint SIGPLAN and PUG technical session and its success is attributable to the excellent organizational work of John Knight. There were more than 75 attendees (we completely filled the meeting room.)

At John's request, I began the session with a report on the state of PUG and its membership, standards activity, Pascal software tools and Pascal-6000 Release #3. The information given has since appeared in PN #13. The agenda of the session is listed below.

1. Comments on the state of the Pascal world by R. Cichelli
2. Brief announcement by a representative of Computer Science Press about their new text - PASCAL An Introduction to Methodical Programming, W. Findlay and D. A. Watt.
3. "An Interactive Incremental PASCAL Compiler", Bengt Nordstrom, Goteborg, Sweden
4. "PASCAL-I", R. Cichelli, ANPA-RI
5. "Verifiable PASCAL", S. Saib, General Research Corp.
6. "A Parser Generator", Wilhelm Burger, Univ. of Texas
7. "Use of PASCAL in Undergraduate Computer Science Education", R. Leblanc, Georgia Institute of Technology
8. "PASCAL and Structure Charts", H. Cunningham, Tektronix

A few personal comments on the topics: #3 is a description of a planned system. #4 is an existing #3 with 25 installations. #6 is a generator similar to UNIX's YACC. Generated parse tables for Pascal configured for micro's are about 2K bytes! #8 is an interactive graphic editing system which manipulates Nassi-Shneiderman diagrams. Post processing turns the N-S structure charts into Pascal code.

I hope we will soon see articles from the session speakers in PN. A truly fine technical session.

PUG Finances

PUG FINANCES 1977-1978

Here are the details for our finances for the 77-78 academic year by both PUG(USA) and PUG(UK). PUG(AUS) has decided to do independent accounting and will report in the future. We therefore will rebate no more money to them in the future. 78-79 finances will be reported in either issue #17 or #18 after we complete the academic year with the appearance of #16.

PUG(USA) Summary of Accounts:

Income:

$ 7.29	Interest on money in Bank Account
55.70	Contributions
1198.00	Sale of 599 backissues @ $2
8608.00	2152 subscriptions @ $4 (2396 total - 180 UK - 64 AUS)
$ 9868.99	Total income.

Expenses:

$ 145.00	PUG Australasian rebate for money already collected
20.00	people who still owe us money (5 @ $4)!
39.00	postage for 300 renewal reminders (@ $0.13)
1325.14	postage costs for all issues including return postage
2180.79	printing 9/10 - 2000 copies
2112.78	printing 11 - 2000 copies
1676.83	printing 12 - 2500 copies
875.96	reprinting 9/10 - 750 copies
858.34	reprinting 11 - 750 copies
18.62	miscellaneous photocopying, titles, and production costs
420.00	PUG(UK) rebate for 76-77 deficit
$ 9672.46	Total expenditure.

Excess income = $ 196.53

PUG(UK) Summary of Accounts:

Income:

£ 450.00	180 Subscriptions @ £2.50

Expenses:

£ 115.60	printing 9/10 - 350 copies
327.60	printing 11 - 350 copies
227.50	printing 12 - 350 copies
226.37	postage, envelopes, etc.
£ 897.07	Total expenditure.

Excess expenditure = £447.07 = $ 935.24

Notes: No. 9/10 was the last of the discount printings, hence the very low price.
Had the money for all 350 copies been collected, our income would have been £875, which would have left the books approximately in balance.

An attempt to assess the financial health of PUG:

Given that PUG(USA) covers the balance of PUG(UK) then:

$ 158.63	petty cash
193.52	bank account
2696.35	computer center account
$ 3048.50	Liquid assets
- 2236.00	Future obligations (subscriptions for 78-79-80-81-82)
$ 812.50	Total assets + 1550 backissues on hand

$ 196.53	77-78 surplus
334.94	76-77 surplus
875.96 858.34	backissues not yet sold
$ 2265.77	theoretical assets
- 935.24	rebate to PUG(UK)
$ 1330.53	total theoretical assets

- Andy Mickel 79/06/30.

Roster Increment

ROSTER INCREMENT (79/05/14)

Following is a list of PUG members who either joined or changed address or phone number since the last roster increment was printed dated 78/10/31 in Pascal News #13.

01002 DUANE W. BAILEY/ DEPT. OF MATHEMATICS/ AMHERST COLLEGE/ AMHERST MA 01002/ (413) 542-2377
01002 EARL BILLINGSLEY/ UNIVERSITY COMPUTING CENTER/ G.R.C./ UNIV. OF MASSACHUSETTS/ AMHERST MA 01002/ (413) 545-2690
01003 JEFF BONAR/ COMPUTER AND INFO SCI DEPT./ UNIV. OF MASSACHUSETTS/ AMHERST MA 01003/ (413) 545-2744
01060 EDWARD JUDGE/ 73 BRIDGE ST. #10/ NORTHAMPTON MA 01060
01063 BERT MENDELSON/ COMPUTER CENTER/ 215 MCCONNELL HALL/ SMITH COLLEGE/ NORTHAMPTON MA 01063/ (413) 584-2700 X566
01450 PETER D. MARTIN/ TOWNSEND RD. RFD #2/ GROTON MA 01450/ (617) 448-5395
01451 RALPH S. GOODELL/ HILLCREST DRIVE/ HARVARD MA 01451/ (617) 456-8090
01532 JANICE ANN KELSO/ 64 VALENTINE RD./ NORTHBORO MA 01532/ (617) 393-8015 (HOME)/ (617) 493-3272 (WORK)
01545 RICHARD J. BONNEAU/ 6 TANGLEWOOD DRIVE/ SHREWSBURY MA 01545/ (617) 845-1432
01581 GREG MORRIS/ 297 TURNPIKE RD #120W/ WESTBORO MA 01581/ (617) 366-9815
01581 A. LYMAN CHAPIN/ SOFTWARE DEVELOPMENT/ MS A-60/ DATA GENERAL CORP/ 15 TURNPIKE ROAD/ WESTBOROUGH MA 01581/ (617) 366-8911 X3056
01609 STEPHEN R. ALPERT/ COMP. SCI. DEPT./ WORCESTER POLYTECHNIC INSTITUTE/ WORCESTER MA 01609/ (617) 753-1411 X416
01720 LEESON J. I. WINTER/ 490 GREAT RD. APT. 1R/ ACTON MA 01720/ (617) 263-4786
01730 TERRENCE R. CULLEN/ 12 ASHBY ROAD/ BEDFORD MA 01730/ (617) 727-9500
01730 RICHARD DEROSIER/ LINOLEX SYSTEMS INC./ 3M/ 10 CROSBY DRIVE/ BEDFORD MA 01730/ (617) 275-1420
01730 KEN TAKAHASHI/ PRODUCT DEVELOPMENT/ 3M-LINOLEX SYSTEMS/ 10 CROSBY DRIVE/ BEDFORD MA 01730
01730 H. WILLMAN/ GRA-11/ RAYTHEON COMPANY/ HARTWELL RD/ BEDFORD MA 01730/ (617) 274-7100 X4632
01740 JAMES K. SKILLING/ ACOUSTICS VIBRATION AND ANALYSIS/ MS #50/ GENRAD/ ROUTE 117/ BOLTON MA 01740/ (617) 779-2811
01742 KEVIN T. MAHONEY/ STOP 6/ GENRAD INC./ 300 BAKER AVENUE/ CONCORD MA 01742/ (617) 369-4400 X317
01754 WILLIAM BARABASH/ ML3-5/E82/ DIGITAL EQUIPMENT CORP./ 146 MAIN ST./ MAYNARD MA 01754
01754 RICHARD KIMBALL/ 145 WALTHAM ST./ MAYNARD MA 01754/ (617) 897-9004
01754 JOHN A. MORSE/ ML3-2/E41/ DIGITAL EQUIP. CORP./ 146 MAIN ST./ MAYNARD MA 01754/ (617) 493-5801
01754 ISAAC R. NASSI/ ML3-5/E82/ DIGITAL EQUIPMENT CORP./ 146 MAIN STREET/ MAYNARD MA 01754/ (617) 493-4487
01775 JOHN R. GOTTHARDT/ 91 OLD BOLTON ROAD/ STOW MA 01775
01776 WILLIAM GARD/ GRAPHICS SYSTEMS/ RAYTHEON CO./ 528 BOSTON POST ROAD/ SUDBURY MA 01776/ (617) 443-9521
01776 RICHARD HOLMES/ INC./ ELECTRONICS FOR MEDICINE/ 56 UNION AVE./ SUDBURY MA 01776
01776 DAVID PETERSON/ SPERRY RESEARCH/ 100 NORTH RD/ SUDBURY MA 01776/ (617) 369-4000 X250
01824 WALTER J. RATAJ/ ACCUTEST CORP./ 25 INDUSTRIAL AVE./ CHELMSFORD MA 01824/ (617) 256-8124
01842 R. A. FREEDMAN/ P.O. BOX 1136/ LAWRENCE MA 01842
01851 ODD W. RYDEN/ CONTROL EQUIPMENT CORP./ 171 LINCOLN STREET/ LOWELL MA 01851/ (617) 459-0573
01854 CHARLES A. STEELE JR./ MATHEMATICS DEPT/ UNIV. OF LOWELL/ LOWELL MA 01854/ (617) 452-5000 X2512
01862 LES SLATER/ TRANTI SYSTEMS INC./ 1 CHELMSFORD RD/ N. BILLERICA MA 01862/ (617) 667-8321
01862 THOMAS BAKER/ NEW ENGLAND NUCLEAR CORP./ 601 TREBLE COVE RD./ N.BILLERICA MA 01862
01876 BERT BEANDER/ C10/ DIGITAL EQUIPMENT CORP./ 1925 ANDOVER ST./ TEWKSBURY MA 01876/ (617) 851-5071 X2088
01876 REID L. BROWN/ TW/E10/ DIGITAL EQUIPMENT CORP./ 1925 ANDOVER STREET/ TEWKSBURY MA 01876/ (617) 851-5071 X2686
01876 BILL PAGE/ C10/ DIGITAL EQUIPMENT CORP./ 1925 ANDOVER ST./ TEWKSBURY MA 01876/ (617) 851-5071
01880 DAVID L. PRESSBERG/ MASS. COMPUTER ASSOC. INC./ 26 PRINCESS STREET/ WAKEFIELD MA 01880/ (617) 245-9540
01880 ROBERT VINCENT/ ANALOGIC CORP./ AUDUBON ROAD/ WAKEFIELD MA 01880/ (617) 246-0300
01886 STEVEN O. HOBBS/ 87 DEPOT ST./ WESTFORD MA 01886
01890 JOHN W. JORDAN/ 5 THORNTON ROAD/ WINCHESTER MA 01890/ (617) 729-8397
01905 THOMAS J. SOUCY/ MICROCOMPUTER SERVICES/ 13 MILDRED STREET/ LYNN MA 01905/ (617) 599-8014
01908 JOSEPH AYERS/ MARINE SCIENCE INSTITUTE/ NORTHEASTERN UNIV./ EAST POINT/ NANANT MA 01908/ (617) 581-7370
02062 ALAN STRELZOFF/ UNION CARBIDE IMAGING SYSTEMS/ 333 PROVIDENCE HWY./ NORWOOD MA 02062/ (617) 769-5400 X464
02090 ALAN HOCHBERG/ ORTHO INSTRUMENTS/ 410 UNIVERSITY AVE./ WESTWOOD MA 02090
02110 JOSEPH J. GAL/ HELLMAN GAL & CO. INC./ ONE FEDERAL STREET/ BOSTON MA 02110/ (617) 482-7735
02114 ROY A. WILSKER/ COMPUTER NETWORK/ MASS. STATE COLLEGE/ 150 CAUSEWAY STREET/ BOSTON MA 02114/ (617) 727-9500
02115 ROBERT J. LECHNER/ DEPT. OF E.E./ 401 DA/ NORTHEASTERN UNIV./ BOSTON MA 02115/ (617) 437-3046
02116 BARTLEY C. JOHNSON/ 92 BOTOLPH STREET/ BOSTON MA 02116/ (617) 266-8128
02138 NORTON GREENFELD/ BOLT BERANEK AND NEWMAN INC./ 50 MOULTON STREET/ CAMBRIDGE MA 02138/ (617) 491-1850
02139 ERIK T. MUELLER/ 410 MEMORIAL DRIVE/ CAMBRIDGE MA 02139/ (617) 253-1000 X5-8153
02139 JIM PERCHIK/ 295 HARVARD ST. APT 607/ CAMBRIDGE MA 02139/ (617) 354-1993
02139 ALLEN SPRINGER/ SCIENTIFIC CENTER/ IBM/ 545 TECHNOLOGY SQUARE/ CAMBRIDGE MA 02139/ (617) 421-9228
02139 COYT C. TILLMAN JR./ IBM CAMBRIDGE SCIENTIFIC CENTER/ 545 TECHNOLOGY SQUARE/ CAMBRIDGE MA 02139/ (617) 421-9250
02154 TERRY HARRIS/ SM DO/ DEPT 3920/ RAYTHEON CO./ SECOND AVE./ WALTHAM MA 02154
02154 ALAN LILLICH/ SOFTECH INC./ 460 TOTTEN POND ROAD/ WALTHAM MA 02154/ (617) 890-6900/ (617) 926-0768
02154 MICHAEL MCKENNA/ 4209 STEARNS HILL RD./ WALTHAM MA 02154/ (617) 894-9713
02154 MICHAEL ROONEY/ THE BOSTON SYSTEMS OFFICE INC./ 469 MOODY ST./ WALTHAM MA 02154/ (617) 894-7800
02154 MICHAEL T. WYMAN/ INTERACTIVE DATA CORP./ 486 TOTTEN POND ROAD/ WALTHAM MA 02154/ (617) 890-8802
02155 BENJAMIN KUIPERS/ DEPT OF MATHEMATICS/ TUFTS UNIVERSITY/ MEDFORD MA 02155/ (617) 628-5000 X6650
02158 DONALD D. FRENCH/ INSTITUTE FOR ADVANCED PROFESSIONAL S*/ ONE GATEWAY CENTER/ NEWTON MA 02158/ (617) 964-1412
02160 FLOYD O. ARNTZ/ 44 GROVE HILL AVENUE/ NEWTONVILLE MA 02160
02162 PRESCOTT TURNER/ PRIME COMPUTER INC./ 3 NEWTON EXECUTIVE PARK/ NEWTON MA 02162/ (617) 964-1730
02168 EARL M. YARNER/ 195 VARICK RD/ NEWTON MA 02168
02169 GEORGE C. HETRICK/ COMPUTING CENTER/ BOSTON COLLEGE/ CHESTNUT HILL MA 02169/ (617) 969-0100 X3400
02172 TIMOTHY ALLEN/ KEYDATA CORP./ 108 WATER STREET/ WATERTOWN MA 02172/ (617) 237-6930

02173 DOUG CHAMBERLIN/ APPLIED DECISION SYSTEMS/ 33 HAYDEN AVE./ LEXINGTON MA 02173/ (617) 861-7580
02173 GEORGE S. GORDON JR./ 7 COACH RD./ LEXINGTON MA 02173/ (617) 861-0470
02173 FRANK SCHWARTZ/ SOFTWARE ASSISTANCE INC./ 18 HARBELL ST./ LEXINGTON MA 02173/ (617) 862-0581
02173 GEORGE M. SHANNON/ LINCOLN LAB/ J-148G/ M.I.T./ 244 WOOD STREET/ LEXINGTON MA 02173/ (617) 862-5500 X5719
02173 MAUREEN J. STILLMAN/ LINCOLN LAB/ M.I.T./ LEXINGTON MA 02173/ (617) 862-5500
02173 H. YOSHIDA/ NEC SYSTEMS LABORATORY/ FIVE MILITIA DRIVE/ LEXINGTON MA 02173/ (617) 862-6415
02178 ATTN: LIBRARY/ DIALOG SYSTEMS INC./ 32 LOCUST STREET/ BELMONT MA 02178/ (617) 489-2830
02178 JAMES E. SMITH/ 87 BEECH ST. 3RD FL./ BELMONT MA 02178
02181 JAMES M. HUDSON/ CULLINANE CORP./ 20 WILLIAMS ST./ WELLESLEY MA 02181/ (617) 237-6600
02194 ALAN EPSTEIN/ IMLAC CORP/ 150 A ST./ NEEDHAM MA 02194/ (617) 449-4600
02871 DAVID J. DE FANTI/ SUBMARINE SIGNAL DIVISION/ MS 177/ RAYTHEON COMPANY/ P.O. BOX 360/ PORTSMOUTH RI 02871/ (401) 847-8000 X2746
02912 RICHARD B. MILLWARD/ PSYCHOLOGY DEPT./ BROWN UNIVERSITY/ BOX 1853/ PROVIDENCE RI 02912/ (401) 863-2608
03031 H. R. MORSE/ FREY ASSOCIATES/ CHESTNUT HILL RD/ AMHERST NH 03031/ (603) 472-5185
03051 LESLIE J. MILLER/ RFD #3/ 18 WOODCREST AVE./ HUDSON NH 03051/ (603) 889-7226 (HOME)/ 851-5071 X2653 (WORK)
03053 JAMES A. CURTIS/ 10 HUNTER BLVD. / P.O. BOX 498/ LONDONDERRY NH 03053
03060 STEFAN M. SILVERSTON/ 23 DEERHAVEN DR./ NASHUA NH 03060/ (603) 883-3882
03242 J. P. MACCALLUM/ BOX 349/ HENNIKER NH 03242/ (603) 428-7275
03801 JAMES NICHOLS/ 375 OCEAN RD/ PORTSMOUTH NH 03801/ (603) 436-4084
03857 MARK KLEIN/ INFORMATION ENGINEERING/ BOX 198 / 8 BAY ROAD/ NEWMARKET NH 03857/ (603) 659-5891
04469 THOMAS E. BYTHER/ COMPUTING CENTER/ UNIV. OF MAINE/ ORONO ME 04469/ (207) 581-2614
04469 RONALD A. ROHRER/ ELECTRICAL ENGINEERING/ BORROWS HALL/ UNIV. OF MAINE - ORONO/ ORONO ME 04469
06095 JEFFREY KATZ/ DEPT 9488 -4BB/ COMBUSTION ENGINEERING INC./ 1000 PROSPECT HILL ROAD/ WINDSOR CT 06095/ (203) 688-1911 X2600
06103 R. REMBERT ARANDA/ MANAGEMENT SYSTEMS/ HARTFORD BOARD OF EDUCATION/ 249 HIGH STREET/ HARTFORD CT 06103/ (203) 566-6506
06468 RICHARD L. ROTH/ TSA SOFTWARE INC/ 39 WILLIAMS DR./ MONROE CT 06468/ (203) 261-7963
06484 MICHAEL BEETNER/ 22 COBBLESTONE DRIVE/ HUNTINGTON CT 06484/ (203) 929-1035
06484 BRUCE HIBBARD/ 60 SAGINAW TRAIL/ SHELTON CT 06484/ (203) 929-8792
06492 KEN M. MA/ COROMETRICS MEDICAL SYSTEMS INC./ 61 BARNES PARK ROAD NORTH/ WALLINGFORD CT 06492/ (203) 265-5631
06520 ARTHUR PERLO/ DEPT. OF MOLECULAR BIOPHYSICS/ YALE UNIV./ BOX 1937 YALE STATION/ NEW HAVEN CT 06520/ (203) 436-4826
06520 ROBERT W. TUTTLE/ COMPUTER SCIENCE DEPT./ YALE UNIVERSITY/ 10 HILLHOUSE AVE. - DUNHAM LAB./ NEW HAVEN CT 06520/ (203) 436-8160
06602 ATTN: SPCC LIBRARY 24EE/ GENERAL ELECTRIC CO./ 1285 BOSTON AVE./ BRIDGEPORT CT 06602/ (203) 334-1012
06608 CHARLES E. REED/ 3200 PARK AVE./ BRIDGEPORT CT 06608
06787 JOHN V. VILKAITIS/ P.O. BOX 26/ THOMASTON CT 06787/ (203) 283-4232
06810 RODNEY BLACK/ BLDG #2/ BURROUGHS CORP./ 105 NEWTON ROAD/ DANBURY CT 06810/ (203) 792-6000
06856 F. ERIC ROBERTS/ CCF SOFTWARE ENGINEERING/ MS 284/ PERKIN ELMER CORP./ MAIN AVENUE/ NORWALK CT 06856/ (203) 762-1797
06880 MICHAEL BEHAR/ 75 COMPO RD. NORTH/ WESTPORT CT 06880
06896 NICHOLAS R. GETI/ 241 ROUTE 107/ W. REDDING CT 06896/ (203) 544-8109
06897 D. KONIGSBACH/ NATIONAL CSS/ 187 DANBURY ROAD/ WILTON CT 06897/ (203) 762-2511 X559
06902 JAMES MOLONEY/ ITT-TTC/ 1351 WASHINGTON BLVD./ STAMFORD CT 06902/ (203) 357-8000
07044 LAWRENCE E. BAKST/ 100 PARK AVE./ VERONA NJ 07044/ (201) 239-3518
07110 STEVEN R. RAKITIN/ SOFTECH INC./ 492 RIVER RD./ NUTLEY NJ 07110/ (201) 284-3291
07430 JOHN RYZLAK/ WESTERN UNION TELEGRAPH CO./ 90 MCKEE DR./ MAHAWAH NJ 07430/ (201) 529-6472
07602 ALLEN A. WATSON/ PRODUCTION SYSTEMS/ THE RECORD/ 150 RIVER STREET/ HACKENSACK NJ 07602/ (201) 646-4000
07666 RICHARD D. SPILLANE/ DEPT OF MATH/COMPUTER SCIENCE/ FAIRLEIGH DICKINSON UNIV./ TEANECK NJ 07666/ (201) 836-6300 X427
07730 ROBERT HALLORAN/ 21 KERRY DR./ HAZLET NJ 07730/ (201) 264-3162
07733 P. E. RUTTER/ HO 1E-408/ BELL LABORATORIES/ HOLMDEL NJ 07733
07753 J. F. DICKSON/ 100 LAKEWOOD RD./ NEPTUNE NJ 07753
07801 MARTIN NICHOLS/ 100 GUY STREET/ DOVER NJ 07801/ (201) 628-9000 X777 (WORK)/ (201) 361-7180 (HOME)
07821 BEN SCHWARTZ/ 495 CROWS NEST ROAD / FOREST LAKES/ ANDOVER NJ 07821/ (201) 786-5897
07846 RANDOLPH BENTSON/ BOX 476/ JOHNSONBURG NJ 07846/ (201) 852-6935
07876 ROBERT KAST/ 11 CENTER LANE/ SUCCASUNNA NJ 07876/ (201) 584-4119
07922 DENNIS K. THORSON/ 243 MCMANE AVE/ BERKELEY HTS NJ 07922/ (201) 464-9534
07960 L. RIANHARD/ 103 SHADY LANE/ MORRISTOWN NJ 07960/ (201) 533-3021 WORK
08034 LEON S. LEVY/ 1021 MT. PLEASANT WAY/ CHERRY HILL NJ 08034
08536 JOSEPH CUSACK/ 21-01 DEER CREEK DRIVE/ PLAINSBORO NJ 08536/ (609) 799-3088
08540 A. CHARLES BUCKLEY/ ADR SERVICES INC./ ROUTE 206 CENTER/ PRINCETON NJ 08540/ (609) 921-8550 X396\
08540 D. CARACAPPA/ DAVID SARNOFF RESEARCH CENTER/ RCA CORP./ P.O. BOX 432/ PRINCETON NJ 08540
08540 JAMES C. EMERY/ INTERUNIVERSITY COMMUNICATIONS COUNCIL/ EDUCOM/ P.O. BOX 364/ PRINCETON NJ 08540/ (609) 921-7575
08540 DAVID RIPLEY/ SARNOFF RESEARCH CENTER/ RCA CORP./ P.O. BOX 432/ PRINCETON NJ 08540/ (202)
08540 HENRY WOOD/ 259 MT. LUCAS ROAD/ PRINCETON NJ 08540
08541 JOHN C. LOCKHART/ D233/ EDUCATIONAL TESTING SERVICE/ ROSEDALE RD./ PRINCETON NJ 08541/ (609) 921-9000 X3562
08854 ATTN: COMPUTER REFERENCE CENTER/ CCIS/ RUTGERS UNIV./ P.O. BOX 879/ PISCATAWAY NJ 08854/ (201) 932-2296
08854 ATTN: DON T. HO/ TECHNICAL INFORMATION LIBRARY/ PY 1G114A/ BELL LABS/ 6 CORPORATE PLACE/ PISCATAWAY NJ 08854/ (201) 981-6500
08854 NARAIN GEHANI/ 1E-134/ BELL LABS/ 6 CORP. PLACE/ PISCATAWAY NJ 08854/ (201) 981-3269
08854 RUSSELL J. PEPE/ 142 MOUNTAIN AVE/ PISCATAWAY NJ 08854/ (201) 527-6869
08873 ROBERT BOYLAN/ P.O. BOX 23/ EAST MILLSTONE NJ 08873/ (201) 874-5449
10003 STEVEN L. MITCHELL/ 5 ST. MARKS PLACE APT 3/ NEW YORK NY 10003/ (212) 228-5796
10006 DAVID EISENBERG/ 19TH FLOOR/ CUTTING EDGE OF TECHNOLOGY INC./ 61 BROADWAY/ NEW YORK NY 10006/ (212) 480-0480
10010 TAIWAN CHANG/ 18V/ METROPOLITAN LIFE/ 1 MADISON AVE./ NEW YORK NY 10010/ (212) 578-2258
10010 LUTHER SPERBERG/ EMPIRE STATE REPORT/ 17 LEXINGTON AVE./ NEW YORK NY 10010/ (212) 725-3313
10013 BILL LIPSKY/ 310 GREENWICH ST 38E/ NEW YORK NY 10013
10016 JUAN RADULOVIC/ SADELMI NEW YORK INC./ 2 PARK AVENUE/ NEW YORK NY 10016/ (212) 750-2462
10016 RAMON TAN/ 305 E. 40TH ST. APT. 12W/ NEW YORK NY 10016/ (212) 754-6464
10017 ROBIN KASCKOW/ DECISION STRATEGY CORP./ 708 3RD AVE/ NEW YORK NY 10017/ (212) 687-2660
10020 MICHAEL ROSENBERG/ NBC - 1401W/ 30 ROCKEFELLER PLAZA/ NEW YORK NY 10020/ (212) 664-4444 X5087
10022 LARRY ARONSON/ 16TH FLOOR/ BOEING COMPUTER SERVICES/ 825 THIRD AVE./ NEW YORK NY 10022/ (212) 486-7275
10025 JOHN I. FREDERICK/ 306 W. 100TH ST. APT 81/ NEW YORK NY 10025
10028 MICHAEL OLFE/ 226 E. 83RD ST. APT. 44/ NEW YORK NY 10028/ (212) 794-0178
10028 CHRISTOPHER YORK/ THE SPENCE SCHOOL/ 22 EAST 91 STREET/ NEW YORK NY 10028/ (212) 289-5940
10029 NORMAN R. KASHDAN/ INST. OF COMP. SCI./ MT. SINAI SCHOOL OF MEDICINE/ FIFTH AVE. AT 100TH ST./ NEW YORK NY 10029/ (212) 650-7253
10031 HIDEHIKO TANAKA/ DEPT. OF ELECTRICAL ENGR./ CITY COLLEGE OF NEW YORK/ CONVENT AVE. @ 140TH ST/ NEW YORK NY 10031/ (212) 690-6621
10304 JOHN D. OWENS/ 147 NORWOOD AVE./ STATEN ISLAND NY 10304/ (212) 448-6283
10516 CHARLES D. FOLEY III/ 4 KNOLLWOOD LANE/ COLDSPRING NY 10516/ (914) 265-9602
10549 DANIEL R. MCGLYNN/ 71 N. MOGER AVE./ MT. KISCO NY 10549/ (914) 666-4665
10550 CHARLES PRINDLE/ MAGNETIC ANALYSIS CORP./ 535 SOUTH 4TH AVE./ MT. VERNON NY 10550/ (914) 699-9450
10577 JOSEPH F. SCHAUB JR./ INFORMATION SYSTEMS DEPT./ PEPSI-COLA COMPANY/ PURCHASE NY 10577
10591 GORDON UBER/ 410 BENEDICT AVE APT 3-D/ TARRYTOWN NY 10591
10598 VICTOR S. MILLER/ THOS J. WATSON RESEARCH CENTER/ IBM/ P.O. BOX 218/ YORKTOWN HGTS NY 10598
10598 MARK SEIDEN/ IBM RESEARCH/ PO BOX 218/ YORKTOWN HGTS NY 10598/ (914) 945-2992
10804 GLEN R. J. MULES/ 263 BEECHMONT DRIVE/ NEW ROCHELLE NY 10804/ (914) 235-7323
10901 J. SCOTT DIXON/ 35 PARK AVE. APT 5K/ SUFFERN NY 10901/ (914) 357-1256
10954 JON BANGS/ 3-2 NORMANDY VILLAGE/ NANUET NY 10954/ (914) 623-1222
10964 NORMAN M. EVENSEN/ LAMONT-DOHERTY GEOLOGICAL OBSERVATORY/ PALISADES NY 10964/ (914) 359-2900 X302
10965 ROBERT NORRIS/ LAWLER MATUSKY & SKELLY/ ONE BLUE HILL PLAZA/ PEARL RIVER NY 10965/ (914) 735-8300
10996 ROBERT L. LEECH/ DEPT. OF ELEC. ENGR./ U.S. MILITARY ACADEMY/ WEST POINT NY 10996/ (914) 938-3071
11020 ROBERT LEVINE/ MAIL STA F5/ SPERRY SYSTEMS MANAGEMENT/ GREAT NECK NY 11020
11020 WARREN K. MELHADO/ MAIL STATION H-3/ SPERRY SYSTEMS MGMT./ GREAT NECK NY 11020/ (516) 574-3407
11040 TOM SCALLY/ P.O. BOX 864/ NEW HYDE PARK NY 11040
11415 GILBERT KAPLAN/ 83-52 TALBOT ST./ KEW GARDENS NY 11415
11716 JAMES A. COLE/ MEGADATA CORP./ 35 ORVILLE DRIVE/ BOHEMIA NY 11716/ (516) 589-6800
11725 FRED ROMEO/ 7 FRUITWOOD LANE/ COMMACK NY 11725/ (516) 575-5723
11725 ASHOK SHENOLIKAR/ 22 GREENE DRIVE/ COMMACK NY 11725/ (516) 499-9166
11727 DONALD R. COSCIA/ SUFFOLK C. C. COLLEGE/ 11 FAIRWOOD LN./ CORAM NY 11727/ (516) 233-5291
11767 RICHARD J. LAW/ 75 MIDWOOD AVE/ NESCONSET NY 11767
11772 GEORGE A. CACIOPPO JR./ 238 MARTHA AVENUE/ EAST PATCHOGUE NY 11772/ (516) 286-8475
11776 BILLIE S. GOLDSTEIN/ UNIVERSITY GARDENS - APT. 2D/ 460 OLD TOWN ROAD/ PT JEFFERSON * NY 11776/ (516) 928-3291
11973 ARTHUR L. Y. LAU/ DEPT OF BIOLOGY/ BROOKHAVEN NATIONAL LABORATORY/ UPTON NY 11973/ (518) 345-3394
11973 FRANK LEPERA/ APPLIED MATH. DEPT./ BLDG 515/ BROOKHAVEN NATIONAL LABORATORY/ UPTON NY 11973/ (516) 345-4112
12206 ALLEN BROWN/ MIKROS SYSTEMS CORP/ 845 CENTRAL AVE./ ALBANY NY 12206/ (518) 489-2561
12305 HONOR REYNOLDS/ 33 FERRY ST./ SCHNECTADY NY 12305/ (518) 385-8489 (WORK)
12308 JOHN D. COATES/ COMPUTER CENTER/ UNION COLLEGE/ SCHENECTADY NY 12308/ (518) 370-6293
12309 FRANCIS FEDERIGHI/ 2109 BAKER AVE/ SCHENECTADY NY 12309/ (518) 457-3998
12401 G. KREMBS/ DEPT 66A / BLDG 003/ IBM CORPORATION/ NEIGHBORHOOD ROAD/ KINGSTON NY 12401/ (914) 383-0123
12546 PAUL F. FITTS/ SYSTEMS DEVELOPMENT/ INNOVATEK MICROSYSTEMS INC./ SMITHFIELD ROAD/ MILLERTON NY 12546/ (914) 373-9003
13069 ROBERT NARAD/ 407 S. 3RD ST./ FULTON NY 13069/ (315) 598-1550
13206 JOHN C. WYMAN/ 263 ROXBURY RD./ SYRACUSE NY 13206/ (315) 423-4320
13440 ATTENTION: H. SPAANENBURG/ MEASUREMENT CONCEPT CORPORATION/ 1333 E. DOMINICK STREET/ ROME NY 13440/ (315) 337-1000
13502 THEO RAMAKERS/ ICL INC/ COSBY MANOR RD/ UTICA NY 13502/ (315) 797-5750
14215 ALLAN MOORE/ 69 EASTON/ BUFFALO NY 14215/ (716) 897-2041
14226 MIKE MANTHEY/ C.S. DEPT./ SUNY - BUFFALO/ 4226 RIDGE LEA ROAD/ AMHERST NY 14226/ (716) 831-1351
14527 DAN DORROUGH/ 1103 E. BLUFF DR./ PENN YAN NY 14527
14580 RICHARD ALRUTZ/ 241 W128/ XEROX CORP./ 800 PHILLIPS RD./ WEBSTER NY 14580/ (716) 422-5154
14580 WERNER SCHENK/ TECHNICAL PROGRAMMING SERV./ XEROX CORP./ 800 PHILLIPS ROAD W128/ WEBSTER NY 14580/ (716) 422-5301
14601 LEOB KOPF/ TAYLOR INSTRUMENT CO./ 95 AMES ST./ ROCHESTER NY 14601/ (716) 235-5000
14609 LOUIS B. JAMES/ SOFTWARE ENGINEERING/ COMPUTER CONSOLES INC./ 97 HUMBOLT STREET/ ROCHESTER NY 14609/ (716) 482-5000
14619 DANIEL A. EHMANN/ 165 WINBOURNE ROAD/ ROCHESTER NY 14619/ (716) 436-2271
14620 LARRY GERTZOG/ COMPUTING CENTER/ UNIV. OF ROCHESTER/ 727 ELMWOOD AVE/ ROCHESTER NY 14620/ (716) 275-4181
14627 RICHARD D. MOSAK/ DEPT. OF MATHEMATICS/ MATH SCIENCES BLDG/ UNIV. OF ROCHESTER/ ROCHESTER NY 14627
14650 ATTN: EASTMAN KODAK CO./ 525 ENGINEERING LIBRARY/ KODAK PARK DIV BLDG 23/ ROCHESTER NY 14650
14850 ALISON A. BROWN/ OFFICE OF COMPUTER SERVICES/ G-24 URIS HALL/ CORNELL UNIV./ ITHACA NY 14850/ (607) 256-7341
14850 DAVID J. LEWIS/ MATHEMATICS DEPT./ ITHACA COLLEGE/ ITHACA NY 14850/ (607) 274-3107

15146 HENRY J. BOWLDEN/ WESTINGHOUSE R&D CENTER/ 1310 BEULAH ROAD/ PITTSBURGH PA 15146/ (412) 256-3375
15213 CHUCK AUGUSTINE/ COMPUTATION CENTER/ CARNEGIE MELLON UNIV./ SCHENLEY PARK/ PITTSBURGH PA 15213/ (412) 578-2649
15213 ANDY HISGEN/ COMPUTER SCIENCE DEPT./ CARNEGIE-MELLON UNIVERSITY/ PITTSBURGH PA 15213/ (412) 578-3053
15213 CHARLES Y. MORROW/ COMPUTER ENGR. DIV./ CARNEGIE-MELLON INST. OF RESEARCH/ 4616 HENRY ST./ PITTSBURGH PA 15213/ (412) 578-3361
15213 BRIAN ROSEN/ THREE RIVERS' COMPUTER CORP./ BOX 235 SCHENLEY PARK/ PITTSBURGH PA 15213/ (412) 621-6250
15213 JAMES B. SAXE/ COMPUTER SCIENCE DEPT./ CARNEGIE-MELLON UNIVERSITY/ PITTSBURGH PA 15213/ (412) 518-3073
15213 RICHARD SNODGRASS/ DEPT. OF COMPUTER SCIENCE/ CARNEGIE-MELLON UNIV./ PITTSBURGH PA 15213/ (412) 578-3044
15213 KEVIN WEILER/ SCHOOL OF URBAN AND PUBLIC AFFAIRS/ INSTITUTE OF PHYSICAL PLANNING/ CARNEGIE MELLON UNIV/ SCHENLEY PARK/ PITTSBURGH PA 15213 (412) 578-2177
15229 CAROL SLEDGE/ ON-LINE SYSTEMS INC/ 115 EVERGREEN HEIGHTS DRIVE/ PITTSBURGH PA 15229/ (412) 931-7600
16802 S. BROOKS MCLANE/ DEPT OF PHYSICS/ 104 DAVEY LABS/ PENN STATE UNIV./ UNIVERSITY PK PA 16802/ (814)
17055 E. R. BEAUREGARD/ CODE: 9442T/ NAVY FLEET MATERIAL SUPPORT OFFICE/ MECHANICSBURG PA 17055/ (717) 790-4130/ (717) 766-1446 (HOME)
17257 CHARLES E. MILLER/ RD 5 - CRESCENT DRIVE/ SHIPPENSBURG PA 17257/ (717) 532-5169 (HOME)/ (717) 532-1540 (WORK)
18015 RAYMOND G. MORETZ JR./ 1102 SENECA STREET/ BETHLEHEM PA 18015/ (215) 948-7900 X377/ (215) 691-6902 (HOME)
18042 PETER A. APGAR/ 401 FROST HOLLOW ROAD/ EASTON PA 18042/ (215) 252-2176
18104 THOMAS M. MORRISETTE/ 2219 GREENLEAF ST./ ALLENTOWN PA 18104/ (215) 434-2993
18936 LAWTHER O. SMITH/ GAS SPRING CORP./ 17 COMMERCE DRIVE/ MONTGOMERYVL PA 18936/ (215) 368-7105
18976 FRANCIS W. YEUNG/ P.O. BOX 489/ WARRINGTON PA 18976/ (215) 343-4758
19002 IRA RUBEN/ 2104 LINCOLN DRIVE EAST/ AMBLER PA 19002/ (215) 542-2174
19002 NEIL R. BAUMAN/ HEALTHCOM/ AXE WOOD WEST/ BROADAXE PA 19002/ (215) 643-7330
19085 JAMES SOLDERITSCH/ DEPT OF MATHEMATICS/ VILLANOVA UNIV./ VILLANOVA PA 19085/ (215) 527-2100 X669
19090 WILLIAM L. BAIRD/ 36 WOODHILL DRIVE/ WILLOW GROVE PA 19090/ (215) 659-4929
19101 ROBERT A. EPPING/ ENVIRONMENTAL SYSTEMS/ 1260M/ GENERAL ELECTRIC CO./ 3198 CHESTNUT ST./ PHILADELPHIA PA 19101/ (215) 823-3242
19102 H. R. WRIGHT/ 13TH FLOOR/ BELL OF PENNSYLVANIA/ 1 PARKWAY/ PHILADELPHIA PA 19102/ (215) 466-3478
19104 ATTN: SERIALS DEPT./ DREXEL UNIV. LIBRARIES/ 32ND & CHESTNUT STREETS/ PHILADELPHIA PA 19104
19104 JOHN F. LUBIN/ THE WHARTON SCHOOL/ DE-111 CC/ UNIV. OF PENNSYLVANIA/ PHILADELPHIA PA 19104/ (215) 243-7601
19104 JOSEPH O'ROURKE/ 4103 CHESTNUT ST./ PHILADELPHIA PA 19104
19104 STEPHEN M. PLATT/ 4060 IRVING ST./ PHILADELPHIA PA 19104/ (215) 222-6432
19144 WARREN G. POWELL/ PHILADELPHIA COLLEGE TEXTILES AND SCI*/ SCHOOL HOUSE LANE & HENRY AVE./ PHILADELPHIA PA 19144/ (215) 843-9700
19147 DENIS KALTHOFER/ 613 SOUTH STREET/ PHILADELPHIA PA 19147/ (215) 923-7850
19422 RICHARD D. LADSEN/ MS A-1/ SPERRY UNIVAC/ P.O. BOX 500/ BLUE BELL PA 19422/ (215) 542-4011
19454 KURT MEYLE/ MD #148/ LEEDS & NORTHRUP/ DICKERSON RD./ NORTH WALES PA 19454/ (215) 643-2000 X3033
19518 RICHARD A. JOKIEL/ P.O. BOX 136/ DOUGLASVILLE PA 19518/ (215) 385-6324/ (215) 948-7900
19713 ROBERT F. BASHFORD/ 704 MANFIELD RD./ NEWARK DE 19713
20003 VANESSA AXELROD/ EDS FEDERAL CORP./ 229 PENNSYLVANIA AVE./ WASHINGTON DC 20003/ (202) 546-8700
20012 RICK THOMAS/ 408 DOMER AVENUE/ TAKOMA PARK MD 20012/ (301) 565-2678 (HOME)/ (301) 454-2946 (WORK)
20014 JOHN M. SHAW/ BLDG 36 / ROOM 2A29/ NATIONAL INSTITUTES OF HEALTH/ BETHESDA MD 20014/ (301) 496-3204
20015 W. G. BLASDEL/ 4513 CUMBERLAND AVE./ CHEVY CHASE MD 20015
20015 ROBERT L. MCGHEE/ 4417 BRADLEY LANE/ CHEVY CHASE MD 20015
20018 LARRY LANGDON/ 3132 APPLE RD. N.E./ WASHINGTON DC 20018
20024 GARY A. KUDIS/ 3224/ COMSAT GENERAL CORP./ 950 L'ENFANT PLAZA SW/ WASHINGTON DC 20024/ (202) 554-6438
20034 ROY MADDUX/ FEDERAL SYSTEMS DIV./ IBM/ 10215 FERNWOOD RD./ BETHESDA MD 20034/ (301) 897-3345
20036 N. RAMACHANDRAN/ LEXICO ENTERPRISES/ 1333 NEW HAMPSHIRE AVE. NW - SUITE 510/ WASHINGTON DC 20036/ (202) 457-0320
20052 ATTN: TECHNICAL ASSISTANCE/ UNIVERSITY COMPUTER CENTER/ GEORGE WASHINGTON UNIVERSITY/ 2013 G STREET N.W. #201/ WASHINGTON DC 20052/ (202) 676-6140
20229 STEVE O'KEEFE/ 7328/ U.S. CUSTOMS DATA CENTER/ 1301 CONSTITUTION AVE. N.W./ WASHINGTON DC 20229/ (202) 566-2974
20590 ATTN: COMMANDANT (G-DOE-3/TP54)/ U.S. COAST GUARD/ 2100 2ND ST. SW/ WASHINGTON DC 20590
20601 DONALD H. RINGLER/ MICROWAVE SPACE RESEARCH FACILITY/ NAVAL RESEARCH LABORATORY/ RFD NO.2 BOX 126A/ WALDORF MD 20601
20755 ATTN: S86/ I# PIG/ NATIONAL SECURITY AGENCY/ FT. GEO. MEADE MD 20755/ (301) 688-6015
20760 PEGGY DUNN/ OLD DOMINION SYSTEMS/ 4 PROFESSIONAL DRIVE - SUITE 119/ GAITHERSBURG MD 20760/ (301) 948-5200
20770 LEO R. DAVIS/ 40 LAKESIDE DRIVE/ GREENBELT MD 20770/ (301) 474-9125
20771 ADOLPH GOODSON/ GODDARD SPACE FLIGHT CENTER/ CODE 5331/ NASA/ GREENBELT MD 20771
20776 BETTY A. COLHOUN/ IVY NECK/ HARWOOD PO MD 20776/ (301) 867-2348
20795 KENNETH R. JACOBS/ 10112 ASHWOOD DR./ KENSINGTON MD 20795/ (301) 946-4769
20852 ALLEN E. BENDER/ 5003 MACON ROAD/ ROCKVILLE MD 20852
20854 LOUIS V. RUFFINO/ FEDERAL SYSTEMS DIVISION/ IBM/ 18100 FREDERICK PIKE/ GAITHERSBURG MD 20854/ (301) 840-7978
20854 ATTN: APPLIED BUSINESS COMPUTER SYSTE*/ 12913 MISSIONWOOD WAY/ POTOMAC MD 20854/ (301) 340-8708
20901 DAVID P. WALSH/ 319 HILLMOOR DRIVE/ SILVER SPRING MD 20901
21031 ATTN: GENERAL INSTRUMENT CORPORATION/ C/O TECHNICAL LIBRARY/ 11126 MCCORMICK ROAD/ HUNT VALLEY MD 21031/ (301) 666-8700 X333
21031 WALTER J. KLOS/ DISPLAY DATA CORP./ EXECUTIVE PLAZA IV/ HUNT VALLEY MD 21031/ (301) 667-9211
21040 LAWRENCE W. BAIN JR./ 804 FISHERMAN LANE/ EDGEWOOD MD 21040/ (301) 676-4791
21044 RON GRAVES/ GENERAL PHYSICS CORP./ 1000 CENTURY PLAZA/ COLOMBIA MD 21044/ (301) 730-4055
21045 RICHARD LLEWELLYN/ 5355 RED LAKE/ COLUMBIA MD 21045/ (301) 997-4079
21202 WAYNE N. OVERMAN/ OFFICE OF THE PUBLIC DEFENDER/ 800 EQUITABLE BLDG./ BALTIMORE MD 21202/ (301) 383-7743
21204 EDWARD W. KNUDSEN/ G/157/ AAI CORP./ P.O. BOX 6767/ BALTIMORE MD 21204/ (301) 666-1400
21234 KEVIN A. PARKS/ 1806 DALHOUSIE CT. APT B2/ BALTIMORE MD 21234/ (301) 668-2067
21793 PAUL C. BERGMAN/ DIGITAL SYSTEMS CORP./ 3 NORTH MAIN ST./ WALKERSVILLE MD 21793/ (301) 845-4141
22003 PATRICIA TIMPANARO/ 4504 COMMONS DRIVE #102/ ANNANDALE VA 22003/ (202) 223-5676
22030 WILLIAM F. AMON III/ 13312 PENNYPACKER LANE/ FAIRFAX VA 22030/ (703) 790-8620
22043 ROBERT ROSE/ 2205 GRAYSON PLACE/ FALLS CHURCH VA 22043/ (703) 534-1984
22090 GEORGE W. CHERRY/ 1542 GOLDENRAIN CT./ RESTON VA 22090/ (703) 437-4450
22090 STEPHEN GERKE/ 1646 PARKCREST CIR. #301/ RESTON VA 22090/ (703) 437-4319
22091 RICHARD STADTMILLER/ 1454 GREEMONT CT./ RESTON VA 22091
22101 J. J. LOGAN/ INFODYNAMICS/ 6636 HAZEL LANE/ MCLEAN VA 22101/ (703) 893-5436
22102 DAVID A. GOMBERG/ W-615/ MITRE CORP./ 1820 DOLLEY MADISON BLVD./ MCLEAN VA 22102/ (703) 827-7036
22206 PHILIP R. MYLET/ 3373 S. STAFFORD ST./ ARLINGTON VA 22206/ (202) 692-3585
22209 ARTHUR E. SALWIN/ SUITE 711/ RIVERSIDE RESEARCH INSTITUTE/ 1701 N. FT MYER DR/ ARLINGTON VA 22209/ (703) 522-2310
22302 CRAIG E. JACKSON/ 3778 GUNSTON ROAD/ ALEXANDRIA VA 22302/ (703) 998-8262
22304 THOMAS E. SHIELDS/ 300 SOUTH VAN DORN STREET APT #R113/ ALEXANDRIA VA 22304
22312 MICHAEL D. HURLEY/ 437 N. ARMISTEAD ST. APT #5/ ALEXANDRIA VA 22312
22801 JOSEPH W. MAST/ EASTERN MENNONITE COLLEGE/ HARRISONBURG VA 22801/ (703) 433-2771
22980 ATTN: W. H. GENTRY/ DCPBD LIBRARY/ GENERAL ELECTRIC/ WAYNESBORO VA 22980
23185 KATHLEEN S. MICKEN/ DRAWER EE/ WILLIAMSBURG VA 23185/ (804) 564-9350
23666 ATTN: HAMPTON TECHNICAL CENTER/ C/O DOVI-KURTZE/ KENTON INTERNATIONAL INC./ 3221 NORTH ARMISTEAD AVE./ HAMPTON VA 23666
24501 MARK FURTNEY/ 1427 TUNBRIDGE RD/ LYNCHBURG VA 24501/ (804) 384-5799
27101 A. J. SUTTON/ 1135 WEST FOURTH STREET/ WINSTON-SALEM NC 27101/ (919) 723-4735
27605 LENNY HEATH/ MICRONICS INC/ P.O. BOX 12545/ RALEIGH NC 27605
28704 CARROLL B. ROBBINS JR./ APT 32/ ARDEN ARMS APTS./ ARDEN NC 28704/ (919) 684-0168/ (704) 684-8111 (WORK)
30060 FRANK MONACO/ 679 LOWELL DRIVE/ MARIETTA GA 30060/ (404) 424-1460
30303 E. G. SWARTZMEYER/ INFORMATION SYSTEMS DEPT/ GEORGE STATE UNIVERSITY/ UNIVERSITY PLAZA/ ATLANTA GA 30303/ (404) 658-3883
30306 PAUL D. FIELD/ ATLANTA COMPUTER SYSTEMS/ 1019 ROSEDALE ROAD N.E./ ATLANTA GA 30306/ (404) 872-9968
30313 AL SHEPPARD/ SIR-ATLANTA INC./ 331 LUCKIE STREET NW/ ATLANTA GA 30313/ (404) 522-6317
30327 JOHN P. WEST/ DIGITAL SYSTEMS DESIGN GROUP/ 4559 DUDLEY LANE NW/ ATLANTA GA 30327/ (404) 894-2264
30328 M. L. MCGRAW/ 655 SPALDING DR./ ATLANTA GA 30328/ (404) 394-2017
30332 KOZAI KATSUTOSHI/ GEORGIA TECH/ P.O. BOX 33843/ ATLANTA GA 30332/ (404) 874-7881
30332 JOHN PEATMAN/ SCHOOL OF EE/ GEORGIA TECH/ ATLANTA GA 30332/ (404) 894-2901
30332 JERRY W. SEGERS/ OFFICE OF COMPUTING SERVICES/ GEORGIA INSTITUTE OF TECHNOLOGY/ ATLANTA GA 30332/ (404) 894-4676
30341 MIKE HAYES/ 4122 ADMIRAL DRIVE/ CHAMBLEE GA 30341/ (404) 451-1176
32308 C. EDWARD REID/ P.O. BOX 12578/ TALLAHASSEE FL 32308/ (904) 488-2451
32901 CRAIG NELSON/ 635 AUBURN AVENUE/ MELBOURNE FL 32901/ (305) 727-3207
32905 STEPHEN E. WOODBRIDGE/ 642 STEARNS AVE./ PALM BAY FL 32905/ (305) 727-5202
33143 S. M. MINTON/ 6562 S.W. 76 TERRACE/ SOUTH MIAMI FL 33143
33319 A. I. STOCKS/ 3730 INVERRARY DRIVE #1-W/ LAUDERDALE FL 33319
33334 J. NIEL HAYNIE/ NORTH RIDGE DATA INC./ 971 E. COMMERCIAL BLVD./ FT. LAUDERDALE FL 33334/ (305) 771-6344
33432 ROBERT K. STEVENS/ 601 GOLDEN HARBOUR DRIVE/ BOCA RATON FL 33432/ (305) 391-6213
33601 JOHN L. HALL JR./ DC 156/ GTE DATA SERVICES INC./ P.O. BOX 1548/ TAMPA FL 33601/ (813) 224-3286
33620 DAVID B. CAMERON/ COMPUTER RESEARCH CENTER/ INSTRUCTION AND RESEARCH SYSTEMS/ UNIV. OF SOUTH FLORIDA/ TAMPA FL 33620/ (813) 974-2585
35773 MALCOLM GILLIS/ MEGA CORP./ 1001 REYNOLDS RD./ TONEY AL 35773/ (205) 828-0922/ (205) 453-1455
35805 JERRY R. BROOKSHIRE/ 3402 WILKS PL SW/ HUNTSVILLE AL 35805/ (205) 881-9539
35805 MIKE D. PESSONEY/ ANALYSTS INTERNATIONAL CORP./ 2317 BOB WALLACE AVE SW/ HUNTSVILLE AL 35805/ (205) 533-4220
36582 ROY KEELEY JR/ RT. 3 BOX 316/ THEODORE AL 36582/ (205) 973-2516
37076 LARRY D. BOLES/ 649 DENVER DRIVE/ HERMITAGE TN 37076/ (615) 885-1942
40506 LAVINE THRAILKILL/ COMPUTING CENTER/ 72 MCVEY HALL/ U OF KENTUCKY/ LEXINGTON KY 40506/ (606) 258-2916
43201 ATTN: COMPUTER CENTER LIBRARY/ BATTELLE MEMORIAL INSTITUTE/ 505 KING AVE./ COLUMBUS OH 43201/ (614) 424-7329
43201 KEVIN CADMUS/ BATTELLE COLUMBUS LABS/ 505 KING AVENUE/ COLUMBUS OH 43201/ (614) 424-7331
43402 REX KLOPFENSTEIN JR/ 400 NAPOLEON RD APT 332/ BOWLING GREEN OH 43402/ (413) 353-5311
44022 TOM ZWITTER/ 17991 MILLSTONE RD./ CHAGRIN FALLS OH 44022/ (216) 543-5405
44092 EDWARD S. MALLINAK JR./ BAILEY CONTROLS CO./ 29801 EUCLID AVE. 2F8/ WICKLIFFE OH 44092/ (216) 943-5500 X2821
44103 DALE BRAINARD/ SOFTWARE ENGR. - TURNING MACHINE DIV./ WARNER & SWASEY COMPANY/ 5701 CARNEGIE AVENUE/ CLEVELAND OH 44103/ (216) 368-5000
44107 STEVEN B. HALL/ 1599 ORCHARD GROVE/ LAKEWOOD OH 44107/ (216) 521-4178
44115 KARL J. CASPER/ DEPT. OF PHYSICS/ CLEVELAND STATE UNIV./ CLEVELAND OH 44115/ (216) 687-2432
44124 STUART W. ROWLAND/ 1436 GOLDENGATE BLVD. #G4/ MAYFIELD HTS OH 44124/ (216) 473-0347
44139 ROBERT STRADER/ WARNER & SWASEY RESEARCH DIV./ 28999 AURORA RD./ SOLON OH 44139/ (216) 368-6178
44141 DONALD E. WHILE/ RESEARCH AND DEVELOPMENT/ B.F.GOODRICH/ 9921 BRECKSVILLE ROAD/ BRECKSVILLE OH 44141/ (216) 526-4311
44202 AUSTIN CHANEY/ 738-2 CLARIDGE LANE/ AURORA OH 44202/ (216) 562-7289
45214 JAMES HARGREAVES/ P.O. BOX 14734/ CINCINNATI OH 45214/ (513) 385-7048
45219 LARRY BEITCH/ 458 LLOYD PLACE/ CINCINNATI OH 45219/ (513) 621-8275
45324 RICHARD L. TUCKER/ 8007 PHILADELPHIA DR./ FAIRBORN OH 45324
45387 JOHN S. WADDELL/ 113 EAST NORTH COLLEGE STREET/ YELLOW SPRINGS OH 45387/ (513) 767-9157
45429 BOB MYERS/ 4941 ACKERMAN BLVD./ KETTERING OH 45429/ (513) 434-9548
45433 STEVEN ROGERS/ 1011 MIDDY DR./ WPAFB OH 45433/ (513) 253-5860

45701 MARK L. OLSON/ DEPT OF CHEMISTRY/ OHIO UNIVERSITY/ ATHENS OH 45701

45840 EDGAR N. SVENDSEN/ 1823 PARK ST./ FINDLAY OH 45840/ (419) 422-8908

46240 BILL ELLIOTT/ 2649 MARINA DRIVE/ INDIANAPOLIS IN 46240/ (317) 253-1085

46514 NED J. KISER/ CROWN INTERNATIONAL INC./ 1718 W. MISHAWAKA RD./ ELKHART IN 46514/ (219) 294-5571

46628 CHARLES DAVIS/ GENERAL MICROCOMPUTER INC./ 165/ COMMERCE DRIVE/ SOUTH BEND IN 46628/ (219) 233-9171

46755 WILLIAM G. BENTLEY/ KING-SEELEY THERMOS CO./ KENDALLVILLE IN 46755

46808 DALE GAUMER/ GOVT. & INDUSTRIAL DIV./ MAGNAVOX/ 1313 PRODUCTION ROAD/ FORT WAYNE IN 46808/ (219) 482-4411

47150 DOUGLAS H. QUEBBEMAN/ COMPUTING SERVICES/ INDIANA UNIV. - SOUTHEAST/ 4201 GRANTLINE ROAD/ NEW ALBANY IN 47150/ (812) 945-2731 X287

47401 PAUL E. DAWSON/ WRUBEL COMPUTER CENTER/ 12 MEMORIAL/ INDIANA UNIV./ BLOOMINGTON IN 47401/ (812) 337-9255

47401 DAVE DELAUTER/ WRUBEL COMPUTER CENTER/ 78 HPER BLDG/ INDIANA UNIV./ BLOOMINGTON IN 47401/ (812) 337-1911

47401 DAVID S. WISE/ COMPUTER SCIENCE DEPT./ 101 LINDLEY HALL/ INDIANA U/ BLOOMINGTON IN 47401/ (812) 337-4866

47905 MARK SENN/ 107 DIGBY ROAD/ LAFAYETTE IN 47905

48076 LAURENCE L. RAPER/ 29497 SPRING HILL DR./ SOUTHFIELD MI 48076/ (313) 559-6781

48103 JAMES BLYTHE/ THE GREAT LAKES SOFTWARE SYSTEMS LTD./ 5 RESEARCH DRIVE/ ANN ARBOR MI 48103/ (313) 663-6533

48103 JAMES W. KUIPER/ 542 LINDEN LANE/ ANN ARBOR MI 48103/ (313) 994-3500 X489 (WORK)/ (313) 663-7653 (HOME)

48103 TOM WEISZ/ BALANCE TECHNOLOGY INC./ 120 ENTERPRISE DRIVE/ ANN ARBOR MI 48103/ (313) 769-2100

48104 JONATHAN BAUER/ COMSHARE INC./ 3001 S. STATE ST./ ANN ARBOR MI 48104/ (313) 994-4800

48104 W. J. HANSEN/ INC./ SYCOR/ 100 PHOENIX DRIVE/ ANN ARBOR MI 48104/ (313) 995-1234

48104 DAVID LIPPINCOTT/ ANN ARBOR COMPUTER CORP./ 3211 PACKARD RD/ ANN ARBOR MI 48104/ (313) 971-3740

48105 JAMES BLYTHE/ GREAT LAKES SOFTWARE SYSTEMS LTD./ 5 RESEARCH DR./ ANN ARBOR MI 48105/ (313) 663-6533

48105 WILLIAM E. BULLEY/ 314 CLOVERDALE/ ANN ARBOR MI 48105/ (313) 995-2188

48105 MARK HERSEY/ 1114 MAIDEN LANE COURT APT. 112/ ANN ARBOR MI 48105/ (313) 994-3934/ (517) 355-1764 (OFFICE)

48106 RICHARD C. VILE JR./ NORTHERN TELECOM/ 100 PHOENIX DR. - 4 W.T./ ANN ARBOR MI 48106/ (313) 973-6851

48130 MARK WOLCOTT/ 5051 MAST ROAD/ DEXTER MI 48130/ (313) 426-2034 HOME/ (313) 668-4313 WORK

48176 LARRY ENGELHARDT/ P.C.S./ 750 N. MAPLE ROAD/ SALINE MI 48176/ (313) 429-4971

48176 DAVID MATTHEWS/ PROCESS COMPUTER SYSTEMS/ 750 N. MAPLE RD./ SALINE MI 48176/ (313) 429-4971

48176 JOHN VAN ROEKEL/ PROCESS COMPUTER SYSTEMS/ 750 N. MAPLE RD./ SALINE MI 48176/ (313) 429-4971

48197 ROBERT M. OTTOSEN/ 4444 SWISS STONE LANE E. #3C/ YASILANTI MI 48197/ (313) 434-4969

48197 RICHARD L. MAHN/ 2473 DRAPER/ YPSILANTI MI 48197/ (313) 761-3050

48239 ATTN: RCS DATA SYSTEMS/ 26032 FIVE MILE ROAD/ DETROIT MI 48239/ (313) 532-0554

48640 BOB METZGER/ COMPUTER TECHNOLOGY DEV./ DOW CHEMICAL CO./ 2040 DOW CENTER/ MIDLAND MI 48640/ (517) 636-1352

48804 GEORGE SARGENT/ 4961 SIOUX WAY/ OKEMOS MI 48804/ (517) 353-3187

48823 THOMAS W. SKELTON/ 315 WEST SAGINAW STREET/ EAST LANSING MI 48823/ (517) 332-4368/ (517) 351-2530

49007 ATTN: COVELL & HARWOOD CONSULTANTS/ 714 ISB BLDG./ KALAMAZOO MI 49007/ (616) 382-6665

49007 MARK T. O'BRYAN/ PRESTIGE APARTMENT E/ 421 STANWOOD DRIVE/ KALAMAZOO MI 49007

49008 JACK R. MEAGHER/ COMPUTER SCIENCE AND MATHEMATICS/ WESTERN MICHIGAN UNIV./ KALAMAZOO MI 49008/ (616) 383-0095

49269 ROGER KLOEPFER/ 7774 BROWN ROAD/ PARMA MI 49269

49503 MEL PRUIS/ DATA PROCESSING CENTER/ GRAND RAPIDS PUBLIC SCHOOLS/ 143 BOSTWICK N.E./ GRAND RAPIDS MI 49503

49855 R. BHARATH/ 1330 NORWOOD ST. APT 6/ MARQUETTE MI 49855/ (906) 227-2605

50158 DAVID HICKOK/ R.A. ENGEL TECH. CENTER/ FISHER CONTROLS CO./ P.O. BOX 11/ MARSHALLTOWN IA 50158/ (515) 754-3923

52240 HAROLD HARTMAN/ 1912 F ST/ IOWA CITY IA 52240/ (319) 338-7092

52240 JOHN JOHNSON/ 2906 WAYNE AVENUE/ IOWA CITY IA 52240/ (319) 354-1303

52240 CHARLES LARSON/ RR #2/ IOWA CITY IA 52240/ (319) 351-5997

52242 DONALD L. EPLEY/ DEPT. OF COMPUTER SCIENCE/ UNIV. OF IOWA/ IOWA CITY IA 52242/ (319) 353-5605

52333 BOB WERNER/ ROUTE 3 BOX 237A/ SOLON IA 52333/ (319) 644-2657

53012 MARK J. SEBERN/ SEBERN ENGINEERING INC/ W55 N815 CEDAR RIDGE DRIVE/ CEDARSBURG WI 53012/ (414) 375-2200

53092 W. JANSSEN/ 11541 SHORECLIFF LA./ MEQUON WI 53092/ (414) 241-5768

53115 BILL NORTON/ BORG INSTRUMENTS DIV./ BUNKER RAMO CORP./ 902 WISCONSIN STREET/ DELAVAN WI 53115/ (414) 728-5531 X265

53126 THOMAS L. BECK/ UNICO INC./ 3725 NICHOLSON RD/ FRANKSVILLE WI 53126/ (414) 632-6121

53141 D. T. PIELE/ UNIV. OF WISCONSIN - PARKSIDE/ KENOSHA WI 53141/ (414) 553-2231

53151 JOHN J. MERTZ/ WISCONSIN ELECTRICAL MFG CO INC/ BOX 148/ NEW BERLIN WI 53151/ (414) 782-2340

53210 ROBERT F. JAKOB/ 2445 N. 50TH ST./ MILWAUKEE WI 53210/ (414) 445-4800 (HOME)/ (414) 276-9200 (WORK)

53211 W. A. HINTON/ 3469 N. CRAMER ST./ MILWAUKEE WI 53211/ (414) 964-2671 (HOME)/ (414) 963-4005 (OFFICE)

53211 RICHARD LINTON/ 3027 NORTH SHEPARD AVE./ MILWAUKEE WI 53211/ (414) 332-0070

53211 BROOKS DAVID SMITH/ 4473 N. NEWHALL ST./ SHOREWOOD WI 53211/ (414) 963-6413

53214 EDWARD E. KIRKHAM/ ELECTRONIC PRODUCTS DIV./ KEARNEY & TRECKER CORP./ 11000 THEODORE TRECKER WAY/ MILWAUKEE WI 53214/ (414) 476-8300

53280 BABAK CHUBAK/ 168/ WISCONSIN TELEPHONE/ 345 N. 35TH ST./ MILWAUKEE WI 53280/ (414) 456-3000

53706 FRED M. JACOBSON/ ACADEMIC COMPUTING CENTER/ UNIV. OF WISCONSIN - MADISON/ 1210 WEST DAYTON STREET/ MADISON WI 53706/ (608) 262-9553

53706 PAUL C. LUSTGARTEN/ COMPUTER SCIENCE DEPT./ UNIV. OF WISCONSIN/ 1210 W. DAYTON ST./ MADISON WI 53706/ (608) 262-7784

53715 WILLIAM FOLZ/ 1317 MILTON ST./ MADISON WI 53715/ (608) 256-6789

54601 JOHN A. NIERENGARTEN/ COMPUTER CENTER/ UNIV. OF WISCONSIN - LA CROSSE/ LA CROSS WI 54601/ (608) 785-8029

55016 DANIEL DASSOW/ 8745 GREENE AVE. SO./ COTTAGE GROVE MN 55016/ (612) 459-3293

55057 CLAYTON HAAPALA/ CARLETON COLLEGE/ NORTHFIELD MN 55057/ (507) 645-4431 X369

55101 DANIEL ETHIER/ 507 E. NEVADA AVE/ ST. PAUL MN 55101/ (612) 771-3281

55101 CHAD HANSEN/ TIMESHARING SERVICES/ 224-3E/ 3M CENTER/ ST. PAUL MN 55101/ (612) 736-1384

55101 KURT PAPKE/ ES&T/ BLDG 518/ 3M CENTER/ ST. PAUL MN 55101

55104 BRUCE NERASE/ BOX 4155/ ST. PAUL MN 55104/ (612) 645-9401

55104 T. D. POPPENDIECK/ DEPT. OF PHYSICS/ HAMLINE UNIV./ 1536 HEWITT/ ST. PAUL MN 55104/ (612) 641-2293

55107 KERRY SHORE/ ASTRO COM CORP./ 120 WEST PLATO BLVD/ ST. PAUL MN 55107/ (612) 227-8651

55112 ED KATZ/ 3564 N. SNELLING/ ARDEN HILLS MN 55112/ (612) 636-3472

55112 RUSSELL B. KEGLEY/ 316 CLEVELAND AVE S.W. #1B/ NEW BRIGHTON MN 55112/ (612) 636-1758 HOME/ (612) 631-5718 WORK

55112 ROGER E. MILLER/ 4217 SHIRLEE LANE NO./ SHOREVIEW MN 55112/ (612) 483-5374

55112 W. B. CHAPIN/ ARH 242/ CONTROL DATA CORP./ 4201 N. LEXINGTON/ ST. PAUL MN 55112/ (612) 483-4673

55112 WAYNE A. SANDERSON/ 892 SHIRLEE LANE/ ST. PAUL MN 55112/ (612) 482-2712

55112 E. L. STECHMANN/ ARH272/ CONTROL DATA CORP./ 4201 N. LEXINGTON AVE./ ST. PAUL MN 55112/ (612) 482-2181

55113 PETER M. RAMSTAD/ 1260 W. LARPENTEUR/ ST. PAUL MN 55113/ (612) 488-4595

55116 HAROLD MELAMED/ 18 ORME CT./ ST. PAUL MN 55116/ (612) 699-1313

55117 STEPHEN S. MCGRANE/ 330 W. COTTAGE AVE. #104/ ST. PAUL MN 55117

55303 HENRY C. BROM/ 2740 NORTH FERRY ST./ ANOKA MN 55303/ (612) 421-8740

55317 LANCE K. FISHER/ 401 HIGHLAND DRIVE/ CHANHASSEN MN 55317/ (612) 474-5138/ (612) 941-8090 WORK

55372 DUANE W. SEBEM/ CREEKWOOD BOX 258/ PRIOR LAKE MN 55372

55404 VICTOR A. JOHNSON/ MARC CORPORATION/ 2527 COLUMBUS AVE. S./ MINNEPOLIS MN 55404/ (612) 871-4440

55405 ANDREW S. WOYAK/ 245 SHERIDAN AVENUE SOUTH/ MINNEAPOLIS MN 55405/ (612) 374-5377

55406 OTTO BAADE/ 2925 37TH AVE S./ MINNEAPOLIS MN 55406/ (612) 729-6250

55409 CARLIN R. COVEY/ 3917 3RD AVE SO./ MINNEAPOLIS MN 55409/ (612) 827-5202

55409 MARY NOERENBERG/ 4615 1ST AVE. S./ MINNEAPOLIS MN 55409/ (612) 827-1545

55410 DAVID E. COLGLAZIER/ 4434 THOMAS AVE S./ MINNEAPOLIS MN 55410/ (612) 854-4600 WORK/ (612) 920-7792 HOME

55414 RICHARD KUBAT/ 1010 15TH AVE. SE - APT 204/ MINNEAPOLIS MN 55414/ (612) 379-1799

55414 JEFF L. POMEROY/ 1321 6TH STREET S.E./ MINNEAPOLIS MN 55414/ (612) 331-5475

55420 TOM WRIGHT/ D/B/A AERIAL SYSTEMS/ P.O. BOX 20330/ MINNEAPOLIS MN 55420/ (612) 944-3046

55424 D. E. SAARELA/ 4508 W. 64TH ST./ MINNEAPOLIS MN 55424/ (612) 926-2561

55424 LADONNA THOMPSON/ MTS SYSTEMS CORP./ BOX 24012/ MINNEAPOLIS MN 55424/ (612) 944-4022

55426 PAUL MINKIN/ 3141 RHODE ISLAND AVE. S./ ST. LOUIS PARK MN 55426/ (612) 922-7366

55427 DAVID PERLMAN/ 8309 NORTHWOOD PKWY./ MINNEAPOLIS MN 55427/ (612) 546-2627

55427 RICHARD SCHROEDEL/ APT #318/ 2907 HILLSBORO AVE. N./ NEW HOPE MN 55427/ (612) 544-5466

55435 GEORGE H. MUELLER/ HMR INC./ 7200 FRANCE AVE SO./ EDINA MN 55435/ (612) 831-7400

55435 KENT SCHROEDER/ CONTROL DATA CORP./ 4550 WEST 77TH STREET/ EDINA MN 55435/ (612) 830-6487

55435 RON THOMAS/ COMPUTER PRODUCTS CORP./ 7625 BUSH LAKE ROAD/ EDINA MN 55435/ (612) 835-7361

55435 GREG STEELE/ NCR/ 4640 W. 77TH ST/ MINNEAPOLIS MN 55435/ (612) 831-5606

55435 RICHARD A. STONE/ DATA 100 CORP./ 7725 WASHINGTON AVE. SO./ MINNEAPOLIS MN 55435/ (612) 932-8000

55440 JON HANSON/ SYSTEM DEVELOPMENT/ DATA 100 CORP/ BOX 1222/ MINNEAPOLIS MN 55440/ (612) 932-8000

55440 GENE MARTINSON/ SYSTEM DEVELOPMENT/ DATA 100 CORP/ BOX 1222/ MINNEAPOLIS MN 55440/ (612) 932-8000

55440 JEREMY S. NICHOLS/ DIGITAL IMAGE SYSTEMS DIV./ HQM284/ CONTROL DATA CORP./ BOX 1249/ MINNEAPOLIS MN 55440/ (612) 374-4880

55440 DOUG PIHL/ SYSTEM DEVELOPMENT/ DATA 100 CORP/ BOX 1222/ MINNEAPOLIS MN 55440/ (612) 932-8000

55440 BILL SIMMONS/ SYSTEM DEVELOPMENT/ DATA 100 CORP/ BOX 1222/ MINNEAPOLIS MN 55440/ (612) 932-8000

55440 RICHARD SPELLERBERG/ SYSTEM DEVELOPMENT/ DATA 100 CORP/ BOX 1222/ MINNEAPOLIS MN 55440/ (612) 932-8000

55440 JERRY STODDARD/ SYSTEM DEVELOPMENT/ DATA 100 CORP/ BOX 1222/ MINNEAPOLIS MN 55440/ (612) 932-8000

55440 TOM URSIN/ SYSTEM DEVELOPMENT/ DATA 100 CORP/ BOX 1222/ MINNEAPOLIS MN 55440/ (612) 932-8000 (WORK)/ (612) 784-1658 (HOME)

55440 JAMES A. VELLENGA/ SYSTEM DEVELOPMENT/ DATA 100 CORP/ BOX 1222/ MINNEAPOLIS MN 55440/ (612) 932-8000

55440 JIM VERNON/ SYSTEM DEVELOPMENT/ DATA 100 CORP/ BOX 1222/ MINNEAPOLIS MN 55440/ (612) 932-8000

55441 PAUL THOMPSON/ 2ND FLOOR - SOUTHGATE OFF. PLAZA/ CONTROL DATA CORP./ 5001 W 80TH ST./ BLOOMINGTON MN 55441/ (612) 830-6937

55455 CHRIS BOYLAN/ UNIVERSITY COMPUTER CENTER/ 132 SPACE SCIENCE CENTER/ UNIV. OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 439-0707
(612) 376-2895

55455 STEVE BRUELL/ C. SCI. DEPT./ 136 LIND HALL/ UNIV. OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-3958

55455 K. FRANKOWSKI/ COMPUTER SCIENCE DEPARTMENT/ 136 LIND HALL/ U OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-7591

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 MICHAEL ROBERT MEISSNER/ C.SCI. DEPT./ 136 LIND HALL/ U OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-6018

55455 DAVE NAUMAN/ MANAGEMENT SCI. DEPT./ 761 B.A./ U OF MINNESOTA/ WEST BANK/ MINNEAPOLIS MN 55455

55455 JOHN NAUMAN/ SSRFC/ 25 BLEGEN HALL/ U OF MINNESOTA/ WEST BANK/ MINNEAPOLIS MN 55455

55455 STEVEN OYANAGI/ TATE LABORATORY OF PHYSICS/ ROOM 142/ UNIVERSITY OF MINNESOTA/ 116 CHURCH STREET S.E./ EAST BANK/ MINNEAPOLIS MN 55455
(612) 771-6326

55455 MICHAEL PRIETULA/ MGMT. SCIENCES DEPT./ 773 BA/ U OF MINNESOTA/ WEST BANK/ MINNEAPOLIS MN 55455/ (612) 373-7885

55455 J. BEN ROSEN/ C.SCI DEPT./ 136 LIND HALL/ U OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-0133

55455 TIM J. SALO/ UNIVERSITY COMPUTER CENTER/ LAUDERDALE/ U OF MINNESOTA/ MINNEAPOLIS MN 55455/ (612) 376-5607

55455 G. MICHAEL SCHNEIDER/ C.SCI. DEPT./ 136 LIND HALL/ U OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-7582

55455 C. J. WADDINGTON/ SCHOOL OF PHYSICS/ TATE LAB OF PHYSICS/ U OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-3847

55455 KIET T. YEN/ 221 SANFORD HALL/ UNIVERSITY OF MINNESOTA/ MINNEAPOLIS MN 55455/ (612) 376-6837

55455 PETER H. ZECHMEISTER/ UNIVERSITY COMPUTER CENTER/ 227 EXP. ENGR./ U OF MINNESOTA/ EAST BANK/ MINNEAPOLIS MN 55455/ (612) 373-4181

55455 ATTN: SSRFC LIBRARY/ SSRFC/ 25 BLEGEN HALL/ U OF MINNESOTA/ WEST BANK/ MINNEPOLIS MN 55455/ (612) 373-5599

55812 DAN BURROWS/ UMD COMPUTER CENTER/ 178 M.W.ALWORTH HALL/ U OF MINNESOTA - DULUTH/ DULUTH MN 55812/ (218) 726-7587
55812 DAN M. LALIBERTE/ 2015 E 2ND STREET/ DULUTH MN 55812/ (218) 728-6177
55812 DAVID K. TAYLOR/ COMPUTER CENTER/ 172 MWAH/ UNIV. OF MINNESOTA - DULUTH/ DULUTH MN 55812/ (218) 726-7587
55901 ED JOHNSTON/ 715 6TH STREET S.E./ ROCHESTER MN 55901/ (507) 286-2635 WORK/ (507) 288-5383 HOME
55901 DAVE MACHART/ 2412 S.W. 4TH STREET/ ROCHESTER MN 55901/ (507) 286-9147
55901 WILLIAM SAMAYOA/ 1434 34ST NW/ ROCHESTER MN 55901/ (507) 282-9214
55987 HUGH OUELLETTE/ 5072 W. 8TH ST./ WINONA MN 55987/ (507) 452-8732
56464 KEITH BELLAIRS/ LAKE VALLEY DATA SYSTEMS/ R2 BOX 108/ MENAHGA MN 56464/ (218) 732-9677
57701 BYRON G. EVERETT/ 622 E. TALLENT/ RAPID CITY SD 57701/ (605) 342-8797
57709 MIKE HUGHES/ P.O. BOX 393/ RAPID CITY SD 57709/ (605) 348-1090
58107 ATTN: P.S. INC./ BOX 2017/ FARGO ND 58107
58501 JEFF HARLOW/ 1002 N. 4TH STREET - APT #2/ BISMARCK ND 58501
59812 JOHN R. BARR/ COMP. SCI. DEPT./ UNIV. OF MONTANA/ MISSOULA MT 59812/ (406) 243-2883
60004 R. D. STINAFF/ 324 W. BRAESIDE DR./ ARLINGTON HTS IL 60004/ (312) 394-4000 X663
60016 MONTE JAY MELDMAN/ 555 WILSON LANE/ DES PLAINES IL 60016/ (312) 635-4123
60016 MONTE J. MELDMAN/ 555 WILSON LANE/ DES PLAINES IL 60016/ (312) 635-4122
60104 RICHARD VILMUR/ 418 FREDRICK AVE./ BELLWOOD IL 60104
60137 EDWARD N. DEKKER III/ 22W 615 ELMWOOD DRIVE/ GLEN ELLYN IL 60137/ (312) 858-5302
60164 REGIS B. SNYDER JR/ DEPT. 470/ TUBE A2/ GTE AUTOMATIC ELECTRIC LABS/ 400 NORTH WOLF ROAD - BOX 2317/ NORTHLAKE IL 60164/ (312) 681-7100 X4327
60164 PRAKASH THATTE/ GTE AUTOMATIC ELECTRIC LABS/ P.O. BOX 2317/ NORTHLAKE IL 60164/ (312) 681-7090
60174 KEITH GARLAND/ ARTHUR ANDERSEN & CO./ 1405 N. FIFTH AV/ ST. CHARLES IL 60174
60196 G. W. GAUGHRAN/ NUCLEAR DATA INC./ GOLF AND MEACHAM ROADS/ SCHAUMBERG IL 60196/ (312) 884-3600
60196 DAVID R. HOPPE/ NUCLEAR DATA/ GOLF & MEACHAM RDS/ SCHAUMBURG IL 60196/ (312) 884-3654
60201 RICHARD A. KARHUSE/ COMPUTER SCI. RESEARCH LAB./ TECH B626/ NORTHWESTERN UNIV./ 2145 SHERIDAN ROAD/ EVANSTON IL 60201/ (312) 492-5248
60439 RICHARD D. GEORGE/ RAS 208/ ARGONNE NATIONAL LABORATORY/ 9700 S. CASS AVENUE/ ARGONNE IL 60439
60540 EDWARD R. BYRNE/ 464 TICONDEROGA LANE/ NAPERVILLE IL 60540
60540 DAVID J. RYPKA/ 2B-401F/ BELL LABORTORIES/ NAPERVILLE IL 60540/ (312) 690-3766
60542 JOHN R. JACKSON/ 834 SHAGBARK LANE #303/ NORTH AURORA IL 60542/ (312) 840-3522
60601 MIKE COLLIGAN/ DEDICATED SYSTEMS INC./ 180 N. MICHIGAN AVE./ CHICAGO IL 60601/ (312) 372-4222
60618 THOMAS P. HOVEKE/ 3223 W. BERTEAU AVE./ CHICAGO IL 60618/ (312) 661-8017 (WORK)/ (312) 539-8747 (HOME)
60680 DAVID M. WEIBLE/ 203 GRANT HALL/ UNIVERSITY OF ILLINOIS AT CHICAGO CIR*/ BOX 4348/ CHICAGO IL 60680/ (312) 996-8836
61008 FRANK D. DOUGHERTY/ BLACKHAWK BIT BURNERS CLUB/ 325 BEACON DRIVE/ BELVIDERE IL 61008/ (815) 544-5206
61107 STANTON D. ERICSON/ 1816 COUNCIL CREST DR./ ROCKFORD IL 61107/ (815) 399-2943
61625 MARIAN FROBISH/ COMPUTER CENTER/ BRADLEY UNIV./ PEORIA IL 61625/ (309) 676-7611 X468
61701 DAVID C. BRAUGHT/ ILLINOIS WESLEYAN UNIVERSITY/ BLOOMINGTON IL 61701/ (309) 556-3146
61742 LENN S. HUNT/ BOX 302/ GOODFIELD IL 61742/ (309) 965-2617
61752 JACK KOCHER/ RR #1/ LEROY IL 61752/ (309) 962-6891
61801 DICK NORTON/ 291 COORDINATED SCIENCE LAB/ UNIV. OF ILLINOIS/ URBANA IL 61801/ (217) 333-8252
61832 SCOTT HERR/ 3819 N. VERMILION/ DANVILLE IL 61832/ (217) 446-2319
62025 WALT PARRILL/ MID. ILLINOIS COMPUTER CO-OP/ COTTONWOOD ROAD/ EDWARDSVILLE IL 62025/ (618) 288-7268
62563 J. R. WEISTART/ 513 E. MAIN STREET/ ROCHESTER IL 62563
62906 JOE B. MONTGOMERY/ P.O. BOX 462/ ANNA IL 62906/ (618) 833-6013
63045 LARRY MUSBACH/ WESTERN ELECTRIC/ 502 EARTH CITY PLAZA/ EARTH CITY MO 63045
63045 CHARLES NEUMANN/ SOFTWARE ENGINEERING/ AUTOCONTROL INC./ 4284A RIVERLINE DRIVE/ EARTH CITY MO 63045/ (314) 291-8150
63110 MICHAEL W. VANNIER/ MALLINCKRODT INSTITUTE/ 510 SOUTH KINGS HWY/ ST. LOUIS MO 63110/ (314) 454-2291
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 SUE D. BURKLUND/ ATTN: DRXAL-TC/ ALMSA/ P.O. BOX 1578/ ST. LOUIS MO 63188/ (314) 268-5271
63701 LARRY LOOS/ COMPUTER SCIENCE DEPT./ SOUTHEAST MISSOURI STATE UNIV./ CAPE GIRARDEAU MO 63701/ (314) 651-2244
64108 ATTN: DOCUMENTATION CENTER/ UNITED COMPUTING SYSTEMS INC./ 2525 WASHINGTON/ KANSAS CITY MO 64108/ (816) 221-9700
64468 GARY MCDONALD/ DIV. OF MATH / CS/ NORTHWEST MISSOURI STATE UNIV./ MARYVILLE MO 64468/ (816) 582-7141
65211 ATTN: ARJUN REDDY - LIBRARIAN/ HEALTH CARE TECHNOLOGY CENTER/ 137 CLARK HALL/ UNIV. OF MISSOURI/ COLUMBIA MO 65211
65211 DAN SMITH/ CAMPUS COMPUTING CENTER/ 103 LEFEVRE HALL/ UNIV. OF MISSOURI-COLUMBIA/ COLUMBIA MO 65211/ (314) 882-7876
65401 GERALD P. ALLDREDGE/ PHYSICS DEPARTMENT/ UNIV. OF MISSOURI - ROLLA/ ROLLA MO 65401/ (314) 341-4372
66102 DAVID M. ALLEN/ 1317 CENTRAL AVE./ KANSAS CITY KS 66102/ (913) 371-6136 (WORK)/ (913) 381-5588 (HOME)
66216 RUDOLF F. WROBEL/ 12725 W. 55TH TERRACE/ SHAWNEE KS 66216/ (913) 631-5131
66506 WILLIAM J. HANKLEY/ DEPT. OF COMP. SCI./ KANSAS STATE UNIV./ MANHATTAN KS 66506/ (913) 532-6352
66506 BRYAN D. HAROLD/ COMPUTING CENTER/ CARDWELL HALL/ KANSAS STATE UNIV./ MANHATTAN KS 66506/ (913) 532-5311
66506 MIKE MILLER/ COMPUTING CENTER/ CARDWELL HALL/ KANSAS STATE UNIV./ MANHATTAN KS 66506/ (913) 532-6311
67203 JEFF PALMER/ 2303 W. 1ST/ WICHITA KS 67203/ (316) 942-1988
67226 DAN C. RICHARD/ M.S. 19/ NCR/ 3718 NORTH ROCK RD./ WICHITA KS 67226/ (316) 687-5228 (WORK)/ (316) 688-5074 (HOME)
68005 KEN RITCHIE/ 1013 BLUFF ST./ BELLEVUE NE 68005/ (402) 291-7224 (HOME)/ (402) 291-5400 (WORK)
68025 PAT SNYDER/ 1941 EAST 16TH ST./ FREMONT NE 68025
68134 CURT HILL/ 7535 SHERMAN DR./ OMAHA NE 68134/ (402) 471-3701 BUS./ (402) 392-2138 HOME
68503 S. RAY HUTTON/ 1714 N 31ST ST./ LINCOLN NE 68503/ (402) 466-0212
68588 GEORGE NAGY/ DEPT. OF COMP. SCI./ 110 FERGUSON HALL/ U OF NEBRASKA/ LINCOLN NE 68588/ (402) 472-3200/ (402) 472-2402
68701 ATTN: DIRECTOR OF COMPUTER SERVICES/ NORTHEAST TECHNICAL COMMUNITY COLLEGE/ 801 E. BENJAMIN/ NORFOLK NE 68701
69341 GARY J. BOOS/ 2350 CHATEAU WAY/ GERING NE 69341/ (308) 436-4687
70005 JOHN R. SOUVESTRE/ 211 ATHERTON DR./ METAIRIE LA 70005/ (504) 837-7882
70118 ERVING S. PFAU/ COMPUTER LABORATORY/ TULANE UNIVERSITY/ 6823 ST. CHARLES AVE./ NEW ORLEANS LA 70118/ (504) 865-5631
70808 JAN R. WILSON/ 3132 EUGENE ST./ BATON ROUGE LA 70808/ (504) 383-1371
73190 MINEO YAMAKAWA/ PHYSIOLOGY AND BIOPHYSICS H. S. C./ UNIV. OF OKLAHOMA/ BOX 26901/ OKLAHOMA CITY OK 73190/ (405) 271-2226
73505 FRANCIS B. HAJEK/ MATH DEPT./ CAMERON UNIVERSITY/ LAWTON OK 73505/ (405) 248-2200 X49
74004 J. B. KLAHN/ APPLIED AUTOMATIC INC./ 206 RB2 PRC/ BARTLESVILLE OK 74004
74102 KENNETH R. DRIESSEL/ AMOCO RESEARCH/ P.O. BOX 591/ TULSA OK 74102/ (418) 644-3551
74128 NED N. MAYRATH/ 10909 E. 3RD. ST./ TULSA OK 74128/ (918) 437-6720
74128 NED N. MAYRATH/ 10909 E. 3RD ST./ TULSA OK 74128/ (918) 437-6720
74171 JACQUES LAFRANCE/ DEPT. OF MATHEMATICAL SCIENCE/ ORAL ROBERTS UNIV./ TULSA OK 74171/ (918) 492-6161 X2722
74601 MIKE BURGHER/ CONTINENTAL OIL COMPANY/ 378C N PARK/ PONKA CITY OK 74601/ (405) 762-3456 X2752
75006 RONALD DAWES/ 2211 GREEN VALLEY/ CARROLLTON TX 75006/ (214) 234-7653/ (214) 245-3200
75006 TOM EKBERG/ MS 503/ MOSTEK/ 1215 WEST CROSBY ROAD/ CARROLLTON TX 75006
75006 JOHN P. JENKINSON/ 2006 PETERS COLONY/ CARROLLTON TX 75006/ (214) 245-1206
75075 GERALD PFEIFFER/ 3100 WINCHESTER/ PLANO TX 75075/ (214) 423-0597
75075 LEO PUTCHINSKI/ 3313 REGENT DR./ PLANO TX 75075/ (214) 234-7685
75080 MARVIN ELDER/ ELDER COMPUTING CORP./ 801 BUSINESS PARKWAY/ RICHARDSON TX 75080/ (214) 231-9142
75080 ASHOK D. INGLE/ P.O. BOX 2902/ RICHARDSON TX 75080/ (214) 996-2273
75080 D. W. MCCAMMISH/ 908 REDWOOD/ RICHARDSON TX 75080/ (214) 234-8432
75223 WILLIAM LYNN/ BOX 11245/ DALLAS TX 75223
75229 PHILLIP R. CALDWELL/ 3239 DOTHAM LANE/ DALLAS TX 75229
75235 ATTN: LIBRARY/ HEALTH SCIENCE CENTER/ UNIV. OF TEXAS - DALLAS/ 5601 MEDICAL CTR. DR./ DALLAS TX 75235/ (214) 688-2383
75235 ARNOLD H. MUECKE/ MCRC/ UNIV. OF TEXAS HEALTH SCIENCE CENTER/ 5323 HARRY HINES/ DALLAS TX 75235/ (214) 688-3936
75240 ROB SPRAY/ ARTHUR A. COLLINS INC/ 13601 PRESTON RD/ DALLAS TX 75240/ (214) 661-2928
75240 BRADLEY M. TATE/ DATA COMMUNICATIONS DIV./ HARRIS CORP./ P.O. BOX 400010/ DALLAS TX 75240/ (214) 386-2236
75401 PAUL D. HELVICK/ 1910 LOOP 315 E. APT 248/ GREENVILLE TX 75401/ (214) 454-1226
76101 P. L. HUTCHISON/ PLANT MZ 2811/ GENERAL DYNAMICS/ P.O. BOX 748/ FORT WORTH TX 76101/ (817) 732-4811 X3267
77005 SCOTT K. WARREN/ ROSETTA ALGORITHMS/ 5925 KIRBY #215/ HOUSTON TX 77005/ (713) 528-8350
77024 WILLIAM A. MITCHELL/ 365 N. POST OAK LANE/ HOUSTON TX 77024/ (213) 686-3383
77025 JAYASHREE RAMANATHAN/ 3834 GRENNOCH LANE/ HOUSTON TX 77025/ (713) 749-3104
77036 R. L. IRWIN/ SEISCOM/ BOX 36928/ HOUSTON TX 77036/ (713) 789-6020
77036 PETE ZIEBELMAN/ MS 6404/ TEXAS INSTRUMENTS/ 8600 COMMERCE PARK DRIVE/ HOUSTON TX 77036/ (713) 776-6589
77042 WESTON W. HASKELL/ 22 BRIAR HILL DRIVE/ HOUSTON TX 77042/ (713) 789-7678
77043 ATTN: MICROPROCESSOR LABORATORIES INC./ 10690 SHADOW WOOD #110/ HOUSTON TX 77043/ (713) 465-7559
77056 VERNON J. MALLU/ 5366 MCCULLOCH CIRCLE/ HOUSTON TX 77056/ (713) 840-7099
77058 CHARLES W. MCKAY/ UNIV. OF HOUSTON - CLEAR LAKE CITY/ 2700 BAY AREA BLVD - PO BOX 446/ HOUSTON TX 77058/ (713) 488-9386
77072 THOMAS BARBARA/ 6512 S. BRIAR BAYOU DR./ HOUSTON TX 77072/ (713) 933-9701
77074 GARY L. BECHTOLD/ DATA 100 CORP./ 6776 SW FREEWAY #400/ HOUSTON TX 77074/ (713) 977-8833
77092 PAUL L. KELLY/ THE ANALYSTS / SCHLUMBERGER/ 4120 D DIRECTOR'S ROW/ HOUSTON TX 77092/ (713) 686-5516
77092 STANLEY M. SUTTON/ RESOURCE DEVELOPMENT & ENGINEERING/ INTER COMP/ 1201 DAIRY ASHFORD RD./ HOUSTON TX 77092/ (713) 497-8400 WORK
77546 ATTN: INTERMETRICS INC./ 4815 FM 2351 - SUITE 103/ FRIENDSWOOD TX 77546/ (713) 482-4411
77843 STANLEY M. SWANSON/ DEPT OF BIOCHEMISTRY/ TEXAS A&M UNIV./ COLLEGE STA. TX 77843/ (713) 845-1744
78209 FRANCIS A. BROGEN/ 115 RIDGEHAVEN/ SAN ANTONIO TX 78209/ (512) 822-0230
78220 GORDON B. ALLEY/ DIGITAL SYSTEMS/ AUTOMATIC CONTROL ELECTRONICS CO./ P.O. BOX 20264/ SAN ANTONIO TX 78220/ (512) 661-4111
78291 DELL ANTONIA/ HARTE-HAWKS COMMUNICATIONS INC./ P.O. BOX 269/ SAN ANTONIO TX 78291
78704 ROBERT L. BYRNE III/ 1114 E. OLTORF #207/ AUSTIN TX 78704/ (512) 471-3032
78704 FRANK DUNN/ 3622 MANCHACA APT 222/ AUSTIN TX 78704/ (214) 231-3423
78704 JAY TROW/ 2200 DE VERNE/ AUSTIN TX 78704/ (512) 444-5045
78712 STEPHEN P. HUFNAGEL/ APPLIED RESEARCH LAB/ ACOUSTICAL MEASUREMENTS DIV./ UNIV. OF TEXAS/ P.O. BOX 8029/ AUSTIN TX 78712/ (512) 836-1351
78712 L. KIRK WEBB/ ASTRONOMY DEPT./ UNIV. OF TEXAS - AUSTIN/ AUSTIN TX 78712
78731 S. VAN ERP/ TCC CORP./ 3429 EXECUTIVE CENTER DR./ AUSTIN TX 78731/ (512) 345-5700
78746 ROBERT PIERCE/ 3806B ISLAND WAY/ AUSTIN TX 78746/ (512) 327-3313
78751 THORNTON KEEL/ 917 E. 40TH STREET/ AUSTIN TX 78751/ (512) 452-8746
78753 JOHN ENGLAND/ 11606 OAK TRAIL/ AUSTIN TX 78753/ (512) 471-5854 WORK/ (512) 836-0375 HOME
78766 BOB ORR/ BOX 9948/ AUSTIN TX 78766/ (512) 454-4797 X426
79409 JOHN JENSEN/ DEPT. OF MATHEMATICS/ TEXAS TECH UNIVERSITY/ LUBBOCK TX 79409/ (806) 742-2571
79604 JOHN L. WEAVER/ HERALD OF TRUTH/ BUSINESS DEPT./ CHURCH OF CHRIST/ P.O. BOX 2439/ ABILENE TX 79604/ (915) 698-4370
80004 CHARLES P. HOWERTON/ 6740 YOUNGFIELD COURT/ ARVADA CO 80004/ (303) 422-6197
80004 J. RICHARD PEARSON/ 5910 FLOWER ST./ ARVADA CO 80004
80020 JIM TURLEY/ 2315 RIDGE CIRCLE/ BROOMFIELD CO 80020/ (303) 469-4778/ (303) 571-6742
80027 PAULA BARRETT/ STORAGE TECHNOLOGY CORP./ 2270 S. 88TH STREET/ LOUISVILLE CO 80027/ (303) 497-7443
80123 H. JAMES SCHNELKER/ 7932 S. LAMAR COURT/ LITTLETON CO 80123/ (303) 979-8284
80202 ATTN: COMPUTING CENTER/ 221/ UNIVERSITY OF COLORADO - DENVER/ 1100 14TH ST./ DENVER CO 80202/ (303) 629-2583

80202 DAVID HORNBAKER/ 1020 15TH ST. #10K/ DENVER CO 80202/ (303) 573-6717/ (303) 629-2678
80221 DENNIS SIMMS/ REGIS HIGH SCHOOL/ 3539 W 50TH STREET/ DENVER CO 80221/ (303) 433-8471
80222 ARTHUR W. GOTTMAN/ BIOMEDICAL & HOSPITAL SYSTEMS LTD./ 2137 S. BIRCH/ DENVER CO 80222/ (303) 758-0517
80222 R. KENT LEONARD/ 3071 S. RESTER WAY/ DENVER CO 80222/ (303) 499-1000 X6811/X6388 (DAY)/ (303) 629-2895 OR 756-4229 (NITE)
80230 ANNE MONTGOMERY/ P.O. BOX 30204/ LOWRY AFB CO 80230/ (303) 394-2904
80302 ATTN: PASCAL DISTRIBUTION/ COMPUTING CENTER LIBRARY/ UNIVERSITY OF COLORADO/ 3645 MARINE STREET/ BOULDER CO 80302/ (303) 492-8131
80302 DONALD HALFORD/ 1492 COLUMBINE AVE./ BOULDER CO 80302
80302 JAY SCHUMACHER/ 1322 ARAPAHOE/ BOULDER CO 80302
80302 TERRY L. SPEAR/ 419 22ND STREET/ BOULDER CO 80302/ (303) 442-3273
80302 PHILIP R. ZIMMERMAN JR./ 1842 CANYON BLVD. #105/ BOULDER CO 80302/ (303) 447-8591
80303 ATTN: NATIONAL CENTER FOR ATMOSPHERIC*/ P.O. BOX 3000/ BOULDER CO 80303
80303 PAUL H. HALENDA/ 4917 THUNDERBIRD DR. #33/ BOULDER CO 80303/ (303) 499-1468
80307 BRUCE K. RAY/ POLYMORPHIC COMPUTER SYSTEMS/ P.O. BOX 3581/ BOULDER CO 80307/ (303) 530-2210
80401 L. S. HENSHAW/ 2003 BEECH COURT/ GOLDEN CO 80401/ (303) 238-9804
81212 PAUL LEBRETON/ PSITRONICS GROUP SYSTEMS LAB/ 502 ALLISON AVENUE/ CANON CITY CO 81212
81501 BURT E. HARTMANN/ HARTMANN ENGINEERING INC./ P.O. BOX 1238/ GRAND JUNCTION CO 81501/ (303) 243-0776
82071 HENRY R. BAUER III/ COMPUTER SCIENCE DEPT./ UNIVERSITY OF WYOMING/ BOX 3682/ LARAMIE WY 82071/ (307) 766-5134
83401 B. H. ANDERSON/ E.G. & G. IDAHO INC./ P.O. BOX 1625/ IDAHO FALLS ID 83401/ (208) 526-1183
83705 LAURENCE R. LANGDON/ 2710 AUGUSTA ST./ BOISE ID 83705
83814 JACK STEVE/ NORTH IDAHO COLLEGE/ 1000 WEST GARDEN AVE./ COEUR D'ALENE ID 83814/ (208) 667-7422
84102 DAVID L. IRVINE/ MICROPOINT CORP./ 363 SOUTH 5TH EAST/ SALT LAKE CITY UT 84102/ (801) 322-4065
84112 RICHARD C. BRANDT/ PHYSICS DEPT/ UNIV. OF UTAH/ SALT LAKE CITY UT 84112/ (801) 581-6076
84115 MARK MICHELSON/ BECTON DICKINSON IMMUNODIAGNOSTICS/ 180 WEST 2950 SOUTH/ SALT LAKE CITY UT 84115/ (801) 487-8773
84116 RICHARD G. LYMAN/ MS U7-2/ SPERRY UNIVAC/ 322 NORTH 2200 WEST/ SALT LAKE CITY UT 84116/ (801) 539-5192
84147 DON B. HALES/ RESEARCH CENTER/ KENNECOTT COPPER CORP./ P.O. BOX 11299/ SALT LAKE CITY UT 84147/ (801) 322-1533
84601 FARREL OSTLER/ 987 E. 2620 N./ PROVO UT 84601/ (801) 375-3668
85012 DENNIS K. BOSWELL/ IBM CORP./ 4502 N. CENTRAL AVE./ PHOENIX AZ 85012/ (602) 263-2005
85019 C. R. CORLES/ PMSD-P/ MD 530/ HONEYWELL/ 2222 W. PEORIA AVE./ PHOENIX AZ 85019/ (602) 997-3000
85019 R. H. DOUGLAS/ PMSD-P/ MD 530/ HONEYWELL/ 2222 W. PEORIA AVE./ PHOENIX AZ 85019/ (602) 997-3000
85019 R. A. HENZEL/ PMSD-P/ MD 530/ HONEYWELL/ 2222 W. PEORIA AVE./ PHOENIX AZ 85019/ (602) 997-3000
85019 J. C. HUNTINGTON/ PMSD-P/ MD 530/ HONEYWELL/ 2222 W. PEORIA AVE/ PHOENIX AZ 85019/ (602) 997-3000
85019 D. P. METZGER/ PMSD-P/ MD 530/ HONEYWELL/ 2222 W. PEORIA AVE/ PHOENIX AZ 85019/ (602) 997-3000
85019 T. L. PHINNEY/ PMSD-P/ MD 530/ HONEYWELL/ 2222 W. PEORIA AVE/ PHOENIX AZ 85019/ (602) 997-3000
85019 E. H. RACHLIN/ PMSD-P/ MD 530/ HONEYWELL/ 222 W. PEORIA AVE/ PHOENIX AZ 85019/ (602) 997-3000
85019 W. VAUGHN/ PMSD-P/ MD 530/ HONEYWELL/ 2222 W. PEORIA AVE/ PHOENIX AZ 85019/ (602) 997-3000
85021 DAVID R. WALLACE/ GTE AUTOMATIC ELECTRIC LABS/ 11226 N. 23RD. AVE./ PHOENIX AZ 85021/ (602) 995-6930
85028 AUTHOR R. JETER/ 3946 EAST ALTADENA/ PHOENIX AZ 85028/ (602) 996-6921
85201 DENNIS GRAY/ 1543 N. SPRUCE CIRCLE/ MESA AZ 85201/ (602) 833-8830
85202 DOUGLAS W. HAWKINS/ MOTOROLA MICROSYSTEMS/ 2200 W. BROADWAY (M318)/ MESA AZ 85202/ (602) 962-5256
85253 LARRY DI LULLO/ DI LULLO CONSTRUCTION COMPANY/ 8724 NORTH 67TH STREET/ PARADISE VLY AZ 85253/ (602) 991-4556
85254 IAN LEMAIR/ 5030 E. POINSETTIA/ SCOTTSDALE AZ 85254/ (602) 996-5458
85257 JAMES HENDRICKSON/ 7301 E. PIERCE ST./ SCOTTSDALE AZ 85257
85281 JAMES E. HOLBROOK/ ITT COURIER TERMINAL SYSTEMS/ 1515 WEST 14TH STREET/ TEMPE AZ 85281/ (602) 275-7555
85704 DON M. WRATHALL/ 6945 N. VISTA PLACE/ TUCSON AZ 85704/ (602) 538-3582
85715 G. A. KORN/ 6801 OPATAS STREET/ TUCSON AZ 85715/ (602) 298-7054
87002 TOM SANDERSON/ RURAL ROUTE 1 / BOX 459/ BELEN NM 87002
87106 DENNIS S. DUNCAN/ 2948 SANTA CRUZ SE/ ALBUQUERQUE NM 87106/ (505) 266-0126/ (505) 277-5536
87108 ATTN: LOVELACE CENTER FOR THE HEALTH */ 5200-5400 GIBSON BLVD SE/ ALBUQUERQUE NM 87108
87112 DAVID T. SCOTT/ SCOTT SYSTEMS/ 10701 LOMAS N.E. SUITE 114/ ALBUQUERQUE NM 87112/ (505) 293-2757
87115 BRUCE LINK/ DIVISION 1716/ SANDIA LABORATORIES/ ALBUQUERQUE NM 87115/ (505) 264-1281
87185 B. C. CASKEY/ DIVISION 4716/ SANDIA LABORATORIES/ ALBUQUERQUE NM 87185
87185 RONDALL E. JONES/ DIVISION 2642/ SANDIA LABORATORIES/ P.O. BOX 5800/ ALBUQUERQUE NM 87185/ (505) 264-7462
87544 ALBERT F. MCGIRT/ 115 GLENVIEW DR./ LOS ALAMOS NM 87544/ (505) 667-7750
87545 SUE JOHNSON/ MS-540 Q-1/ LOS ALAMOS SCIENTIFIC LAB/ LOS ALAMOS NM 87545/ (505) 667-6515
87701 KIM A. KIRKPATRICK/ P.O. BOX 2790/ LAS VEGAS NM 87701
90010 SANDRA DIRKS/ PAWLUK ADVERTISING INC./ 3660 WILSHIRE BLVD./ LOS ANGELES CA 90010/ (213) 386-1164
90023 GEORGE A. MARTINEZ JR./ 654 1/2 S. SOTO ST./ LOS ANGELES CA 90023/ (213) 262-9827
90024 BRADLEY N. YEARWOOD/ TRANSACTION TECHNOLOGY INC./ 10880 WILSHIRE BLVD./ LOS ANGELES CA 90024/ (213) 879-1212
90025 CALVIN W. JACKSON/ ABACUS PROGRAMMING CORP./ 12301 WILSHIRE BLVD/ LOS ANGELES CA 90025/ (213) 820-6955
90036 PENNY CRANE/ INSTRUCTIONAL SUPPORT GROUP/ CALIFORNIA STATE UNIVERSITY/ 5670 WILSHIRE BOULEVARD/ LOS ANGELES CA 90036/ (213) 852-5789
90045 ATTN: K. MICHAEL - LIBRARIAN/ LOS ANGELES SCIENTIFIC CENTER/ IBM/ 9045 LINCOLN BLVD./ LOS ANGELES CA 90045/ (213) 670-8350
90045 DAVID P. MARTIN/ 9619 BELFORD AVE. #3/ LOS ANGELES CA 90045
90046 KEN SIBERZ/ 1720 N. VISTA STREET/ HOLLYWOOD CA 90046/ (213) 874-7224
90049 JOHN BELEW/ JOHN BELEW ASSOCIATES/ 11621 CHENAULT/ LOS ANGELES CA 90049/ (213) 476-4078
90049 PAUL R. EGGERT/ 1151 AMHERST AV #1/ LOS ANGELES CA 90049/ (213) 826-5397
90065 LYNN BLICKENSTAFF/ SELF-REALIZATION FELLOWSHIP/ 3880 SAN RAFAEL AVE./ LOS ANGELES CA 90065/ (213) 225-2471
90066 EDWARD W. BOLTON/ 4253 MOORE STREET/ LOS ANGELES CA 90066/ (213) 391-9998
90067 MICHAEL HADJIOANNOLL/ SUITE 862/ TICOM SYSTEMS INC./ 10100 SANTA MONICA BLVD./ LOS ANGELES CA 90067/ (213) 552-5328
90068 MOSHA CORNFELD/ 6712 HILLPARK DRIVE - #408/ LOS ANGELES CA 90068/ (213) 876-6270
90230 NORM WHEELER/ 11175 WOOLFORD STREET/ CULVER CITY CA 90230
90245 BOB ROOSTH/ TEXAS INSTRUMENTS/ 831 SOUTH DOUGLAS/ EL SEGUNDO CA 90245/ (213) 973-2571
90266 GENE DREHER/ 128-16TH PLACE/ MANHATTAN BCH CA 90266/ (213) 648-2345
90266 CAROLYN A. ROSENBERG/ FORTH INC./ 815 MANHATTAN AVE./ MANHATTAN BCH CA 90266/ (213) 372-8493
90272 ALEX J. BASKIN/ 18008 SANDY CAPE DR./ PACIFIC PALSDS CA 90272/ (213) 454-4960
90274 DAVID J. GRIEP/ 2204 CHELSEA RD/ PALOS VERDES E CA 90274/ (213) 648-7246
90274 LOUIS BARNETT/ 28203 RIDGEFERN CT./ RANCHO PALOS V CA 90274
90274 JOSEPH A. O'BRIEN/ 29319 GOLDEN MEADOW DRIVE/ RANCHO PALOS V CA 90274/ (213) 377-8657
90274 MARK L. ROBERTS/ RYAN MCFARLAND CORPORATION/ 609 DEEP VALLEY DRIVE/ ROLL.H.ESTATES CA 90274/ (213) 377-0491
90278 TIM LOWERY/ 1926 GATES AVE #2/ REDONDO BEACH CA 90278
90291 PATRICK D. GARVEY/ D 3047/ 7742 REDLANDS ST/ PLAYA DEL REY CA 90291/ (213) 821-5663
90291 BARRY A. COLE/ 540 RIALTO AVE./ VENICE CA 90291/ (213) 396-9376
90302 DONALD E. SCHLUTER/ JOHNSON & ASSOCIATES/ 313 EAST BEACH AVENUE/ INGLEWOOD CA 90302/ (213) 678-3222 (WORK)/ (213) 765-1146 (HOME)
90403 LEE A. BENBROOKS/ P.O. BOX 3248/ SANTA MONICA CA 90403/ (213) 472-1165
90403 CARROLL R. LINDHOLM/ P.O. BOX 3007/ SANTA MONICA CA 90403
90404 LLOYD RICE/ COMPUTALKER CONSULTANTS/ 1730 21ST STREET/ SANTA MONICA CA 90404/ (213) 392-5230
90503 JACK MCDONNELL/ COMPUTER COMMUNICATIONS INC/ 2610 COLUMBIA ST./ TORRANCE CA 90503/ (213) 320-9101
90604 LEE L. C. SORENSEN/ 10226 VICTORIA AVE/ WHITTIER CA 90604/ (213) 941-3609
90631 THEODORE C. BERGSTROM/ CHEVRON OIL FIELD RESEARCH CO./ BOX 446/ LA HABRA CA 90631/ (213) 694-7301
90731 WILLIAM C. COX/ 552C OLD DOCK ST./ TERMINAL IS. CA 90731/ (213) 547-4772
90746 D. M. WILBORN/ PACIFIC DATASYSTEMS/ 1007 E. DOMINGUEZ ST. SUITE F/ CARSON CA 90746/ (213) 538-3982
90801 RAY WEISS/ COMPUTER CAREERS INC./ P.O. BOX 2531/ LONG BEACH CA 90801/ (213) 435-5651
90803 J. F. NIEBLA/ INFOTEC DEVELOPMENT INC./ 5855 NAPLES PLAZA - SUITE 210/ LONG BEACH CA 90803/ (213) 433-5224
90813 M. F. DOORE/ 1015 E 10TH ST./ LONG BEACH CA 90813
91011 GARRETT PAINE/ P.O. BOX 895/ LA CANADA CA 91011/ (213) 354-4047 (WORK)/ (213) 790-3390 (HOME)
91103 JULIAN GOMEZ/ 125-241/ JET PROPULSION LABORATORY/ 4800 OAK GROVE DRIVE/ PASADENA CA 91103/ (213) 354-2112
91103 E. N. MIYA/ MS 125-241/ JET PROPULSION LAB./ 4800 OAK GROVE DRIVE/ PASADENA CA 91103/ (213) 354-3251
91103 SAMUEL M. REYNOLDS/ 238 / 601/ 4800 OAK GROVE/ PASADENA CA 91103/ (213) 354-5311
91107 ATTN: MICROSYSTEMS INC./ 2500 E. FOOTHILL BLVD. SUITE 102/ PASADENA CA 91107/ (213) 577-1471
91107 G. DENNIS BARNES/ BLDG 100 / M.S. 241/ XEROX/ 300 N. HALSTEAD/ PASADENA CA 91107/ (213) 351-2351
91107 BARRY SMITH/ 3343 FAIRPOINT ST./ PASADENA CA 91107/ (213) 798-7246
91107 TOM WOLFE/ 2330 E. DEL MAR BLVD. APT #213/ PASADENA CA 91107/ (213) 354-6662 (WORK)/ (213) 793-4046 (HOME)
91125 LARRY SEILER/ 256-80/ CALIFORNIA INST. OF TECHNOLOGY/ PASADENA CA 91125/ (213) 795-6811 X1879
91301 BRUCE D. WALSH/ 5904 LAKE LINDERO DRIVE/ AGOURA CA 91301/ (213) 889-0529
91303 ARI OLIVEIRA/ SYSTEMS COMPUTING INT'L/ 6919 ETON AVE./ CANOGA PARK CA 91303/ (213) 884-6655
91303 GARY A. RICHARDSON/ BLDG 21 MS 6/ LITTON AERO PRODUCTS/ 6700 ETON AVENUE/ CANOGA PARK CA 91303/ (213) 887-2596
91311 TOM SANDERSON/ MICROSYSTEMS DIVISION/ MAIL STOP 63-02/ PERTEC COMPUTER CORP./ 20630 NORDHOFF/ CHATSWORTH CA 91311/ (213) 998-1800 X256
91320 ATTN: TECHNICAL INFORMATION CENTER/ VENTURA DIVISION/ NORTHRUP CORP./ 1515 RANCHO CONEJO BLVD./ NEWBURY PARK CA 91320/ (805) 498-3131 X1050
91320 C. HENNICK/ 127 DEVIA DR./ NEWBURY PARK CA 91320
91320 MARTIN LIPELES/ AUTOLOGIC INC./ 1050 RANCHO CONEJO BLVD./ NEWBURY PARK CA 91320/ (805) 498-9611 X173
91326 CHARLES RIDER/ 19100 KILLOCH WAY/ NORTHRIDGE CA 91326/ (213) 360-3254
91330 ALOIS GLANC/ DEPT. OF COMP. SCI./ CALIFORNIA STATE UNIV./ NORTHRIDGE CA 91330
91342 CHARLES A. WOLFE/ 13376 DRONFIELD AVE./ SYLMAR CA 91342/ (213) 367-6798
91364 JOHN SPIKER/ 5515 PENFIELD - #125/ WOODLAND HILLS CA 91364/ (213) 346-9108
91367 GENE MURROW/ SUITE E/ 6300 VARIEL AVE/ WOODLAND HILLS CA 91367/ (213) 992-4425
91405 L. F. MELLINGER/ 13622 HART ST./ VAN NUYS CA 91405/ (213) 354-2505
91602 FRED WILSON/ 10519 VALLEY SPRING LANE/ N. HOLLYWOOD CA 91602/ (213) 762-2808
91604 STEVEN J. GREENFIELD/ 4311 COLFAX AVE #226/ STUDIO CITY CA 91604/ (213) 762-6560
91724 RICHARD DIEVENDORFF/ 1040 DARFIELD AVENUE/ CORVINA CA 91724
91761 ROBERT L. RHODES/ DEPT 1-373/ LOCKHEED AIRCRAFT SERVICE CO./ P.O. BOX 33/ ONTARIO CA 91761
91775 WILLIAM Y. FUJIMOTO/ SUNNY SOUNDS/ 927-B E. LAS TUNAS DR./ SAN GABRIEL CA 91775/ (213) 287-1811
91792 DAN L. EISNER/ 2801 E. VALLEY VIEW/ WEST COVINA CA 91792/ (213) 965-8865
92021 V. L. MOBERG/ 1127 FLAMINGO AVE/ EL CAJON CA 92021/ (714) 444-5910
92024 ROGER A. COLLINS/ 1653 OLMEDA ST./ ENCINITAS CA 92024/ (714) 437-5586
92037 W. H. AKESON/ 7425 CAMINITO RIALTO/ LA JOLLA CA 92037/ (715) 294-5944
92037 BORDEN COVEL II/ CONTROL DATA CORP./ 4455 EASTGATE MALL/ LA JOLLA CA 92037/ (714) 542-6312
92037 K. J. HARRIS/ BOX 4455/ LA JOLLA CA 92037/ (714) 452-9252
92037 DENNIS NICKOLAI/ CONTROL DATA CORPORATION/ 4455 EASTGATE MALL/ LA JOLLA CA 92037/ (714) 452-6000
92041 KENNETH C. BONINE/ 7985 ANDERS CIRCLE/ LA MESA CA 92041/ (714) 277-8900 X2589
92067 LANCE A. LEVENTHAL/ EMULATIVE SYSTEMS CO./ P.O. BOX 1258/ RANCHO SANTAFE CA 92067/ (714) 452-0101
92093 J. A. LEVIN/ COMMUNICATIONS DEPT./ D-003/ UNIV. OF CALIFORNIA - SAN DIEGO/ LA JOLLA CA 92093/ (714) 452-4410

92093 TERRENCE C. MILLER/ C-014 A.P.I.S. DEPT./ UNIV. OF CALIF - SAN DIEGO/ LA JOLLA CA 92093/ (714) 452-3889
92106 KENNETH O. LELAND/ 3922 LIGGETT DRIVE/ SAN DIEGO CA 92106/ (714) 225-2176
92110 DWIGHT R. BEAN/ ACADEMIC COMPUTING COORDINATOR/ UNIV. OF SAN DIEGO/ SAN DIEGO CA 92110/ (714) 291-6480 X4417 OR X4201
92110 ROBERT CALDWELL/ ENVIRONMENTAL MANAGEMENT SYSTEMS/ 3045 ROSECRANS STREET SUITE 112/ SAN DIEGO CA 92110/ (714) 223-5551
92110 G. G. GUSTAFSON/ COMPUTER SCIENCES CORP./ 2251 SAN DIEGO AVE./ SAN DIEGO CA 92110
92111 GUY KELLY/ CUBIC WESTERN DATA/ 5650 KEARNEY MESA ROAD/ SAN DIEGO CA 92111
92117 STEVE HARRISON/ 5161 COLE ST./ SAN DIEGO CA 92117/ (714) 273-5242
92122 DAVID KUHLMAN/ 6885 ROBBINS CT./ SAN DIEGO CA 92122/ (714) 453-3436
92123 CARL F. NIELSEN/ ALEXANDER ENGINEERING CO./ 9161 CHESAPEAKE DR./ SAN DIEGO CA 92123/ (714) 292-7418
92127 F. TEMPEREAU/ BURROUGHS CORP./ 16701 W. BERNARDO DR./ SAN DIEGO CA 92127
92128 NEAL A. HENDERSON/ 12561 CRESTA PLACE/ SAN DIEGO CA 92128/ (715) 487-6309
92521 ATTN: DEPT. OF MATHEMATICS/ UNIVERSITY OF CALIFORNIA - RIVERSIDE/ RIVERSIDE CA 92521
92625 PAUL MICHAEL REA/ 701-1/2 BEGONIA/ CORONA DEL MAR CA 92625/ (714) 675-1977
92626 H. W. MOORE/ 3150 LIMERICK LANE/ COSTA MESA CA 92626/ (714) 545-3018
92626 WILLIAM H. SEAVER/ GLOBAL COMPUTER SYSTEMS/ 3176 PULLMAN STREET #104/ COSTA MESA CA 92626/ (714) 754-0292
92627 SHAWN M. FANNING/ 2650 HARLA AVE #121/ COSTA MESA CA 92627/ (714) 545-5148
92634 THOMAS M. NEAL/ BECKMAN INSTRUMENTS/ 2500 N. HARBOR BLVD./ FULLERTON CA 92634/ (714) 871-4848 X 3259
92634 VINCENT VIGUS/ FULLERTON COLLEGE/ 321 EAST CHAPMAN AVE./ FULLERTON CA 92634/ (714) 871-8000
92663 DALE BROWN/ 164 CENTRAL SERVICES/ FORD AEROSPACE/ FORD ROAD/ NEWPORT BEACH CA 92663/ (714) 759-5030
92663 JOE DEVITA/ WESTERN DIGITAL CORP./ P.O. BOX 2180/ NEWPORT BEACH CA 92663/ (714) 557-3550
92663 BOB HUTCHINS/ WESTERN DIGITAL CORP./ P.O. BOX 2180/ NEWPORT BEACH CA 92663/ (714) 557-3550 X335
92663 LARRY A. LOTITO/ WESTERN DIGITAL CORPORATION/ P.O. BOX 2180/ NEWPORT BEACH CA 92663/ (714) 557-3550
92667 W. S. DORSEY/ BOX 5118/ ORANGE CA 92667
92677 JIM GILBERT/ SYSTEMS STRUCTURING TECHNOLOGY/ 30436 NORTH HAMPTON RD./ LAGUNA NIGUEL CA 92677/ (714) 640-5222 WORK/ (714) 495-6039 HOME
92680 DAVID S. BAKIN/ MD #151/ BASIC FOUR CORP./ 14101 MYFORD ROAD/ TUSTIN CA 92680/ (714) 731-5100
92680 GEORGE HOMER/ 13271 NIXON CIRCLE/ TUSTIN CA 92680
92683 MIKE CANADAY/ 15271 QUEENSBOROUGH ST./ WESTMINSTER CA 92683/ (714) 839-4122
92686 FRANK BURGER/ 6750 CHAMPAGNE CIRCLE/ YORBA LINDA CA 92686/ (714) 970-0143
92686 HARRY N. CAMPBELL/ 5721 PLACERVILLE PLACE/ YORBA LINDA CA 92686/ (714) 970-7315
92691 JOHN FRENCH/ 26712 VALPARISO DRIVE/ MISSION VIEJO CA 92691/ (714) 768-3411
92705 C. V. GAYLORD/ GARRETT COMPUTER ASSOCIATES/ 18702 ERVIN LANE/ SANTA ANA CA 92705/ (714) 557-1037
92707 JAMES F. SULLIVAN/ 1330 S. ROSEWOOD/ SANTA ANA CA 92707
92708 W. BRYAN HENNINGTON/ 9770 LA ZAPATILLA CIR./ FOUNTAIN VLY CA 92708/ (714) 963-2368 (HOME)/ (714) 632-4079
92713 GREGORY L. HOPWOOD/ MINICOMPUTER OPERATIONS/ SPERRY UNIVAC/ P.O. BOX C-19504/ IRVINE CA 92713/ (714) 833-2400
92713 OSCAR RTOS/ DEPT. 11-0775/ COMPUTER AUTOMATION/ 18651 VON KARMAN/ IRVINE CA 92713/ (714) 833-8830 X295
92713 MARIUS TROOST/ MINICOMPUTER OPERATIONS/ SPERRY UNIVAC/ P.O. BOX C-19504/ IRVINE CA 92713/ (714) 833-2400 X113
92714 LON ATKINS/ 17112 ARMSTRONG AVE./ IRVINE CA 92714/ (714) 540-8340 X543
92714 JIM KHALAF/ 17112 ARMSTRONG AVE/ IRVINE CA 92714/ (714) 540-8340
92714 RICK RAGER/ 17112 ARMSTRONG AVE./ IRVINE CA 92714/ (714) 540-8340
92714 MARIE WALTER/ SCIENTIFIC-TECHNICAL BOOK CENTER/ 17801 MAIN ST./ IRVINE CA 92714/ (714) 557-8324
92715 PAUL HOLBROOK/ 103B CAMINO - MESA COURT/UCI/ IRVINE CA 92715/ (714) 752-2172
92805 JAMES YORK/ GENERAL AUTOMATION/ 1055 SOUTH EAST STREET/ ANAHEIM CA 92805/ (714) 778-4800 X443
92806 DON LEWIS/ 2880 E. HEMPSTEAD RD./ ANAHEIM CA 92806
92807 WILLIAM F. PHILLIPS/ 482 S. PASEO SERENA/ ANAHEIM CA 92807/ (714) 998-7496
93017 ATTENTION: DAN LAPORTE/ M.S. 72/ SANTA BARBARA RESEARCH CENTER/ 75 COROMAR DRIVE/ GOLETA CA 93017/ (805) 968-3511
93017 THOMAS M. BURGER/ BURROUGHS CORP./ 6300 HOLLISTER AVE./ GOLETA CA 93017/ (805) 964-6881 X456
93017 RON JEFFRIES/ 651 ARDMORE/ GOLETA CA 93017/ (805) 964-8964
93017 STEVE LASSMAN/ IMAGE PROCESSING SOFTWARE/ 5773 DAWSON/ GOLETA CA 93017/ (805) 964-4741
93017 RAY L. ANDERSON/ CONCEPT SYSTEMS/ 6885 TRIGO RD./ ISLA VISTA CA 93017/ (805) 968-6995
93021 P. L. SHIMER-ROWE/ 218 HARRY STREET/ MOORPARK CA 93021
93106 ATTN: USER SERVICES GROUP/ COMPUTER CENTER/ UNIV OF CALIF - SANTA BARBARA/ SANTA BARBARA CA 93106
93111 JIM WINSALLER/ P.O. BOX 6679/ SANTA BARBARA CA 93111/ (805) 685-1626
93277 K. B. HOWARD/ DEPT. OF COMP. SCI./ COLLEGE OF THE SEQUOIAS/ VISALIA CA 93277
93407 R. H. DOURSON/ C.S.C. & STAT. DEPT./ CAL POLY STATE UNIV./ SAN LUIS OBIS* CA 93407/ (805) 546-1255
93407 NEIL W. WEBRE/ DEPT. OF COMP. SCI. AND STAT./ CALIF. POLY. STATE UNIV./ SAN LUIS OBIS. CA 93407/ (805) 481-2969
93555 L. W. LUCAS/ CODE 3132/ NAVAL WEAPONS CENTER/ CHINA LAKE CA 93555/ (714) 939-2836
94010 WILLIAM E. BLUM/ SPCOMMUNICATIONS/ 1 ADRIAN COURT - P.O. BOX 974/ BURLINGAME CA 94010/ (415) 692-5600 X444
94019 PAUL BARINA/ 404 KEHOE AVE./ HALF MOON BAY CA 94019
94025 ARTHUR W. DANA JR./ 1670 EL CAMINO REAL/ MENLO PARK CA 94025
94025 C. ROADS/ COMPUTER MUSIC JOURNAL/ BOX E/ MENLO PARK CA 94025/ (415) 323-3111
94035 CHUCK JACKSON/ MS 210-9/ NASA AMES RESEARCH CENTER/ MOFFETT FIELD CA 94035/ (415) 965-6081
94043 JEANE ABITBOUL/ SCANCOM CORP./ 1957B OLD MIDDLEFIELD WY./ MOUNTAIN VIEW CA 94043/ (415) 967-4211
94043 D. DONAHUE/ JOHN FLUKE MFG. CO. INC./ 630 CLYDE AVE/ MTN. VIEW CA 94043
94043 CARY KORNFELD/ 1758 VILLA ST #15/ MTN. VIEW CA 94043/ (415) 966-3731 (WORK)/ (415) 967-7004 (HOME)
94062 MICHAEL K. STAUFFER/ 3660 ALTAMONT WAY/ REDWOOD CITY CA 94062/ (408) 732-2400 (WORK)/ (415) 367-8135 (HOME)
94086 DENNIS S. ANDREWS/ AMDAHL CORP./ 1250 E. ARQUES AVE/ SUNNYVALE CA 94086/ (408) 746-6301
94086 MICHAEL C. ARYA/ SIGNETICS/ 811 EAST ARQUES AVE/ SUNNYVALE CA 94086/ (408) 739-7700
94086 PETER H. HAAS/ MS 203/ AMDAHL CORP./ P.O. BOX 5070/ SUNNYVALE CA 94086/ (408) 746-7340
94086 RAY HOLT/ SYNERTEK SYSTEMS/ 150 S. WOLFE RD./ SUNNYVALE CA 94086/ (408) 988-5691
94086 MASAHIRO HONDA/ AMDAHL CORP./ 1250 E. ARQUES AVE./ SUNNYVALE CA 94086/ (408) 746-6688
94086 PETER KOOLISH/ 02-996/ AMDAHL CORP./ 1250 EAST ARQUES/ SUNNYVALE CA 94086/ (408) 746-6364 (WORK)/ 446-3156 (HOME)
94086 GEORGE LEWIS/ R & D/ BTI COMPUTER SYSTEMS/ 870 WEST MAUDE AVENUE/ SUNNYVALE CA 94086/ (408) 733-1122
94086 JEFFRY L. PARKER/ 1091 CLEMATIS DRIVE/ SUNNYVALE CA 94086/ (408) 247-0814
94087 THOMAS W. CROSLEY/ SOFTWEST/ 1675 NEW BRUNSWICK AVE./ SUNNYVALE CA 94087/ (408) 737-1927
94087 ALLAN B. DELFINO/ 1504 FANTAIL COURT/ SUNNYVALE CA 94087/ (408) 735-1534
94087 PAUL MILLER/ ENGINEERING/ AVERA TECHNOLOGY/ 1643 WRIGHT AVE./ SUNNYVALE CA 94087/ (408) 732-8218
94087 CRAIG W. REYNOLDS/ 400 E. REMINGTON AVE. - APT C-223/ SUNNYVALE CA 94087/ (408) 245-8106
94087 SAMUEL SOLON/ 575 E. REMINGTON DRIVE #11B/ SUNNYVALE CA 94087/ (408) 739-8950
94088 ROSS R. W. PARLETTE/ CHEMICAL SYSTEMS/ P.O. BOX 358/ SUNNYVALE CA 94088/ (408) 739-4880 X2149
94088 JEFFRY G. SHAW/ P.O. BOX 60457/ SUNNYVALE CA 94088/ (408) 257-7676 (EV+WKE)
94104 ROBERT J. RAKER/ PACIFIC GAS & ELECTRIC CO./ 1 POST ST. - NO. 2200/ SAN FRANCISCO CA 94104/ (415) 781-4211 X1296
94104 IRA SLODODIEN/ AUTOMATED DATA EXCHANGE/ 582 MARKET STREET/ SAN FRANCISCO CA 94104/ (415) 421-8824
94109 BRUCE W. RAVENEL/ LANGUAGE RESOURCES/ 1311 LOMBARD ST./ SAN FRANCISCO CA 94109/ (415) 928-8086
94114 LAURA L. KING/ 330 EUREKA STREET/ SAN FRANCISCO CA 94114/ (415) 285-9804
94122 DANIEL CARROLL/ 1709 17TH AVE./ SAN FRANCISCO CA 94122
94131 JOHN PEMBERTON/ 3955 ARMY STREET/ SAN FRANCISCO CA 94131/ (415) 282-1387
94132 MARK SCOTT JOHNSON/ DEPT. OF MATHEMATICS/ SAN FRANCISCO STATE UNIV./ 1600 HOLLOWAY AVE./ SAN FRANCISCO CA 94132/ (415) 469-1104
94133 MARCUS L. BYRUCK/ 448 VALLEJO ST./ SAN FRANCISCO CA 94133/ (415) 956-6272
94301 COLIN MCMASTER/ 202 RAMONA STREET #C/ PALO ALTO CA 94301
94301 ATTN: JEANNE L. TOULOUSE - LIBRARIAN/ 02-558/ AMDAHL CORP./ 1250 EAST ARQUES AVENUE/ SUNNYVALE CA 94301/ (408) 746-6654
94303 MICHAEL H. GROSS/ D-317/ VARIAN ASSOCIATES/ 611 HANSEN WAY/ PALO ALTO CA 94303/ (415) 493-4000 X3568
94303 KIM R. HARRIS/ 1055 OREGON AVE./ PALO ALTO CA 94303/ (415) 324-1069
94303 HANK S. MAGNUSKI/ GAMMA TECHNOLOGY INC./ 2452 EMBARCADERO WAY/ PALO ALTO CA 94303/ (415) 856-7421
94303 JOSEPH C. SHARP/ K122/ VARIAN CORPORATE RESEARCH/ 611 HANSEN WAY/ PALO ALTO CA 94303/ (415) 493-4000 X4145
94304 J. P. MARKS/ TELESENSORY SYSTEMS INC./ P.O. BOX 10099/ PALO ALTO CA 94304/ (415) 493-2626
94304 R. K. SUMMIT/ PALO ALTO RESEARCH LAB/ D/5208 B/201/ LOCKHEED/ 3251 HANOVER STREET/ PALO ALTO CA 94304
94304 LEN WEISBERG/ SYSTEMS PROGRAMMING/ BLDG 3L/ HEWLETT-PACKARD CO/ 1501 PAGE MILL RD/ PALO ALTO CA 94304/ (415) 856-2495
94305 ATTN: LIBRARY / SERIALS/ BIN 82/ STANFORD LINEAR ACCELERATOR CENTER/ P.O. BOX 4349/ STANFORD CA 94305
94305 JOHN HENNESSY/ COMPUTER SYSTEMS LAB./ STANFORD UNIV./ STANFORD CA 94305/ (415) 497-1835
94305 M. SHAHID MUJTABA/ ARTIFICIAL INTELLIGENCE LAB/ STANFORD UNIV./ STANFORD CA 94305/ (415) 325-6359
94306 ROY HARRINGTON/ 450 OLIVE AVE/ PALO ALTO CA 94306/ (415) 328-2709/ (415) 964-7400 X43 (WORK)
94510 STANLEY J. HUBER/ 318 STEVEN CT./ BENICIA CA 94510/ (707) 745-8089
94536 CLEVE HART/ 546 ALTURA PL/ FREMONT CA 94536/ (415) 792-2516
94545 DICK VAN LEER/ 22634 FOOTHILL BLVD./ HAYWARD CA 94545/ (408) 371-6057
94550 ATTN: LIBRARY L-53 (COPY B)/ LAWRENCE LIVERMORE LIBRARY/ P.O. BOX 5500/ LIVERMORE CA 94550/ (415) 447-1100
94596 GENE POWERS/ VIRTUAL SYSTEMS INC./ 1500 NEWELL AVE SUITE #406/ WALNUT CREEK CA 94596/ (415) 935-4944
94598 DAVE WALLACE/ CHROMATOGRAPHY DATA SYSTEMS/ 2700 MITCHELL DR./ WALNUT CREEK CA 94598/ (415) 939-2400
94608 DAVID BATES/ 4 CAPTAIN DRIVE #301/ EMERYVILLE CA 94608/ (415) 658-2422
94609 PETER E. DOLEMAN/ 6515 TELEGRAPH AVE. #22/ OAKLAND CA 94609/ (415) 654-1949
94611 PHILIP F. MEADS JR./ 7053 SHIRLEY DRIVE/ OAKLAND CA 94611/ (415) 531-8172
94611 DENNIS NEWTON/ 1 KELTON CT. APT 7-G/ OAKLAND CA 94611/ (415) 655-1057
94703 ERIC MARTINOT/ 2206B JEFFERSON/ BERKELEY CA 94703/ (415) 849-2663
94704 JOSEPH FALETTI/ 1945 BERKELEY WAY #220/ BERKELEY CA 94704/ (415) 548-1192
94705 PETE GOODEVE/ 3012 DEAKIN ST #D/ BERKELEY CA 94705/ (415) 642-6440
94707 WALT FRENCH/ 820 ARLINGTON #1621/ BERKELEY CA 94707/ (415) 788-5454 DAYS/ (415) 526-3551
94707 DANA WHEELER/ 1858 TACOMA AVENUE/ BERKELEY CA 94707/ (415) 869-4646
94708 BLAND EWING/ 221 LAKE DRIVE/ KENSINGTON CA 94708/ (415) 525-5888
94720 LAWRENCE A. ROWE/ DEPT. OF EE AND CS - TEOI/ EVANS HALL/ U OF CALIFORNIA/ BERKELEY CA 94720/ (415) 642-5117
94903 JOHN C. FRANZINI/ 65 MERIAM DR./ SAN RAFAEL CA 94903
94903 BILL STACKHOUSE/ 436 MILLER CREEK ROAD/ SAN RAFAEL CA 94903
94941 ATTN: AYERS LOCKSMITHING/ 227 SHORELINE HWY./ MILL VALLEY CA 94941/ (415) 383-1415
94941 ALEXANDER YUILL-THORNTON II/ P.O. BOX 182/ MILL VALLEY CA 94941/ (415) 383-7806
94960 JUNE B. MOORE/ 32 SALINAS AVE/ SAN ANSELMO CA 94960/ (415) 472-3100 X236/ (415) 456-5889
95008 TIM BLUM/ 768 INWOOD DRIVE/ CAMPBELL CA 95008/ (408) 988-7777 X245
95008 HERBERT H. HOY/ 4868 ROUNDTREE DRIVE/ CAMPBELL CA 95008/ (408) 378-7191
95014 WENDY DUBOIS/ ZILOG CORPORATION/ 10460 BUBB RD./ CUPERTINO CA 95014/ (408) 446-4666
95014 DOUG FORSTER/ 10290 PALO VISTA RD./ CUPERTINO CA 95014
95014 LINDA SIENER/ HEWLETT PACKARD DATA SYSTEMS/ 11000 WOLFE ROAD/ CUPERTINO CA 95014
95014 RICHARD TABOR/ ZILOG/ 10460 BUBB ROAD/ CUPERTINO CA 95014/ (408) 446-4666
95030 KEVIN CONRY/ 23449 SUNSET DRIVE/ LOS GATOS CA 95030/ (408) 353-2748

95030 STEPHEN N. ZILLES/ K52/282/ IBM RESEARCH/ 5600 COTTLE RD/ SAN JOSE CA 95030/ (408) 256-7559
95050 CHRISTINE MORRIS/ GENERAL SYSTEMS DIV./ HEWLETT-PACKARD/ 5303 STEVENS CREEK BLVD./ SANTA CLARA CA 95050/ (408) 249-7020
95051 ATTN: AMI INFORMATION CENTER/ 800 HOMESTEAD ROAD/ SANTA CLARA CA 95051/ (408) 246-0330
95051 JOHN BENITO/ INTEL MAGNETICS/ 3000 OAKMEAD VILLAGE RD./ SANTA CLARA CA 95051/ (408) 987-7700
95051 KAREN CAVILEER/ OMEX/ 2323 OWEN STREET/ SANTA CLARA CA 95051/ (408) 249-5801
95051 AL HARTMANN/ INTEL CORPORATION/ 3065 BOWERS AVENUE/ SANTA CLARA CA 95051/ (408) 987-8080
95051 NIKI JORDAN/ GRANGER ASSOCIATES/ 3101 SCOTT BLVD./ SANTA CLARA CA 95051/ (408) 985-7000
95051 ROBERT S. KIRK/ SOFTWARE DEVELOPMENT SECTION/ 778 BLDG. 700/ AMERICAN MICROSYSTEMS INC./ 3800 HOMESTEAD RD./ SANTA CLARA CA 95051/ (408) 246-0330
95051 DUFF KURLAND/ INFORMATION SYSTEMS DESIGN INC./ 3205 CORONADO DRIVE/ SANTA CLARA CA 95051/ (408) 249-8100
95051 JOHN NAGLE/ 3665 BENTON ST. #60/ SANTA CLARA CA 95051/ (408) 244-6675
95051 CONRAD SCHNEIKER/ MS 690/ NATIONAL SEMICONDUCTOR/ 2900 SEMICONDUCTOR DRIVE/ SANTA CLARA CA 95051/ (408) 737-5067
95051 TAZUYKI TSUNEZUMI/ TERMINAL DIVISION/ FUJITSU LTD/ 2945 OAKMEAD VILLAGE CT./ SANTA CLARA CA 95051/ (408) 727-2670
95051 FRED ZEISE/ DATA SYSTEMS DESIGN/ 3130 CORONADO DRIVE/ SANTA CLARA CA 95051/ (408) 249-9353
95064 ALEC DARA-ABRAMS/ DEPT. OF INFO. SCI./ APPLIED SCIENCES BLDG./ UNIV. OF CALIF. - SANTA CRUZ/ SANTA CRUZ CA 95064/ (408) 429-2565
95070 J. E. DOLL/ 19145 BROOKVIEW DR./ SARATOGA CA 95070
95112 DONALD C. DELONG/ TECHNICAL SERVICES/ INTEL CORP./ 1766 JUNCTION AVE./ SAN JOSE CA 95112/ (408) 987-8080
95118 DAVID A. KOHLER/ 1452 PORTOBELO DR./ SAN JOSE CA 95118/ (408) 395-2160 X211
95123 NORMAN R. BARKER/ 5835 INDIAN AVE./ SAN JOSE CA 95123/ (408) 225-1737
95129 CHOI UISIK/ 6562 IVY LANE/ SAN JOSE CA 95129/ (408) 257-5818
95132 ANDREW HARRIS ZIMMERMAN/ 3422 DUTCHESS COURT/ SAN JOSE CA 95132
95133 RONALD MAK/ 2363 BRUSHGLEN WAY/ SAN JOSE CA 95133/ (408) 259-8205
95193 JACK POWERS/ A50/029/ IBM CORP./ 5600 COTTLE RD./ SAN JOSE CA 95193/ (408) 997-4110
95211 WILLIAM H. FORD/ DEPT. OF MATHEMATICS/ UNIV. OF THE PACIFIC/ STOCKTON CA 95211/ (209) 946-2347
95410 B. C. MACDONALD/ P.O. BOX 69/ ALBION CA 95410/ (707) 937-4352
95442 THOMAS TOLLEFSEN/ 4470 LAKESIDE DR./ GLEN ELLEN CA 95442/ (707) 996-5753
95452 JOE WEISMAN/ 2040 LAWNDALE RD./ KENWOOD CA 95452/ (707) 833-6477
95466 PAUL MEILLEUR/ BOX 365/ PHILO CA 95466
95476 COLEMAN YOUNGDAHL/ 844 OAK LANE/ SONOMA CA 95476/ (707) 938-4643
95540 JIM THOMSON/ BOX 794/ FORTUNA CA 95540/ (707) 725-4817
95662 WILLIAM A. HEITMAN/ 5262 MISSISSIPPI BAR DR./ ORANGEVALE CA 95662/ (916) 988-5262
95817 DAN EBBERTS/ 2006 57TH ST./ SACRAMENTO CA 95817/ (916) 456-4689
95818 GENE GARBUTT/ 2025 28TH ST. #112/ SACRAMENTO CA 95818/ (916) 451-2674
95826 ROBERT RESS/ 9248 VANCOUVER DR./ SACRAMENTO CA 95826/ (916) 362-5712
95926 DAN & ROBIN BARNES/ 279 RIO LINDO AVE. NO. 7/ CHICO CA 95926/ (916) 891-1232
95926 GLENN A. BOOKOUT/ CENTRAL VALLEY MANAGEMENT/ 585 MANZANITA - SUITE 7/ CHICO CA 95926/ (916) 895-8321
96274 DAVID A. ROSSER/ DET 5 - 1ST WEA. WG./ PSC #5 - BOX 10977/ APO CA 96274
96821 SCOTT PLUNKETT/ 1025 KAIMOKU PLACE/ HONOLULU HI 96821
96822 LESLIE M. HINO/ MANAGEMENT SYSTEMS OFFICE/ UNIVERSITY OF HAWAII/ 2425 CAMPUS ROAD - SL RM 10-V/ HONOLULU HI 96822/ (808) 948-8919
96827 GEORGE W. HARVEY/ PANPAC LABS/ P.O. BOX 27785/ HONOLULU HI 96827/ (808) 524-5755
96910 SAM E. RHOADS/ FACULTY OF MATHEMATICS/ UNIV. OF GUAM/ P.O. BOX EK/ AGANA GU 96910
97005 JIM ENGILES/ TECHNICAL INFORMATION CENTER/ INTEL CORPORATION/ 3585 SW 198TH AVE/ ALOHA OR 97005/ (503) 642-6598
97005 JOHN E. RIEBER/ 7780 SW WILSON AVE/ BEAVERTON OR 97005/ (503) 641-5806
97005 DONALD A. ZOCCHI/ 2605 S.W. 203RD AVE./ PORTLAND OR 97005/ (503) 649-9262
97034 C. R. SKUTT/ 1694 FIRCREST/ LAKE OSWEGO OR 97034/ (503) 636-0901
97077 PAT CAUDILL/ MS 92-525/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 645-6464 X1753
97077 GLEN FULLMER/ MS 58/126/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 644-0161 X5833
97077 JUDY GOODMAN/ MS 43-042/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077/ (503) 644-0161 X6091
97077 CHARLIE MONTGOMERY/ MS 58-126/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077
97077 PAULA OCHS/ MS 92-801/ TEKTRONIX INC./ P.O. BOX 500/ BEAVERTON OR 97077
97106 JOHN L. RUTIS/ RT 2 BOX 7H/ BANKS OR 97106
97201 A. C. BROWN/ DEPT. OF PHYSIOLOGY/ SD 414/ UNIV. OF OREGON/ 611 SW CAMPUS DRIVE/ PORTLAND OR 97201/ (503) 225-8958
97201 DAVID ROWLAND/ 734 SW WESTWOOD DR./ PORTLAND OR 97201
97203 ROBERT LUCAS/ 6941 N. OLIN AVENUE/ PORTLAND OR 97203/ (503) 289-3457
97206 SCOTT R. TRAPPE/ 2825 S.E. 68TH/ PORTLAND OR 97206/ (503) 775-9292
97216 MARK M. MILLARD/ 8415 S.E. STEPHENS/ PORTLAND OR 97216/ (503) 253-4545
97223 ALAN ROSENFELD/ FLOATING POINT SYSTEMS INC./ P.O. BOX 23489/ PORTLAND OR 97223/ (503) 641-3151
97225 CHUCK FORSBERG/ R & D/ SIDEREAL CORP./ 9600 SW BARNES RD./ PORTLAND OR 97225/ (503) 227-0111
97225 PAUL HOEFLING/ 8665 S.W. CANYON LANE #22/ PORTLAND OR 97225
97229 JERRY SEWELL JR./ SOFTWARE ENGINEERING/ ELECTRO SCIENTIFIC INDUSTRIES/ 13900 NW SCIENCE PARK DRIVE/ PORTLAND OR 97229/ (503) 641-4141
97301 SHELLEY GILES/ COMPUTER CENTER/ WILLAMETTE UNIV./ 900 STATE STREET/ SALEM OR 97301/ (503) 370-6439
97330 OLE L. ANDERSON/ 4210 NW CRESCENT VALLEY DRIVE/ CORVALLIS OR 97330/ (503) 757-9878
97330 ATTN: COMPUTER SOLUTIONS INC./ 4600 NW SULPHER SPRINGS ROAD/ CORVALLIS OR 97330/ (503) 745-5769
97403 BOB DONAHUE/ FOLLOWTHROUGH/ UNIV. OF OREGON/ EUGENE OR 97403/ (503) 686-3555
97404 ATTN: NORTHWEST MICROCOMPUTER SYSTEMS*/ 749 RIVER AVE./ EUGENE OR 97404/ (503) 688-6874
97405 STEVEN HARTLEY/ 650 W. 27TH AVE./ EUGENE OR 97405/ (503) 344-1809
97701 JOHN & BARBARA HUSEBY/ P.O. BOX 5991/ BEND OR 97701
98007 BOB WALLACE/ MICROSOFT/ 10800 NE 8TH #819/ BELLEVUE WA 98007/ (206) 455-8080
98031 RICHARD W. HERMANSON/ 26625 DOVER CT./ KENT WA 98031
98033 PAUL SAMSON/ TELTONE CORP./ 10801 120TH AVE NE/ KIRKLAND WA 98033/ (206) 827-9626
98055 ROBERT N. ADAMSON/ PACIFIC TECHNOLOGY INC./ 235 AIRPORT WAY/ RENTON WA 98055/ (206) 623-9080
98055 STEPHEN F. MERSHON/ 1151 OLYMPIA AVE. N.E. APT. 21/ RENTON WA 98055/ (206) 226-3891
98055 RICHARD N. TAYLOR/ 17002 159TH PL S.E./ RENTON WA 98055/ (206) 255-5856
98107 JEAN DARSIE/ DEL-D / CMO/ HONEYWELL INC./ 5303 SHILSHOLE AVE. N.W./ SEATTLE WA 98107/ (206) 789-2000
98107 DANIEL EDGAR/ HONEYWELL INC./ 5303 SHILSHOLE AVE NW/ SEATTLE WA 98107/ (206) 789-2000
98115 ATTN: PAT MCCLAIN/ ENGINEERING STUDIES GROUP/ NOAA/ 7600 SAND PT. WAY NE / HANGER 32/ SEATTLE WA 98115
98115 PETER CARTWRIGHT/ 7340 23RD AVE NE/ SEATTLE WA 98115/ (206) 525-2756
98115 DAVID C. JENNER/ 3153 NE 84TH STREET/ SEATTLE WA 98115/ (206) 527-2018
98124 ATTN: KENT TECHNICAL LIBRARY - B/ MS 8K-38/ THE BOEING COMPANY/ P.O. BOX 3707/ SEATTLE WA 98124
98124 ATTN: KENT TECHNICAL LIBRARY - C/ MS 8K - 38/ THE BOEING COMPANY/ P.O. BOX 3707/ SEATTLE WA 98124
98133 RALEIGH ROARK/ METRODATA CORP./ 2150 N. 107TH ST. SUITE 120/ SEATTLE WA 98133/ (206) 367-2100
98133 DWIGHT VANDENBERGHE/ 17541 STONE AVE. N./ SEATTLE WA 98133/ (206) 542-8370
98144 JEAN W. BUTLER/ 714 LAKESIDE S. #207/ SEATTLE WA 98144/ (206) 773-0976
98146 JAMES A. FORGEY/ COMPUTER RENTAL & SERVICE/ 10203 47TH AVE SW #10B/ SEATTLE WA 98146/ (206) 246-9330
98178 CHARLES A. DANIELS/ 10215 62ND AVE. S./ SEATTLE WA 98178/ (206) 723-2525
98195 JOHN C. CHAN/ DEPT. OF COMPUTER SCIENCE/ FR-35/ UNIV. OF WASHINGTON/ SEATTLE WA 98195/ (206) 543-2697
98199 BRADLEY K. GJERDING/ 2806 22ND AVENUE WEST/ SEATTLE WA 98199/ (206) 285-7266
98199 THOMAS J. PALM/ 2529 34TH AVE. W./ SEATTLE WA 98199/ (206) 282-2083
98225 MELVIN DAVIDSON/ COMPUTER CENTER/ 334 BOND HALL/ WESTERN WASHINGTON UNIV./ BELLINGHAM WA 98225
98225 KENDALL STAMBAUGH/ 5009 GUIDE MERIDIAN/ BELLINGHAM WA 98225/ (206) 734-9424
98370 GARY B. STEBBINS/ VIKING TERRACE APTS. #C/ 289 HWY. 3/ POULSBO WA 98370/ (206) 779-4174
98632 RICHARD W. HAMILTON/ P.O. BOX 1609/ LONGVIEW WA 98632
98662 C. T. KROUSE/ 7817 NE 69TH STREET/ VANCOUVER WA 98662
98846 ROBERT E. SANDERSON/ DATASYST/ P.O. BOX 373/ PATEROS WA 98846
99123 MARK STEPHENS/ BOX 57/ ELECTRIC CITY WA 99123/ (509) 633-1360 X491
99163 ALAN DEEHR/ NE 545 KAMIAKEN/ PULLMAN WA 99163/ (509) 332-2225
99163 PAUL J. GILLIAM/ P.O. BOX 2202 CS/ PULLMAN WA 99163/ (509) 335-6611 (WORK)
99163 ROBERT E LORD/ COMPUTING CENTER/ WASHINGTON STATE UNIV./ PULLMAN WA 99163/ (509) 335-6611
99164 ATTN: WASHINGTON STATE UNIV./ 3960 NUCLEAR RADIATION/ PULLMAN WA 99164
99164 J. DENBIGH STARKEY/ COMPUTER SCIENCE DEPT./ WASHINGTON STATE UNIV./ PULLMAN WA 99164/ (509) 335-4254
99164 MASAYUKI TOMIMURO/ OFFICE OF INTERNATIONAL PROGRAMS/ 108 BRYAN HALL/ WASHINGTON STATE UNIV./ PULLMAN WA 99164/ (208) 335-1773
99206 WILLIAM G. HAMMER/ NORTH 107 FARR ROAD/ SPOKANE WA 99206/ (509) 924-9872
99352 ATTN: COLUMBIA MICRO-COMPUTER SYSTEMS*/ P.O. BOX 725/ RICHLAND WA 99352/ (509) 946-4509
99352 TOM MATHIEU/ BATTELLE PACIFIC N.W. LABS/ BATTELLE BOULEVARD/ RICHLAND WA 99352/ (509) 946-3711
99352 ALAN OYAMA/ AZURDATA INC./ P.O. BOX 926/ RICHLAND WA 99352/ (509) 946-1683
99501 DAVID CRAWFORD/ CENTER FOR DISEASE CONTROL/ 225 EAGLE STREET/ ANCHORAGE AK 99501/ (907) 271-4011
99701 TOM HEAD/ DEPT. OF MATH/ UNIV. OF ALASKA/ FAIRBANKS AK 99701
RA-1069 ARGENTINA JORGE LINSKENS/ LOGYCON S.A./ CHACABUCO 380 5P/ BUENOS AIRES RA-1069/ 33-6513
RA-1425 ARGENTINA ADRIAN VILLANUSTRE/ BERUTI 3429 - 14B/ BUENOS AIRES RA-1425
AUSTRALIA GEOFFREY R. GRINTON/ STATE ELECTRICITY COMMISSION OF VIC./ HOWARD STREET/ RICHMOND VICTORIA/ (03) 429 1511
2000 AUSTRALIA D. A. FEIGLIN/ AUSTRALIA SQUARE/ P.O. BOX H143/ SYDNEY N.S.W. 2000
2000 AUSTRALIA MAURICE R. MUNSIE/ NETWORK COMPUTER SCIENCES P/L/ 69 CLARENCE STREET/ SYDNEY N.S.W. 2000/ (02) 290-3677
2001 AUSTRALIA W. J. MATHER/ G.P.O. BOX 3198/ SYDNEY N.S.W. 2001
2006 AUSTRALIA ATTN: BASSER DEPT. OF COMPUTER SCIENCE/ SCHOOL OF PHYSICS/ UNIVERSITY OF SYDNEY/ SYDNEY N.S.W. 2006
2006 AUSTRALIA JURGEN HENRICHS/ DEPT OF COMPUTER SCIENCE/ UNI OF SYDNEY/ SYDNEY N.S.W. 2006
2006 AUSTRALIA IAN ROBERTS/ 403 SAMPLE SURVEY CENTRE/ SYDNEY UNIVERSITY/ SYDNEY N.S.W. 2006
2007 AUSTRALIA ATTN: DIRECTOR/ COMPUTER CENTRE/ NSW INSTITUTE OF TECHNOLOGY/ P.O. BOX 123/ BROADWAY N.S.W. 2007/ (02) 218 9438
2010 AUSTRALIA IAN SHANNON/ 39 STANLEY ST/ DARLINGHURST N.S.W. 2010/ (02) 31 3875
2042 AUSTRALIA RODNEY PARKIN/ 16 WATKIN STREET/ NEWTOWN N.S.W. 2042/ 692-3216
2064 AUSTRALIA BRUCE TAYLOR/ 703/4 BROUGHTON RD/ ARTARMON N.S.W. 2064
2067 AUSTRALIA R. D. GUYON/ IP COMPUTER CONSULTANTS/ 7 RAILWAY STREET/ CHATWOOD N.S.W. 2067/ (02) 411-3522
2070 AUSTRALIA R. A. BROWNELL/ HOUSLEY COMPUTER COMMUNICATIONS PTY L*/ 358 PACIFIC HIGHWAY/ LINDFIELD N.S.W. 2070/ (02) 467 2791
2072 AUSTRALIA CARROLL MORGAN/ ASCOMP PTY LTD/ 870 PACIFIC HWY/ GORDON N.S.W. 2072/ (02) 498-7835
2073 AUSTRALIA W. L. DENISON/ SEPP'L SOFTWARE/ P.O. BOX 199/ PYMBLE N.S.W. 2073
2098 AUSTRALIA PETER BLADWELL/ 78 ROSE AVE./ WHEELER HTS. N.S.W. 2098
2113 AUSTRALIA C.N.S. DAMPNEY/ SCHOOL OF MATHS & PHYSICS/ MACQUARIE UNIVERSITY/ NORTH RYDE N.S.W. 2113
2119 AUSTRALIA ERNST LOOSER/ 21 KARRIL AVE./ BEECROFT N.S.W. 2119
2120 AUSTRALIA DAVID HATCH/ 15 HYLAND AVENUE/ W PENNANT HILL N.S.W. 2120/ 816 2211 (BUS.)/ 871 7845 (HOME)
2232 AUSTRALIA JEFFREY TOBIAS/ APPLIED MATHS AND COMPUTING DIV./ AUST. ATOMIC ENERGY COMM. RES. EST./ PRIVATE MAIL BAG/ SUTHERLAND N.S.W. 2232
531-0111
2500 AUSTRALIA E. H. RIGBY/ RIGBY & ASSOC. PTY. LTD./ 10 REGENT ST./ WOLLONGONG N.S.W. 2500

2580	AUSTRALIA	I. PIRIE/ GOULBURN C.A.E./ MCDERMOTT DRIVE/ GOULBURN N.S.W. 2580
2600	AUSTRALIA	R. BRENT/ COMPUTING RESEARCH GROUP/ AUSTRALIAN NATIONAL UNIVERSITY/ P.O. BOX 4/ CANBERRA A.C.T. 2600
2600	AUSTRALIA	MALCOLM C. NEWEY/ COMPUTER SCIENCE DEPT./ AUSTRALIAN NATIONAL UNIV./ P.O. BOX 4/ CANBERRA A.C.T. 2600/ 81-6376 / 49-4216
2600	AUSTRALIA	G. W. GERRITY/ DEPT OF MATHEMATICS/ UNIV. OF NEW SOUTH WALES/ DUNTROON A.C.T. 2600/ CANBERRA 663526
2600	AUSTRALIA	M. G. SMITH/ COMPUTER CENTRE/ UNI. OF NEW SOUTH WALES/ RMC/ DUNTROON A.C.T. 2600
2600	AUSTRALIA	P. KELLEY/ ADP SECTION/ AUSTRALIAN TAXATION OFFICE/ LANGTON STREET/ PARKES A.C.T. 2600
2601	AUSTRALIA	M. CORBOULD/ BRUCE HALL/ AUSTRALIAN NATIONAL UNIVERSITY/ P.O. BOX 827/ CANBERRA A.C.T. 2601
2601	AUSTRALIA	CHARLES LAYTON/ BRUCE HALL/ A.N.U./ P.O. BOX 827/ CANBERRA A.C.T. 2601
2601	AUSTRALIA	R. J. SHARPE/ 33 WYBALENA GROVE/ COOK A.C.T. 2601
2602	AUSTRALIA	JOHN F. AGNEW/ 37 DUMARESQ STREET/ DICKSON A.C.T. 2602/ (062) 49-7304
2602	AUSTRALIA	D. C. GARRATT/ DARAMALAN COLLEGE/ COWPER STREET/ DICKSON A.C.T. 2602
2607	AUSTRALIA	G. W. GORDON/ P.O. BOX 118/ MAWSON A.C.T. 2607
2902	AUSTRALIA	W. J. CAELLI/ P.O. BOX 1/ KAMBAH A.C.T. 2902
3000	AUSTRALIA	R. ZECTZER/ COMMUNICATION ENGINEERING DEPT./ R.M.I.T./ 124 LA TROBE ST./ MELBOURNE VICTORIA 3000/ (03) 341-2639
3042	AUSTRALIA	T. MOWCHANUK/ P.O. BOX 268/ NIDDRIE VICTORIA 3042
3052	AUSTRALIA	PRABHAKER MATETI/ DEPT. OF COMPUTER SCIENCE/ UNIV. OF MELBOURNE/ PARKVILLE VICTORIA 3052/ (03)341-6459
3052	AUSTRALIA	P. C. POOLE/ DEPT. OF COMPUTER SCIENCE/ UNIV. OF MELBOURNE/ PARKVILLE VICTORIA 3052
3053	AUSTRALIA	B. MCCRAE/ MATHS DEPT/ MELBOURNE STATE COLLEGE/ 757 SWANSTON ST/ CARLTON VICTORIA 3053
3068	AUSTRALIA	M. CULLINAN/ 181 ST. GEORGES ROAD/ NORTH FITZROY VICTORIA 3068
3122	AUSTRALIA	P. L. DEMPSEY/ SWINBURNE COLLEGE OF TECH./ P.O. BOX 218/ HAWTHORN VICTORIA 3122
3127	AUSTRALIA	PETER HORAN/ DEAKIN UNIVERSITY/ P.O. BOX 125/ BELMONT VICTORIA 3127/ (052) 26 3313
3131	AUSTRALIA	ATTN: SHATTOCK & ASSOCIATES/ 79 MAHONEYS ROAD/ FOREST HILL VICTORIA 3131
3131	AUSTRALIA	L. P. WHITEHEAD/ AUSTRALIAN ROAD RESEARCH BOARD/ P.O. BOX 156 (BAG 4)/ NUNAWADING VICTORIA 3131/ 233 1211
3145	AUSTRALIA	JOHN CARPENTER/ 29 WESTGARTH ST/ EAST MALVERN VICTORIA 3145/ 509 4909 (HOME)
3161	AUSTRALIA	ATTN: MINI-COMPUTER SYSTEMS/ FIRST FLOOR/ 105 HAWTHORN ROAD/ N. CAULFIELD VICTORIA 3161
3168	AUSTRALIA	ATTN: PROGRAMMING MANAGER/ SWITCHING AND SIGNALLING BRANCH/ TELECOM AUSTRALIA RESEARCH LABS/ 770 BLACKBURN ROAD CLAYTON VICTORIA 3168/ 03-5416-373
3168	AUSTRALIA	P. COUNTY/ COMP. SCI. DEPT./ MONASH UNIVERSITY/ CLAYTON VICTORIA 3168
3168	AUSTRALIA	W. JACKSON/ ADP CENTRAL ADMINISTRATION/ MONASH UNIVERSITY/ CLAYTON VICTORIA 3168
3168	AUSTRALIA	J. ROSENBERG/ COMP. SCI. DEPT./ MONASH UNIVERSITY/ CLAYTON VICTORIA 3168
3168	AUSTRALIA	P. J. TYERS/ COMPUTER APPLICATION & TECHNIQUES/ TELECOM AUST. RESEARCH LAB./ 770 BLACKBURN ROAD/ CLAYTON VICTORIA 3168
3168	AUSTRALIA	C. S. WALLACE/ DEPT. OF COMP. SCI./ MONASH UNIVERSITY/ CLAYTON VICTORIA 3168
3168	AUSTRALIA	C. BILLINGTON/ CSIRO/ BOX 160/ GLAYTON VICTORIA 3168/ 544-0633
3173	AUSTRALIA	LEONARD SPYKER/ 6 CABARITA COURT/ KEYSBOROUGH VICTORIA 3173
3180	AUSTRALIA	ATTENTION: W. WATTS/ INFORMATION & GRAPHIC SYSTEMS/ 23 PARKHURST DRIVE/ KNOXFIELD VICTORIA 3180
3181	AUSTRALIA	HEATHER A. MACKAY/ 27 THE AVENUE/ WINDSOR VICTORIA 3181
3185	AUSTRALIA	L. BORRETT/ 8/34 ELIZABETH STREET/ ELSTERNWICK VICTORIA 3185
3191	AUSTRALIA	IAN J. CASEY/ 274 BLUFF ROAD/ SANDRINGHAM VICTORIA 3191
4001	AUSTRALIA	ATTN: COMPUTERACC/ P.O. BOX 184/ NORTH BRISBANE QUEENSLAND 4001
4067	AUSTRALIA	D. J. YATES/ BOTANY DEPT./ UNIV. OF QUEENSLAND/ ST. LUCIA QUEENSLAND 4067/ (07) 377-2070
4069	AUSTRALIA	R. J. LONG/ 19 CEDARLEIGH ROAD/ KENMORE QUEENSLAND 4069
4350	AUSTRALIA	F. L. IRVINE/ COMPUTER SERVICES UNIT/ DARLING DOWNS INSTITUTE OF A. E./ P.O. DARLING HEIGHTS/ TOOWOOMBA QUEENSLAND 4350
4700	AUSTRALIA	G. FARR/ DEPTS. OF MATH AND COMPUTER SCIENCE/ MS76/ C.I.A.E./ ROCKHAMPTON QUEENSLAND 4700
5000	AUSTRALIA	ATTN: THE MANAGER/ ADP SERVICES BRANCH/ PUBLIC BUILDINGS DEPT./ 15TH FLOOR S.A.C. VICTORIA SQ./ ADELAIDE S.A. 5000
5001	AUSTRALIA	ATTN: THE DIRECTOR/ WEAPONS RESEARCH ESTABLISHMENT/ BOX 2151 GPO/ ADELAIDE S.A. 5001
5001	AUSTRALIA	J. B. SOUTHCOTT/ DEPT. OF COMP. SCI./ UNIV. OF ADELAIDE/ GPO BOX 498/ ADELAIDE S.A. 5001
5001	AUSTRALIA	KELVIN B. NICOLLE/ DEPT. OF COMPUTING SCIENCE/ UNIV. OF ADELAIDE/ G.P.O. BOX 498/ DELAIDE S.A. 5001/ (08) 223-4333
5064	AUSTRALIA	A. C. BERESFORD/ 46 CROSS ROAD/ MYRTLE BANK S.A. 5064
5109	AUSTRALIA	PETER G. PERRY/ SALISBURY C.A.E./ SMITH ROAD/ SALISBURY S.A. 5109
6000	AUSTRALIA	ATTN: STATE ENERGY COMMISSION/ 365 WELLINGTON STREET/ PERTH W.A. 6000
7000	AUSTRALIA	ATTN: ELIZABETH COMPUTER CENTRE/ 256-274 ELIZABETH STREET/ HOBART TASMANIA 7000
7001	AUSTRALIA	ATTN: EDUCATION DEPT./ G.P.O BOX 169B/ HOBART TASMANIA 7001
7001	AUSTRALIA	ATTN: PROGRAMMERS/ COMPUTING CENTRE/ UNIV. OF TASMANIA/ GPO BOX 252C/ HOBART TASMANIA 7001/ 23 0561 X660
7005	AUSTRALIA	ATTN: INFORMATION SCIENCE CLUB/ DEPT. OF INFO. SCI./ UNIVERSITY OF TASMANIA/ SANDY BAY TASMANIA 7005
7005	AUSTRALIA	JOHN PARRY/ 10 BROADWATERS PDE./ SANDY BAY TASMANIA 7005/ 25 2933
7007	AUSTRALIA	NIGEL WILLIAMS/ 56 RIALANNAH RD/ MOUNT NELSON TASMANIA 7007
7011	AUSTRALIA	A. J. W. HARRISON/ FAIRHAVEN/ AUSTINS FERRY/ HOBART TASMANIA 7011
A-1000	AUSTRIA	MICHAEL ISTINGER/ SCHALTERLAGERND 54/ WIEN A-1000
A-1010	AUSTRIA	WALTER BOLTZ/ DIE ERSTE OSTERREICHISCHE/ SPARKASSE (ABT. INFORMATIK)/ NEUTORGASSE 4/ WIEN A-1010/ 0222/66 16 37/290
A-2340	AUSTRIA	HEINZ STEGBAUER/ HTL/ TECHNIKERSTR. 1-5/ MODLING A-2340
A-4020	AUSTRIA	KARL PRAGERSTORFER/ GES. F. AUT. SYSTEME/ RAINERSTRASSE 23A/4/ LINZ A-4020
B-1050	BELGIUM	PIERRE VAN NYPELSTEER/ UNIVERSITE LIBRE DE BRUXELLES/ AVENUE ROOSEVELT 50-CP181/ BRUXELLES B-1050
B-1160	BELGIUM	ATTN: GERBER SCIENTIFIC EUROPE S.A./ RUE E. STEENO 27/ BRUXELLES B-1160
B-1170	BELGIUM	ALAIN PIROTTE/ MBLE/RESEARCH LABORATORY/ AVENUE VAN BECELAERE 2/ BRUXELLES B-1170/ 673.41.90/ 673.41.99
B-1761	BELGIUM	RONALD J. FARMERY/ KLEISTRAAT 31/ BORCHTLOMBEEK B-1761
B-3030	BELGIUM	P. VERBAETEN/ TOEGEPASTE WISKUNDE EN PROGRAMMATIE/ KATHOLIEKE UNIV. LEUVEN/ CELESTIJNENLAAN 200-A/ HEVERLEE -LEUVEN B-3030
22061	BRAZIL	PIERRE J. LAVELLE/ RUA POMPEU LOUREIRO NO 120 APT 602/ RIO DE JANEIRO COPACABANA 22061/ (021) 236.41.81
22453	BRAZIL	GASTON H. GONNET/ DEPTO DE INFORMATICA P.U.C./ RUA M. DE SAO VICENTE 209/ RIO DE JANEIRO 22453
	CANADA	PETER GROGONO/ 73 ROXTON CRESCENT/ MONTREAL WEST QUEBEC/ (514) 879-4251 (DAY)
	CANADA	STUART LYNNE/ 315A EVERGREEN DR./ PORT MOODY B.C./ (604) 939-2757
B2Y 4A2	CANADA	JACK DODDS/ 341/ BEDFORD INSTITUTE OF OCEANOGRAPHY/ P.O. BOX 1006/ DARTMOUTH N.SCOTIA B2Y 4A2
E3B 5A3	CANADA	D. G. BURNLEY/ COMPTROLLERS - I.U.C./ ROOM 001/ UNIV. OF NEW BRUNSWICK/ FREDERICTON N.B. E3B 5A3
G5L 3A1	CANADA	JEAN BOISVERT/ SERVICE INFORMATIQUE/ UNIVERSITE DU QUEBEC A RIMOUSKI/ 300 URSULINES/ RIMOUSIKI QUEBEC G5L 3A1/ (418) 724-1454
HEH 2J8	CANADA	WERNER FERCH/ 2300 ST. MATHIEU #1401/ MONTREAL QUEBEC HEH 2J8/ (514) 932-0256
H1G 3S5	CANADA	M. MICHEL COURCHESNE/ 1147 VALADE/ MONTREAL QUEBEC H1G 3S5/ (514) 324-5694/ (514) 281-8362
H2Z 1A4	CANADA	MICHEL LOUIS-SEIZE/ HYDRO-QUEBEC/ 75 OUEST DORCHESTER/ MONTREAL QUEBEC H2Z 1A4/ (514) 285-1711 X8827
H3C 3J7	CANADA	PIERRE DESJARDINS/ INFORMATIQUE/ UNIVERSITE DE MONTREAL/ C.P. 6128 SUCC "A"/ MONTREAL QUEBEC H3C 3J7/ (514) 343-7662
H3C 3J7	CANADA	GUY LAPALME/ DEPT. D'INFORMATIQUE/ UNIVERSITE DE MONTREAL/ C.P. 6128 / SUCC "A"/ MONTREAL QUEBEC H3C 3J7/ (514) 343-7382
H3C 3J7	CANADA	LUC LAVOIE/ DEPT. I. R. O./ UNIVERSITE DE MONTREAL/ C.P. 6128 SUCCURSALE A/ MONTREAL QUEBEC H3C 3J7/ (514) 737-3700
H3C 3P8	CANADA	YVES MENARD/ INFORMATIQUE/ UNIVERSITE DU QUEBEC A MONTREAL/ B.P. 8888/ MONTREAL QUEBEC H3C 3P8/ (514) 288-4948
H4T 1N1	CANADA	MARY SUTTON/ A.E.S. DATA LTD./ 570 RUE MCCAFFREY/ MONTREAL QUEBEC H4T 1N1/ (514) 341-5430 X307
H9R 1G1	CANADA	GEORGE MACK/ ENGINEERING DEPT./ CENTRAL DYNAMICS LTD./ 147 HYMUS BLVD./ POINTE CLAIRE QUEBEC H9R 1G1/ (514) 697-0810
H9R 1T9	CANADA	PETER ROWLEY/ 178 BRAEBROOK AVE/ POINTE CLAIRE QUEBEC H9R 1T9/ (514) 697-1898
J8X 1C6	CANADA	R. M. YOUNG/ GAATS 2 PROJECT OFFICE/ ATC SIMULATION CENTRE/ TRANSPORT CANADA/ 45 SACRE COEUR BLVD./ HULL QUEBEC J8X 1C6 (819) 997-3888
J9H 6K2	CANADA	GENE MYLES/ 248 BOURGEAU CR. S./ AYLMER QUEBEC J9H 6K2/ (819) 684-8651
KOA 1AO	CANADA	G. X. AMEY/ WGI CORP./ RR 2/ ALMONTE ONTARIO KOA 1AO/ (613) 256-1338
KOA 2WO	CANADA	ROBERT L. FILLMORE/ R.R. 2/ OSGOODE ONTARIO KOA 2WO/ (613) 821-2216
KOA 3GO	CANADA	KENNETH G. SMITH/ BOX 193 - 115 PINE STREET/ STITTSVILLE ONTARIO KOA 3GO/ (613) 596-5217
K1A OR6	CANADA	DAVID WARD/ DEPT OF EE - COMPUTER GRAPHICS/ BLDG 50/ NATIONAL RESEARCH COUNCIL/ OTTAWA ONTARIO K1A OR6
K1V 6N3	CANADA	W. BRUCE FOULKES/ 2719 NORBERRY CR./ OTTAWA ONTARIO K1V 6N3/ (613) 746-4353
K1V 9J1	CANADA	DAVID J. HARRISON/ HARRISON WILLIAMS AND ASSOCIATES/ 1085 CAHILL DR. W/ OTTAWA ONTARIO K1V 9J1/ (613) 521-6812
K2E 6T7	CANADA	ATTENTION: DONALD LINDSAY/ DYNALOGIC CORPORATION LIMITED/ 141 BENTLEY AVENUE/ OTTAWA ONTARIO K2E 6T7/ (613) 226-1383
K2H 8R6	CANADA	R. T. MOORE/ PRIOR DATA SERVICES/ 16 CREDIT UNION WAY - SUITE 301/ NEPEAN ONTARIO K2H 8R6/ (613) 820-7235
K2H 8S9	CANADA	LUCIEN POTVIN/ CANADIAN MARCONI COMPANY/ 1150 MORRISON DRIVE/ OTTAWA ONTARIO K2H 8S9/ (613) 820-9760
K2K 1N8	CANADA	A. SEWARDS/ 34 SELWYN CRES./ KANATA ONTARIO K2K 1N8/ (613) 592-5512
K2P OG2	CANADA	KEN LEESE/ MOBIUS SOFTWARE LIMITED/ 251 COOPER STREET/ OTTAWA ONTARIO K2P OG2/ (613) 238-4727
K7L 3N6	CANADA	ATTN: REFERENCE ROOM/ COMPUTING AND INF. SCI./ QUEEN'S UNIVERSITY/ KINGSTON ONTARIO K7L 3N6
K7L 3N6	CANADA	ROGER RATHBUN/ COMPUTING CENTRE/ QUEEN'S UNIV./ KINGSTON ONTARIO K7L 3N6/ (613) 547-3273
K7L 3N6	CANADA	R. D. TENNENT/ DEPT. OF COMPUTING AND INFORMATION SC*/ QUEEN'S UNIVERSITY/ KINGSTON ONTARIO K7L 3N6/ (613) 547-2645
K8A 3C5	CANADA	THOMAS MACKENZIE/ 270A ESTHER ST./ PEMBROKE ONTARIO K8A 3C5
L2S 3A1	CANADA	F. R. SKILTON/ COMPUTING CENTRE/ BROCK UNIVERSITY/ ST. CATHERINES ONTARIO L2S 3A1/ (416) 688-2533
L6T 3Y3	CANADA	ROBERT WILSON/ COMSHARE LTD/ 2 INDELL LANE/ BRAMALEA ONTARIO L6T 3Y3/ (416) 791-2525
L7P 1W9	CANADA	H. RISTITS/ ALLIED ELECTRONICS CO./ 2121 FARNAM PLACE/ BURLINGTON ONTARIO L7P 1W9/ (416) 335-2801
L8S 4K1	CANADA	ALLAN LEECHAN/ ACADEMIC COMPUTING SERVICES/ MCMASTER UNIVERSITY/ HAMILTON ONTARIO L8S 4K1/ (416) 525-9140 X4702
L8S 4K1	CANADA	N. SOLNTSEFF/ DEPT. OF APPLIED MATH./ MCMASTER UNIVERSITY/ HAMILTON ONTARIO L8S 4K1/ (416) 525-9140 X4689
M1J 2T1	CANADA	J. W. BAKER/ 73 SHIER DR./ SCARBOROUGH ONTARIO M1J 2T1/ (416) 446-4751
M1R 5A6	CANADA	MIKE WARDALE/ HUNTEC ('70) LTD./ 25 HOWDEN ROAD/ SCARBOROUGH ONTARIO M1R 5A6/ (416) 751-8055
M3A 1M3	CANADA	C. J. WILLIAMS/ INFOPRO LTD/ 9 CLINTWOOD PLACE/ DON MILLS ONTARIO M3A 1M3/ (416) 449-1510
M3C 1H7	CANADA	BRUCE DAVIDSON/ DEPT. 806/ IBM CANADA LABORATORY/ 1150 EGLINTON AVE. EAST/ DON MILLS ONTARIO M3C 1H7/ (416) 443-3162
M3H 5S9	CANADA	PEDRO BARROS/ OVAAC8 INTERNATIONAL INC/ 4800 DUFFERIN ST/ DOWNSVIEW ONTARIO M3H 5S9/ (416) 661-5088/ (416) 661-8869
M4S 1J7	CANADA	CHARLES A. ROYNTON/ 18 MILLWOOD RD APT 3/ TORONTO ONTARIO M4S 1J7/ (416) 487-7091
NON 1JO	CANADA	ROBERT J. ENNS/ BOX 808/ FOREST ONTARIO NON 1JO/ (519) 873-2529
N2C 2EO	CANADA	RON NORMAN/ 109 HARCOURT CR./ KITCHNER ONTARIO N2C 2EO
N2G 4E5	CANADA	PAUL J. MOTZ/ KITCHENER-WATERLOO RECORD/ 225 FAIRWAY ROAD/ KITCHENER ONTARIO N2G 4E5/ (519) 579-2231
N2J 4G5	CANADA	F. A. CELLINI/ NCR CANADA LTD./ 580 WEBSTER ST. N/ WATERLOO ONTARIO N2J 4G5/ (519) 884-1710 X196
N2L 3G1	CANADA	F. D. BOSWELL/ COMPUTER SYSTEMS GROUP/ UNIV. OF WATERLOO/ WATERLOO ONTARIO N2L 3G1/ (519) 885-1211
N2L 3G1	CANADA	K. HARRISON/ DEPT. OF COMPUTING SERVICES/ UNIV. OF WATERLOO/ WATERLOO ONTARIO N2L 3G1/ (519) 885-1211
N2L 3G1	CANADA	JAMES A. SMITH/ DEPT. OF COMPUTER SCIENCE/ UNIV. OF WATERLOO/ WATERLOO ONTARIO N2L 3G1/ (519) 885-1211 X2681
N9B 3P4	CANADA	DAVID R. BROWN/ CANTERBURY COLLEGE/ CRANMER HOUSE/ UNIV. OF WINDSOR/ WINDSOR ONTARIO N9B 3P4/ (519) 256-8866
R3C 1P7	CANADA	NORMAN DIMOCK/ 10 EDMONTON ST. SUITE 409/ WINNIPEG MANITOBA R3C 1P7
R3E OW3	CANADA	RICHARD GORDON/ COMPUTER MEDICINE S108/ UNIV. OF MANITOBA/ 753 MCDERMOT AVE./ WINNIPEG MANITOBA R3E OW3
S4P 2H8	CANADA	DAVID L. COLE/ 2161 SCARTH ST./ REGINA SASK. S4P 2H8/ (306) 565-3949
S4P 2H8	CANADA	ROBERT D. NELL/ RESEARCH & PLANNING/ SASK COMP/ 2161 SCARTH ST./ REGINA SASK. S4P 2H8/ (306) 565-3951
S7N OWO	CANADA	ROBERT N. KAVANAGH/ UNIV. OF SASKATCHEWAN/ SASKATOON SASK. S7N OWO/ (306) 343-2638
T2V OH5	CANADA	CARL RICHARDS/ 403-635 57TH AVE SW/ CALGARY ALBERTA T2V OH5/ (403) 253-4057
T6G 2J8	CANADA	ATTN: LIBRARY/ PERIODICALS SECTION/ UNIVERSITY OF ALBERTA/ EDMONTON ALBERTA T6G 2J8
T6H 3X1	CANADA	BASIL MEDDINGS/ 6508-127TH STREET/ EDMONTON ALBERTA T6H 3X1/ (403) 434-3678
V3C 1S5	CANADA	ATTN: BETA SYSTEMS LTD./ 1760 KINGSWAY AVE./ PORT COQUITLAM B.C. V3C 1S5/ (604) 687-1142
V3N 4N8	CANADA	KIM WILLIAMS/ SUITE 301/ VALLEY SOFTWARE INC./ 7818 6TH ST./ BURNABY B.C. V3N 4N8/ (604) 524-9741

V3T 1Y8 CANADA	BARRY DASHER/ DYNAMIC CONTROL SYSTEMS LTD./ 13662 104A AVENUE SUITE 204/ SURREY B.C. V3T 1Y8/ (604) 585-0655
V5A 1A6 CANADA	DAVE STEVENS/ 1080 SHERLOCK AVE./ BURNABY B.C. V5A 1A6/ (604) 298-9255
V5A 1S6 CANADA	ROGER TOREN/ COMPUTING CENTRE/ SIMON FRASER UNIV./ BURNABY B.C. V5A 1S6/ (604) 291-4632
V6E 1P5 CANADA	DAVID GREER/ 108-1270 BURNABY ST./ VANCOUVER B.C. V6E 1P5/ (604) 688-3993
V6H 1K8 CANADA	DAVID HARRIS/ 1396 W. 11TH AVE. #7/ VANCOUVER B.C. V6H 1K8
V6P 5S2 CANADA	CHARLES THOMPSON/ 7339 W. BOULEVARD/ VANCOUVER B.C. V6P 5S2/ (604) 261-6702
V6T 1W5 CANADA	BRUCE JOLLIFFE/ COMPUTING CENTRE/ UNIV. OF BRITISH COLUMBIA/ VANCOUVER B.C. V6T 1W5/ (604) 228-3938
V6T 1W5 CANADA	VINCENT MANIS/ DEPT. OF COMP. SCI./ UNIV. OF BRITISH COLUMBIA/ VANCOUVER B.C. V6T 1W5/ (604) 228-6537
V7R 4L6 CANADA	D. E. SHAW/ 4371 PATTERDALE DRIVE/ N.VANCOUVER B.C. V7R 4L6/ (604) 988-2181
V8W 2Y2 CANADA	FRANK RUSKEY/ DEPT. OF MATH/ UNIV. OF VICTORIA/ P.O. BOX 1700/ VICTORIA B.C. V8W 2Y2/ (604) 477-6911
95112 CANADA	DAVID FISH/ 1042 CALUMET CT/ SAN JOSE CA 95112/ (408) 297-9334
COLOMBIA	J. C. ARANGO/ EMPAQUES/ APARTADOAEREO 1189/ MEDELLIN/ 770355
DK-1606 DENMARK	JAN HOJLUND NIELSEN/ BC NORDISKBROWNBOVERI A/S/ VESTER FARIMAGSGADE 7/ COPENHAGEN V DK-1606/ 45 1 156210 X374
DK-2000 DENMARK	ROBERT TISCHER/ INFOTEKNIK/ SDR. FASANVEJ 49/ COPENHAGEN F DK-2000
DK-2500 DENMARK	JAN LAUGESEN/ I/S DATACENTRALEN AF 1959/ RETORTVEJ 6/ VALBY DK-2500/ (01) 46 81 22
DK-2800 DENMARK	PER GOEBEL/ RAEVEHOJVEJ 36-1204/ LYNGBY DK-2800
DK-3600 DENMARK	MOHENS GLAD/ STRANDHOJEN 25/ FREDERIKSSUND DK-3600/ 03-313959
DK-8000 DENMARK	ATTN: RECAU COPY 1/ NY MUNKEGADE/ AARHUS C DK-8000/ 06-128355
DK-8000 DENMARK	ATTN: RECAU COPY 2/ NY MUNKEGADE/ AARHUS C. DK-8000/ 06-12 83 55
DK-8200 DENMARK	PEDER NEDEBOL NIELSEN/ LABORATORIET FOR GEOFYSIK/ FINLANDSGADE 6-8/ AARHUS N DK-8200
SF-00250 FINLAND	ATTN: DEPT. OF COMPUTER SCIENCE/ UNIVERSITY OF HELSINKI/ TUKHOLMANKATU 2/ HELSINKI 25 SF-00250
SF-00400 FINLAND	HEIKKI KASKELMA/ SANTAVVORENTIE 1B 18/ HELSINKI SF-00400/ 358 0 644306
SF-20500 FINLAND	MARKKU SUNI/ COMPUTING CENTRE/ UNIVERSITY OF TURKU/ TURKU 50 SF-20500/ 912-335599 X280
SF-33500 FINLAND	JUHANI JAMIA/ ILMARINKATU 42 A 15/ TRE 50 SF-33500
SF-33720 FINLAND	REIJO MIEMINEN/ OPISKELIJANK. 4 E 275/ TRE 72 SF-33720
SF-33900 FINLAND	PERTTI YLINEN/ HARMALANKATU 32 B 5/ TAMPERE 90 SF-33900
F-78150 FRANCE	ATTN: IRIA BIBLIOTHEQUE/ B.P. NO. 205/ LE CHESNAY F-78150
F-78150 FRANCE	ATTN: IRIA / BIBLIOTHEQUE/ BP 105/ LE CHESNAY F-78150
F-91710 FRANCE	R. J. CROUZILLES/ CENTRE DE RECHERCHES DU BOUCHET/ SNPE/ BP NO 2/ VERT-LE-PETIT F-91710
F-92410 FRANCE	GERALD MASPERD/ 5 CHEMIN DES CLOSEAUX/ VILLE D'AURAY F-92410
D-1000 GERMANY	ALBRECHT BIEDL/ INSTITUT FUR SOFTWARE/ DV-GRUNDAUSBILDUNG/ TECHNISCHE UNIVERSITAT BERLIN / VSH 5/ OTTO-SUHR-ALLEE 18/20 BERLIN 10 D-1000/ (030) 314-4891
D-1000 GERMANY	MICHAEL TEPPER/ LIMASTR. 1/ BERLIN 37 D-1000/ (030) 801 21 52
D-1000 GERMANY	LOTHAR HAMMERL/ HORNSTR 12/ BERLIN 61 D-1000/ 030 786 2235
D-2000 GERMANY	MANUEL MALL/ BISMARCKSTR. 44/ HAMBURG 19 D-2000
D-2000 GERMANY	ROLF SCHUMACHER/ SCS GMBH/ OEHLECKERRING 40/ HAMBURG 62 D-2000
D-2800 GERMANY	JAN KEISER/ EISLEBENER STR. 37/ BREMEN 41 D-2800/ 0421 / 538 2834 (WORK)
D-3000 GERMANY	ALMUTH FISCHER/ REGIONALES RECHENZENTRUM/ WUNSTORFER STR. 14/ HANNOVER 91 D-3000
D-4790 GERMANY	MICHAEL KALICINSKY/ BIBLIOTHEK ES/ NIXDORF COMPUTER AG/ FUERSTENALLEE/ PADERBORN D-4790/ 05251-200439
D-4790 GERMANY	ERNST WALTER RASCHNER/ EE2/ ENTWICKLUNG ELEKTRONIK/ FUERSTENALLEE 7/ PADERBORN D-4790
D-5000 GERMANY	DIETRICH KREKEL/ RECHEN ZENTRUM/ UNIVERSITAT ZU KOLN/ ROBERT KOCH STR 10/ KOLN 41 D-5000/ 0221/478/5587
D-5100 GERMANY	W. J. GRODDE/ PDV IM IRT/ RWTH - AACHEN/ SOMMERFELD STR. 54 - BLOCK 54/ AACHEN D-5100/ (0241) 80 7490
D-6236 GERMANY	GERHARD BLANKE/ POSTBOX 5107/ ESCHBORN D-6236/ (06198) 32448
D-6450 GERMANY	VOLKER ASSMUS/ INST. FUER WIRTSCHAFTSINFORMATIK/ TH-FONTANE-STRASSE 20/ HANAU 1 D-6450
D-7000 GERMANY	ATTN: BIBLIOTHEK/ INSTITUT FUER INFORMATIK/ UNIVERSITAET STUTTGART/ AZENBERGSTRASSE 12/ STUTTGART 1 D-7000/ 0711 2078-341
D-7000 GERMANY	WALTER WEHINGER/ LANGUAGES AND PROCESSORS GROUP/ RECHENZENTRUM/ UNIVERSITAT STUTTGART/ PFAFFENWALDRING 64/ STUTTGART 80 D-7000 0711-784 2507
D-7000 GERMANY	GERHARD RECHEL/ RECHENZENTRUM/ UNIVERSITAET STUTTGART/ PFAFFENWALDRING 64/ STUTTGART-80 D-7000
D-7406 GERMANY	ATTN: BIBLIOTHEK/ ZENTRUM FUR DATAGAVERARBEITUNG/ UNIVERSITAT TUBINGEN/ TUBINGEN 1 D-7406
D-7500 GERMANY	LUCIEN FEIEREISEN/ HAID-&-NEU-STR. 16 / W 81/ KARLSRUHE 1 D-7500
D-8000 GERMANY	ATTN: GESELLSHAFT FUER SOFTWARE-ENGIN*/ KARESTR. 60/I/ MUENCHEN 2 D-8000/ (089) 555 234
D-8000 GERMANY	ERWIN ZEDNIK/ ZIEBLANDSTR. 13/ MUENCHEN 40 D-8000/ 289961
D-8000 GERMANY	COLIN CLIFFORD/ NATIONAL SEMICONDUCTOR GMBH/ 808 FUERSTENFELOBRUCK/ MUNCHEN D-8000
D-8000 GERMANY	GEORGE BROOKE/ GAUTINGERSTRASSE 10/ MUNICH 71 D-8000/ (081) 755-3647
D-8000 GERMANY	MANFRED SOMMER/ DEPARTMENT D AP GE/ SIEMENS AG/ OTTOHAHN RING 6/ MUNICH 83 D-8000/ 089-722-61276
D-8012 GERMANY	BERNHARD H. BEITINGER/ INDUSTRIEANLAGEN-BETRIEBSGESELLSCHAFT*/ EINSTEINSTRASSE/ OTTOBRUN D-8012/ 089/60082363
D-8046 GERMANY	ROBERT LATHE/ INSTITUT FUER PLASMAPHYSIK/ GARCHING D-8046/ 089-3299308
D-8520 GERMANY	G. GOERZ/ RRZE/ UNIVERSITAET ERLANGEN-NURNBERG/ MARTENSSTR. 1/ ERLANGEN D-8520/ 09131/85 7410
411 001 INDIA	J. G. KRISHNAYA/ SYSTEMS RESEARCH INSTITUTE/ 6 PARVATI VILLA ROAD/ PUNE 411 001
560 003 INDIA	M. A. SRIDHAR/ 53 GAYATHRI DEVI PARK EXTN./ BANGALORE 560 003/ 31011
560 012 INDIA	SUNDAR RAJARATNAM/ CENTRE FOR THEORETICAL STUDIES/ INDIAN INSTITUTE OF SCIENCE/ BANGALORE 560 012/ 34411 X266 & X268
560 020 INDIA	A. S. BALASUBRAMANYAM/ SUBRAMANYAMSWAMY TEMPLE STREET/ KIMAR PARK WEST - NO. 6 II FLOOR/ BANGALORE 20 BANGALORE 560 020
ISRAEL	RUTH WEINBERG/ COMPUTATION CENTER/ HEBREW UNIVERSITY OF JERUSALEM/ JERUSALEM/ 02-32011/280
49512 ISRAEL	SAM LIBAI/ SDS COMPUTERS LTD./ P.O. BOX 22/ PETACH-TIKVA 49512/ 53054
I-20000 ITALY	GIOVANNI DEGLI ANTONI/ ISTITUTO DI CIBERNETICA/ VIA VIOTTI 5/ MILANO I-20000
I-20010 ITALY	ANTONIO CICU/ HONEYWELL INFORMATION SYSTEMS - ITALY/ PREGNANA MILANESE/ MILANO I-20010/ 02/ 93094 11
I-20133 ITALY	SIANLUIGI CASTELLI/ INSTITUTO DI CIBERNETICA/ VIA VIOTTI 5/ MILANO I-20133
I-50100 ITALY	ATTN: INSTITUTO NAZIONALE DI OTTICA/ FIRENZE I-50100
177 JAPAN	TOSHINORI MAENO/ 1-43 SEKI-MACHI/ NERIMA-KU TOKYO 177/ (03) 726-111 X3298
244 JAPAN	S. TAKAGI/ HIRADO-CHO 1 - TOTSUKA-KU/ YOKOHAMA-SHI KANAGAWA 244
361 JAPAN	TOSHIHIKO FUJIWARA/ NIPPON MINI-COMPUTER CO./ 2165 MOCHIDA/ GYODA CITY SAITAMA 361/ 0485-54-7161
560 JAPAN	NOBUKI TOKURA/ DEPT. OF INFORMATION AND COMPUTER SCI*/ OSAKA UNIVERSITY/ 1-1 MACHIKANEYAMA/ TOKONAKA 560/ 06 (856) 1151 X3245
730 JAPAN	A. E. TADASHI/ FAC. OF ENGINEERING (2 RUI)/ HIROSHIMA UNIVERSITY/ SENDA-MACHI 3-8-2/ HIROSHIMA 730
812 JAPAN	KAZUO USHIJIMA/ DEPT OF COMP. SCI. AND COMM. ENGR./ KYUSHU UNIVERSITY/ 36 HAKOZAKI/ HIGASHI-KU FUKUOKA 812/ 092-641-1101 X3185
04-01 MALAYSIA	LAURIE DAVIES VALLENTINE/ JALAN PARRY/ 10 FLOOR ORIENTAL PLAZA/ KUALA LUMPUR 04-01
MEXICO	JOSE I. KAZA/ DEPARTAMENTO DE SISTEMAS/ UNIV. AUTONOMA METROPOLITANA/ P.O. BOX 16-306/ MEXICO D.F./ 382-5000 X215
MEXICO	ATTN: IIMAS BIBLIOTECA/ UNIVERSIDAD NACIONAL AUTONAMA DE MEXI*/ APDO. POSTAL 20-276/ MEXICO 20 D.F./ (905) 548-5465
MEXICO	JOSE R. CEN ZUBIETA/ UNIDAD DE COMPUTO/ EL COLEGIO DE MEXICO/ CAMINO AL AJUSCO #20/ MEXICO 20 D.F./ 5-68-60-33 X393
NEW ZEALAND	JOHN RAE/ MEDICAL LABORATORY/ P.O. BOX 4120/ AUCKLAND/ 778-339 X49
NEW ZEALAND	ATTN: NEW ZEALAND MICROCOMPUTER CLUB/ C/O SECRETARY/ P.O. BOX 6210/ AUCKLAND 1
NEW ZEALAND	ATTN: PROCESSOR ENTERPRISES LTD./ P.O. BOX 31-261/ AUCKLAND 9
NEW ZEALAND	ATTN: THE DIRECTOR/ COMPUTER CENTRE/ UNIVERSITY OF CANTERBURY/ PRIVATE BAG/ CHRISTCHURCH
NEW ZEALAND	R. B. ALEXANDER/ COMPUTING CENTRE/ UNI. OF OTAGO/ BOX 56/ DUNEDIN
NEW ZEALAND	ATTN: JOHN G. CLEARY/ SYSTEMS & PROGRAMS LTD./ P.O. BOX 30-606/ LOWER HUTT
NEW ZEALAND	W. J. MALTHUS/ 29B HAIG STREET/ LOWER HUTT
NEW ZEALAND	M. H. VERHAART/ 25 CORNWALL STREET/ MASTERTON/ 4805
NEW ZEALAND	ATTN: DOCUMENTATION OFFICER/ COMPUTER CENTRE/ MASSEY UNIVERSITY/ PALMERSTON NORTH
NEW ZEALAND	C. R. BOSWELL/ COMPUTING SERVICES CENTRE/ VICTORIA UNIV. OF WELLINGTON/ PRIVATE BAG/ WELLINGTON/ 721-000 X703
NORWAY	FINN-MOGENS S. HAUGI/ ANATOMISK INSTITUTT/ UNIVERSITY OF OSLO/ KARL JOHANSGT. 47/ OSLO 1
NORWAY	IVAR LABERG/ COMPUTER DEPARTMENT/ UNIVERSITY HOSPITAL OSLO/ RIKSHOSPITALET/ OSLO 1/ (471) 20 10 50
N-2007 NORWAY	EGIL HEISTAD/ NORWEGIAN DEFENCE RESEARCH ESTABLISHM*/ BOX 25/ KJELLER N-2007
N-3290 NORWAY	MORTEN MOEN/ SKYTTERUN 19A/ STAVERN N-3290/ 034-98167
N-3600 NORWAY	COLIN R. BLANCHARD/ KONGSBERG INGENIORHOGSKOLE/ KONGSBERG N-3600/ 732330
N-7034 NORWAY	FRODE SANDVIK/ ELAB/ TRONDHEIM-NTH N-7034/ (047) 75-92669
PERU	GUSTAVO HUNG/ LOS GERANIOS 296/ LIMA 14
00901 POLAND	MICHAL IGLEWSKI/ INSTITUTE OF COMPUTER SCIENCE/ POLISH ACADEMY OF SCIENCES/ P.O. BOX 22/ WARSZAWA PKIN 00901/ 200211 X2225
SOUTH AFRICA	I. A. MOULTRIE/ BOX 68882/ BYRANSTON TRANSVAAL/ 706-4053
SOUTH AFRICA	E. M. EHLERS/ PU FOR CHE/ BOX 536/ POTCHEFSTROOM
0001 SOUTH AFRICA	ATTN: KENTRON (PTY) LTD/ PRIVATE BAG X336/ PRETORIA 0001/ 74 6041
0001 SOUTH AFRICA	C. H. HOOGENCIOORN/ NATIONAL INST FOR AERO AND SYSTEMS TE*/ P.O. BOX 395/ PRETORIA 0001/ 74 9111 X2805
1500 SOUTH AFRICA	ATTENTION: TONY CASTLEMAN/ PRO-DATA (PTY) LTD./ P.O. BOX 150/ BENONI 1500/ 826-5111/2/3
1610 SOUTH AFRICA	A. M. DINKELACKER/ 11 N'GANE DIAZ AVE./ EDENVALE 1610/ 609-5582 (HOME)
2000 SOUTH AFRICA	F. G. BOTHA/ ARTHUR ANDERSON & CO./ P.O. BOX 3652/ JOHANNESBURG 2000/ 21 1381
2000 SOUTH AFRICA	J. A. LEWIS/ BROWN BOVERI S.A. (PTY) LTD/ P.O. BOX 1500/ JOHANNESBURG 2000/ 836-5791
2000 SOUTH AFRICA	G. PEREZ/ P.O. BOX 3714/ JOHANNESBURG 2000/ 28 2600
2001 SOUTH AFRICA	ATTN: PERIODICALS LIBRARY (MATHS)/ UNIV. OF THE WITWATERSRAND/ JOHANNESBURG 2001/ 39-4011
2092 SOUTH AFRICA	ATTN: SOFTWARE MANAGER/ ASSOCIATED HOUSE/ MESSINA ELECTRONIC DEVELOPMENT/ 150 CAROLINE STREET/ BRIXTON 2092
2104 SOUTH AFRICA	EDWARD BRITTAIN/ P.O. BOX 44210/ LINDEN 2104/ (011) 467180
2146 SOUTH AFRICA	B. STRONG/ CONTROL DATA/ P.O. BOX 78105/ SANDTON 2146/ 783-5225
2192 SOUTH AFRICA	JEREMY MCLUCKIE/ 10 HELLESPONT COURT/ 1 BIRT STREET/ SYDENHAM 2192/ 845-1804
2193 SOUTH AFRICA	BRIAN T. STACEY/ B301 COUNTRY LANE/ 71 DORSET ROAD/ PARKWOOD 2193/ (011) 47-2440
2195 SOUTH AFRICA	NEIL SARNAK/ 3 KOMATIE ROAD / EMMARENTIA/ JOHANNESBURG 2195/ 46 8432
4001 SOUTH AFRICA	J. E. RADUE/ COMPUTER SCIENCE DEPT./ UNIV. OF NATAL/ DURBAN 4001/ 352461
4001 SOUTH AFRICA	C. G. URMSON/ 100 SAN LEANDRO/ 80 CURRY ROAD/ DURBAN 4001/ 21-5972
6000 SOUTH AFRICA	PETER WENTWORTH/ COMPUTER SCIENCE DEPT./ THE UNIVERSITY/ PORT ELIZABETH 6000/ 529911
7600 SOUTH AFRICA	REG DODDS/ DEPT. OF COMPUTER SCIENCE/ THE UNIVERSITY/ STELLENBOSCH 7600
S-100 00 SWEDEN	JOHN TIMOTHY FRANKLIN/ KRUKMAKERGATTAN #6/ STOCKHOLM S-100 00
S-100 44 SWEDEN	CARL CRAFOORD/ INST. FOR METALLOGRAFI/ KTH/ FACK/ STOCKHOLM S-100 44/ (08) 787-8350
S-126 11 SWEDEN	ATTENTION: K. I. LARSSON/ MILITARY ELECTRONICS DIVISION/ SATT ELEKTRONIK AKTIEBOLAG/ BOX 32006/ STOCKHOLM S-126 11
S-126 12 SWEDEN	BJORN GIMLE/ CONTROL DATA SWEDEN AB/ BOX 42107/ STOCKHOLM 42 S-126 12/ 08 - 840200
S-163 00 SWEDEN	FOLKE ANDERSSON/ FACK/ SN. RADIO AB/ SPANGA S-163 00/ (08) 752-1474
S-171 21 SWEDEN	ATTN: DATEMA AB/ BOX 1056/ SOLNA S-171 21
S-175 86 SWEDEN	NEIL T. KEANE/ SYSTEM DEVELOPMENT/ DATASAAB/ VEDDESTAVAAGEN 13/ JAARFAALLA S-175 86/ 08/36 28 00
S-195 00 SWEDEN	HANS NORDSTROM/ TINGVALLAVAGEN 7F/ MARSTA S-195 00
S-402 20 SWEDEN	AKE WIKSTROM/ DEPT. OF COMPUTER SCIENCES/ CHALMERS UNIV. OF TECHNOLOGY/ FACK/ GOTEBORG 5 S-402 20
S-411 35 SWEDEN	GUNNAR KARLSSON/ CHALMERSSATAN 27A F-4/ GOTEBORG S-411 35
S-440 74 SWEDEN	LARS Y. SVENSSON/ GNEJSVAGEN 3/ HJALTEBY S-440 74/ 031-671193
S-442 00 SWEDEN	ENGELBERT STORK/ AB DATAKONVERTERING/ TRAKTORGATAN 16/ KUNGALV S-442 00
S-581 83 SWEDEN	ARNE BORTEMARK/ DEPT OF COMPUTER SCIENCE/ LINKOPING UNIVERSITY/ FACK/ LINKOPING S-581 83/ 013/ 11 17 00
S-603 78 SWEDEN	STEN LJUNGKVIST/ AXEL SWARTLINGS GATA 10/ NORRKOPING S-603 78/ 011 - 10 80 00 (OFFICE)/ 011 - 17 02 10 (HOME)

S-751 02	SWEDEN	ATTN: COMPUTING CENTRE/ UPPSALA UNIVERSITY/ BOX 2103/ UPPSALA S-751 02/ 018-111330
S-752 51	SWEDEN	CLAES HOJENBERG/ AGRODATA AB/ GALBO B 53/ UPPSALA S-752 51/ 018-302853
S-981 01	SWEDEN	CHRISTER JUREN/ KIRUNA GEOPHYSICAL INST./ KIRUNA 1 S-981 01/ 0980 12240
CH-1007	SWITZERLAND	CHARLES RAPIN/ CHAIRE INFORMATIQUE APPLIQUEE DMA EPFL/ 61 AVENUE DE COUR/ LAUSANNE CH-1007/ (021) 27 31 05
CH-1204	SWITZERLAND	RAYMOND MOREL/ COLLEGE CALVIN/ 2-4 RUE TH.-DE-BEZE/ GENEVA CH-1204
CH-1211	SWITZERLAND	HERVE TIREFORD/ MOTOROLA INC./ 16 CHEMIN DE LA VOIE-CREUSE/ GENEVA 20 CH-1211/ 33-56-07
CH-1216	SWITZERLAND	MAURICE CALVERT/ IATA/ P.O. BOX 160 / COINTRIN/ GENEVE CH-1216
CH-2000	SWITZERLAND	PIERRE-JEAN ERARD/ CENTRE DE CALCUL UNIVERSITAIRE/ CHANTEMERLE 20/ NEUCHATEL CH-2000
CH-2560	SWITZERLAND	JEAN LOUIS DECOSTER/ LYSS. STR. 21/ NICLAU CH-2560
CH-8092	SWITZERLAND	SVEND ERIK KNUDSEN/ INSTITUT FUER INFORMATIK/ ETH - ZENTRUM/ ZUERICH CH-8092/ (01) 32 62 11 X2217
CH-8304	SWITZERLAND	RAFAEL E. EGLOFF/ SPITZACKERSTRASSE 2/ WALLISELLEN CH-8304
	THE NETHERLANDS	S. D. SWIERSTRA/ TECHNISCHE HOGESCHOOL TWENTE/ P.O. BOX 217/ ENSCHEDE/ 31-53-894441
	THE NETHERLANDS	G. J. STAALMAN/ C/O COMPUTING CENTRE/ WAGENINGEN UNIV./ HOLLANDSEWEG 1 WAGENINGEN
	THE NETHERLANDS	ATTN: LIBRARY/ CONTROL DATA B.V./ J. C. VAN MARKENLAAN 5/ RIJSWIJK/ 070-949344
1007 MC	THE NETHERLANDS	ANDREW S. TANENBAUM/ WISKUNDIG SEMINARIUM/ VRIJE UNIVERSITEIT/ POSTBUS 7161/ AMSTERDAM 1007 MC/ 020 548 24 10
1009 AJ	THE NETHERLANDS	ATTN: BIBLIOTHEEK/ INSTITUUT KERNPHYSISCH-ONDERZOEK/ POSTBUS 4395/ AMSTERDAM 1009 AJ/ (020) 930951
1012 VT	THE NETHERLANDS	KWEE TJOE LIONG/ INSTITUUT ATW/ SPUISTR 210/ AMSTERDAM 1012 VT/ (020) 525-3862 OR 3864
1183 AV	THE NETHERLANDS	DICK VAN DEN BURG/ GETSCO/ PROF E. M. MEYERSLAAN 1/ AMSTELVEEN 1183 AV/ 020-473131
2501 BD	THE NETHERLANDS	H.J.J. DE GIER/ PROCESSING AND STATISTICS/ INST. TNO FOR MATHEMATICS INFORMATION/ P.O. BOX 297/ THE HAGUE 2501 BD
2651 VN	THE NETHERLANDS	P. J. VAN DER HOFF/ PIJPERSTRAAT 5/ BERKEL EN RODENRIJS 2651 VN
2804 HS	THE NETHERLANDS	NICO HOLLEBEEK/ GEERTRUIEDE HOEVE 19/ GOUDA 2804 HS
7323 BA	THE NETHERLANDS	ATTN: RIJKS COMPUTERCENTRUM/ FAUSTSTRAAT 1/ APELDOORN 7323 BA
7500 AE	THE NETHERLANDS	C. BRON/ DEPT. OF ELECTRICAL ENGINEERING/ TECHNISCHE HOGESCHOOL TWENTE/ POSTBUS 217/ ENSCHEDE 7500 AE/ (031) 53 894451
9700 AV	THE NETHERLANDS	HARM PAAS/ DEPT. OF SPACE RESEARCH/ UNIV. OF GRONINGEN/ P.O. BOX 800/ GRONINGEN 9700 AV/ 050-116662
	UNITED KINGDOM	W. L. BLUNDELL/ AYLESBURY COLLEGE OF F.E./ OXFORD ROAD/ AYLESBURY BUCKS.
	UNITED KINGDOM	ROBERT NEELY/ 27 CHILTERN ROAD/ HITCHIN HERTS
	UNITED KINGDOM	ROGER I. TURNER/ 13 FIRST CROSS ROAD / TWICKENHAM/ MIDDLESEX ENGLAND/ 01-894 3243
	UNITED KINGDOM	T. BAYUS/ MERTON TECHNICAL COLLEGE/ MORDEN PARK - LONDON ROAD/ MORDEN SURREY
	UNITED KINGDOM	J. R. DOUGLAS/ OXFORD UNIV. COMPUTING SERVICE/ 13 DANBURY ROAD/ OXFORD ENGLAND
	UNITED KINGDOM	RICHARD J.D. KIRKMAN/ DEPT. OF ATMOSPHERIC PHYSICS/ CLARENDON LABORATORY/ PARKS ROAD/ OXFORD ENGLAND
	UNITED KINGDOM	ALAN BLANNIN/ EUROPEAN SOFTWARE ENGINEERING/ DIGITAL EQUIPMENT CO./ FOUNTAIN HOUSE / BUTTS CENTRE/ READING ENGLAND
	UNITED KINGDOM	D. J. ALLERTON/ MARCONI SPACE & DEFENCE SYSTEMS/ WARREN LANE/ STANMORE MIDDLESEX/ 01/ 954-2311 X23
	UNITED KINGDOM	JEREMY KENAGHAN/ R.A.D.C./ INTERNATIONAL COMPUTERS LTD./ FAIRVIEW ROAD/ STEVENAGE HERTS/ STEVENAGE 56111 X252
AL3 6BL	UNITED KINGDOM	RICHARD ROSS-LANGLEY/ 1 FRANCIS AVENUE/ ST ALBANS AL3 6BL
BA2 7AY	UNITED KINGDOM	ATTN: THE INFORMATION OFFICER/ SWURCC/ UNIVERSITY OF BATH/ CLAVERTON DOWN BATH BA2 7AY
BN1 2GS	UNITED KINGDOM	ATTN: COMPUTER CENTRE/ BRIGHTON POLYTECHNIC/ BRIGHTON ENGLAND BN1 2GS
BN1 9PT	UNITED KINGDOM	P. COOKE/ SCHOOL OF ENGR AND APPL. SCIENCE/ UNIV. OF SUSSEX/ BRIGHTON ENGLAND BN1 9PT
BN1 9QH	UNITED KINGDOM	ATTN: COMPUTING CENTRE/ UNIVERSITY OF SUSSEX/ FALMER/ BRIGHTON SUSSEX BN1 9QH
BT36 8LF	UNITED KINGDOM	MAURICE O'FLAHERTY/ 3 RICHMOND PARK EAST / GLENGORMLEY/ NEWTOWNABBEY N. IRELAND BT36 8LF
B15 2TT	UNITED KINGDOM	KATHY LANG/ THE COMPUTER CENTRE/ UNIV. OF BIRMINGHAM/ ELMS ROAD / PO BOX 363/ BIRMINGHAM ENGLAND B15 2TT/ 021-472-1301 X2233
B4 7PB	UNITED KINGDOM	M H ACKROYD/ ELEC ENGR DEPT/ SUMPNER BUILDING/ ASTON UNIVERSITY/ 19 COLESHILL STREET/ BIRMINGHAM ENGLAND B4 7PB/ 021-359-3611 X559
CB2 3QG	UNITED KINGDOM	PETER ROBINSON/ COMPUTER LABORATORY/ CAMBRIDGE UNIVERSITY/ CORN EXCHANGE STREET/ CAMBRIDGE ENGLAND CB2 3QG
CB5 8BA	UNITED KINGDOM	C. A. LANG/ SHAPE DATA LIMITED/ 5 JESUS LANE/ CAMBRIDGE ENGLAND CB5 8BA
CM17 9NA	UNITED KINGDOM	STEVE MOLES/ STANDARD TELECOMMUNICATION LABS./ LONDON ROAD/ HARLOW ESSEX CM17 9NA/ (0279) 29531 X345
EX4 4PU	UNITED KINGDOM	KEITH TIZZARD/ DEPT. OF M.S.O.R./ UNIV. OF EXETER/ STREATHAM COURT/ EXETER ENGLAND EX4 4PU
EX4 4QL	UNITED KINGDOM	J. A. CAMPBELL/ COMPUTER SCIENCE DEPT./ EXETER UNIV./ STOCKER ROAD/ EXETER ENGLAND EX4 4QL
GU14 80H	UNITED KINGDOM	DAVID NEDLAND-SLATER/ 1 BUCKLAND CLOSE/ FARNBOROUGH HANTS GU14 80H/ 0252-43743
GU7 2DP	UNITED KINGDOM	NIGEL STEPHENS/ HODGSONITES/ CHARTERHOUSE/ GODALMING SURREY GU7 2DP/ (04868) 6393
G12 8QQ	UNITED KINGDOM	J. E. JEACOCKE/ DEPT. OF COMPUTING SCIENCE/ THE UNIVERSITY/ GLASGOW SCOTLAND G12 8QQ/ (041) 339-8855 X7458
HU6 7RX	UNITED KINGDOM	D. SPRIDGEON/ COMPUTER CENTRE/ THE UNIVERSITY/ HULL ENGLAND HU6 7RX
IP5 7RE	UNITED KINGDOM	B. CANTWELL/ DEPT R18.1.1/ P.O. RESEARCH STATION/ MARTLESHAM HEATH/ IPSWICH ENGLAND IP5 7RE/ 0473-642 581
KT1 2EE	UNITED KINGDOM	D. E. LAW/ COMPUTER UNIT/ KINGSTON POLYTECHNIC/ PENRHYN ROAD/ KINGSTON-UP-TH ENGLAND KT1 2EE
LA1 4YN	UNITED KINGDOM	BRIAN A. E. MEEKINGS/ DEPT. OF COMPUTER STUDIES/ UNIVERSITY OF LANCASTER/ BAILRIGG/ LANCASTER ENGLAND LA1 4YN/ (0524) 65201
LA1 4YW	UNITED KINGDOM	K. M. JINKS/ COMPUTER SERVICES DEPARTMENT/ UNIV. OF LANCASTER/ LANCASTER ENGLAND LA1 4YW
LE1 7RH	UNITED KINGDOM	H. J. ROWE/ COMPUTER LABORATORY/ LEICESTER UNIVERSITY/ LEICESTER ENGLAND LE1 7RH/ LEIC. 50000
LN12 1NQ	UNITED KINGDOM	S. TAYLOR-REED/ AUTOMATIONS DEPT/ VIKING GAS TERMINAL/ CONOCO LTD/ MABLETHORPE LINCS LN12 1NQ
L69 3BX	UNITED KINGDOM	MARTIN D. BEER/ COMPUTER LABORATORY/ UNIVERSITY OF LIVERPOOL/ P.O. BOX 147/ LIVERPOOL ENGLAND L69 3BX
MK43 0AL	UNITED KINGDOM	C. G. WHITAKER/ CRANFIELD COMPUTER CENTER/ CRANFIELD INSTITUTE OF TECH./ CRANFIELD BEDFORD MK43 0AL
MK7 6AA	UNITED KINGDOM	M. A. BRAMER/ MATHEMATICS FACULTY/ THE OPEN UNIVERSITY/ MILTON KEYNES ENGLAND MK7 6AA
M1 7ED	UNITED KINGDOM	J. TURNBULL/ NATIONAL COMPUTING CENTRE/ OXFORD ROAD/ MANCHESTER ENGLAND M1 7ED/ 061 228 6333
M13 9PL	UNITED KINGDOM	S. S. THAKKAR/ DEPT. OF COMP. SCI./ UNIVERSITY OF MANCHESTER/ OXFORD ROAD/ MANCHESTER ENGLAND M13 9PL
M21 1JF	UNITED KINGDOM	D. L. GRAY/ 2 CHURCHFIELD/ 8 EDGE LANE / CHORLTON-C-HARDY/ MANCHESTER ENGLAND M21 1JF
NE4 8EB	UNITED KINGDOM	K. HALEY/ 24 AXBRIDGE GARDENS / BENWELL/ NEWCASTLE-U-TY ENGLAND NE4 8EB
NG7 2RD	UNITED KINGDOM	A. D. HEYES/ DEPT. OF PSYCHOLOGY/ UNIV. OF NOTTINGHAM/ NOTTINGHAM ENGLAND NG7 2RD
NN7 3LJ	UNITED KINGDOM	SILVIA SUSSMAN/ 5 MANOR WALK / NETHER HEYFORD/ NORTHANTS ENGLAND NN7 3LJ/ WEEDON 40465
NP44 1NX	UNITED KINGDOM	ATTN: W. J. TAYLOR/ GWENT HOUSE/ FERRANTI COMPUTER SYSTEMS LTD./ GWENT SQUARE - CWMBRAN/ GWENT ENGLAND NP44 1NX
NR4 7TJ	UNITED KINGDOM	WENDY MILNE/ SCHOOL OF COMPUTING STUDIES/ UNIV. OF EAST ANGLIA/ NORWICH ENGLAND NR4 7TJ
NW11 8DP	UNITED KINGDOM	ALAIN D. D. WILLIAMS/ 801 FINCHLEY ROAD/ LONDON ENGLAND NW11 8DP
OX2 6PE	UNITED KINGDOM	C. CURRAN/ COMPUTING LABORATORY/ OXFORD UNIVERSITY/ 13 BANBURY ROAD/ OXFORD ENGLAND OX2 6PE
PE19 3LS	UNITED KINGDOM	T. S. MORAN/ 20 MASEFIELD AVENUE / EATON FORD/ ST. NEOTS CAMBS PE19 3LS
PL4 8AA	UNITED KINGDOM	PATRICIA HEATH/ COMPUTER CENTRE/ PLYMOUTH POLYTECHNIC/ DRAKE CIRCUS/ PLYMOUTH ENGLAND PL4 8AA
P09 2PE	UNITED KINGDOM	FRASER G. DINGWALL/ SOUTHLEIGH PARK HOUSE/ PLESSEY ELECTRONICS RESEARCH/ EASTLEIGH ROAD/ HAVANT HANTS P09 2PE
RG6 2AX	UNITED KINGDOM	R. J. LOADER/ COMPUTER SCI. DEPT./ UNIV. OF READING/ WHITEKNIGHTS/ READING ENGLAND RG6 2AX
RG6 2AX	UNITED KINGDOM	J. D. ROBERTS/ DEPT. OF COMPUTER SCIENCE/ UNIV. OF READING/ WHITEKNIGHTS PARK/ READING ENGLAND RG6 2AX
RG6 2BG	UNITED KINGDOM	E. B. AWUAH/ BRIDGES HALL/ WHITEKNIGHTS ROAD/ READING BERKS RG6 2BG
SA2 8PP	UNITED KINGDOM	ATTN: THE SECRETARY/ DEPT. OF COMP. SCI./ UNIVERSITY COLLEGE OF SWANSEA/ SWANSEA SA2 8PP
SE1 7NA	UNITED KINGDOM	H. W. NEWLAND/ IC/32/ SHELL INTERNATIONAL PETROLEUM CO. LTD./ SHELL CENTRE/ LONDON ENGLAND SE1 7NA
SE1 9LU	UNITED KINGDOM	P. F. HEWITT/ DORSET HOUSE/ COMPUTER WEEKLY/ STAMFORD STREET/ LONDON ENGLAND SE1 9LU
SE18 6PF	UNITED KINGDOM	M. J.J. COSTELLO/ COMPUTER CENTRE/ THAMES POLYTECHNIC/ LONDON ENGLAND SE18 6PF
SK11 6SR	UNITED KINGDOM	DAVID BURNS/ SOFTWARE SCIENCES LTD/ LONDON & MANCHESTER HOUSE / PARK STRE*/ MACCLESFIELD ENGLAND SK11 6SR
SK9 3BX	UNITED KINGDOM	J. R. DORE/ 5 NESTON WAY / HANDFORTH/ WILMSLOW ENGLAND SK9 3BX
SL6 1SL	UNITED KINGDOM	D. J. CALVERT/ PARK HOUSE/ COMPUTER ASSOCIATES/ PARK STREET/ MAIDENHEAD BERKS SL6 1SL
SO22 4LD	UNITED KINGDOM	P. B. ORCHARD/ 25 SUNNYDOWN ROAD / OLIVER'S BATTERY/ WINCHESTER HANTS SO22 4LD
SO9 5NH	UNITED KINGDOM	ATTN: 2900 PASCAL PROJECT/ COMPUTING SERVICE/ UNIV. OF SOUTHAMPTON/ SOUTHAMPTON ENGLAND SO9 5NH/ 0703 559 122
SO9 5NH	UNITED KINGDOM	V. L. EVANS/ COMPUTING SERVICE/ UNIV. OF SOUTHAMPTON/ SOUTHAMPTON ENGLAND SO9 5NH/ 0703 559 122
SO9 5NH	UNITED KINGDOM	KEN ROBINSON/ COMPUTER STUDIES GROUP/ UNIV. OF SOUTHAMPTON/ SOUTHAMPTON ENGLAND SO9 5NH/ 0703 559122
ST5 5BG	UNITED KINGDOM	N. WHITE/ DEPT. OF COMPUTER SCIENCE/ UNIV. OF KEELE/ KEELE STAFFS ST5 5BG/ STOKE-ON-TRENT 621111 X410
SW11	UNITED KINGDOM	DENIS LENIHAN/ BATTERSEA LABORATORY/ BRITISH STEEL CORPORATION/ 140 BATTERSEA PARK ROAD/ LONDON ENGLAND SW11/ 01-622-5511 X6
SW7 2AZ	UNITED KINGDOM	STUART J. MCRAE/ DEPT OF COMPUTING & CONTROL/ IMPERIAL COLLEGE/ 180 QUEENSGATE/ LONDON ENGLAND SW7 2AZ/ 01-589-5111 X2733
SW7 2BX	UNITED KINGDOM	DAVID BROWN/ COMPUTER CENTRE/ IMPERIAL COLLEGE/ SOUTH KENSINGTON/ LONDON ENGLAND SW7 2BX
SW7 2BX	UNITED KINGDOM	P. WHITEHEAD/ COMPUTER CENTRE/ IMPERIAL COLLEGE/ SOUTH KENSINGTON/ LONDON ENGLAND SW7 2BX
S10 2TN	UNITED KINGDOM	CHRIS W. MARTIN/ COMPUTING SERVICES/ 2 HOUNSFIELD ROAD/ SHEFFIELD ENGLAND S10 2TN/ (0742) 78555 X263
TS1 3BA	UNITED KINGDOM	ATTN: THE LIBRARY/ TEESIDE POLYTECHNIC/ BOROUGH ROAD - MIDDLESBROUGH/ CLEVELAND ENGLAND TS1 3BA/ 0642-44176
UB8 3PH	UNITED KINGDOM	D. JONES/ COMPUTING UNIT/ BRUNEL UNIVERSITY/ UXBRIDGE UB8 3PH
WC1	UNITED KINGDOM	V. RYBACKI/ CENTRAL COMPUTING SERVICES/ BIRBECK COLLEGE/ MALET STREET/ LONDON ENGLAND WC1
WC1H 0AH	UNITED KINGDOM	ANTHONY B. WELLER/ COMPUTER CENTRE/ UNIVERSITY COLLEGE LONDON/ 19 GORDON STREET/ LONDON ENGLAND WC1H 0AH
WC1H 0PY	UNITED KINGDOM	I. D. GRAHAM/ INSTITUTE OF ARCHAEOLOGY/ 31-34 GORDONSQUARE/ LONDON ENGLAND WC1H 0PY
WC2	UNITED KINGDOM	A. BARKER/ COMPUTER UNIT/ L. S. E./ HOUGHTON ST./ LONDON ENGLAND WC2/ 01-405-7686 X876
WD1 1SA	UNITED KINGDOM	A. J. JONES/ U.C.S.L./ P.O. BOX 110/ WATFORD HERTS WD1 1SA
W14 0ES	UNITED KINGDOM	MIKE BOUDRY/ 42 DEWHURST ROAD/ LONDON ENGLAND W14 0ES/ 01 - 603 0816
YU-41000	YUGOSLAVIA	STJEPAN JARNJAK/ 13 PROLET. BRIG. 247/ ZAGREB YU-41000/ (041) 513-822/767 (OFFICE)
YU-61001	YUGOSLAVIA	ROBERT REINHARDT/ INSTITUT JOZEF STEFAN/ UNIV. V LJUBLJANI/ JAMOVA 39/ LJUBLJANA YU-61001/ 63-261
YU-71000	YUGOSLAVIA	SUAD ALAGIC/ ELEKTROTEHNICKI FAKULTET/ SARAJEVO LUKAVICA YU-71000

JEANE ABITBOUL	94043	
M H ACKROYD	B4 7PB	UNITED KINGDOM
ROBERT N. ADAMSON	98055	
JOHN F. AGNEW	2602	AUSTRALIA
W. H. AKESON	92037	
SUAD ALAGIC	YU-71000	YUGOSLAVIA
R. B. ALEXANDER		NEW ZEALAND
GERALD P. ALLDREDGE	65401	
DAVID M. ALLEN	66102	
TIMOTHY ALLEN	02172	
D. J. ALLERTON		UNITED KINGDOM
GORDON B. ALLEY	78220	
STEPHEN R. ALPERT	01609	
RICHARD ALRUTZ	14580	
G. X. AMEY	K0A 1A0	CANADA
WILLIAM F. AMON III	22030	
B. H. ANDERSON	83401	
OLE L. ANDERSON	97330	
RAY L. ANDERSON	93017	
FOLKE ANDERSSON	S-163 00	SWEDEN
DENNIS S. ANDREWS	94086	

GIOVANNI DEGLI ANTONI	I-20000	ITALY
DELL ANTONIA	78291	
PETER A. APGAR	18042	
R. REMBERT ARANDA	06103	
J. C. ARANGO		COLOMBIA
FLOYD O. ARNTZ	02160	
LARRY ARONSON	10022	
MICHAEL C. ARYA	94086	
VOLKER ASSMUS	D-6450	GERMANY
PETER R. ATHERTON	63166	
LON ATKINS	92714	
ATTENTION: DAN LAPORTE	93017	
ATTENTION: DONALD LINDSAY	K2E 6T7	CANADA
ATTENTION: H. SPAANENBURG	13440	
ATTENTION: K. I. LARSSON	S-126 11	SWEDEN
ATTENTION: TONY CASTLEMAN	1500	SOUTH AFRICA
ATTENTION: W. WATTS	3180	AUSTRALIA
ATTN: AMI INFORMATION CENTER	95051	
ATTN: APPLIED BUSINESS COMPUTER SYSTEMS	20854	
ATTN: ARJUN REDDY - LIBRARIAN	65211	
ATTN: AYERS LOCKSMITHING	94941	

ATTN: BASSER DEPT. OF COMPUTER SCIENCE	2006	AUSTRALIA
ATTN: BETA SYSTEMS LTD.	V3C 1S5	CANADA
ATTN: BIBLIOTHEEK	1009 AJ	THE NETHERLANDS
ATTN: BIBLIOTHEK	D-7406	GERMANY
ATTN: BIBLIOTHEK	D-7000	GERMANY
TN: COLUMBIA MICRO-COMPUTER SYSTEMS INC*	99352	
ATTN: COMMANDANT (G-DOE-3/TP54)	20590	
ATTN: COMPUTER CENTER LIBRARY	43201	
ATTN: COMPUTER CENTRE	BN1 2GS	UNITED KINGDOM
ATTN: COMPUTER REFERENCE CENTER	08854	
ATTN: COMPUTER SOLUTIONS INC.	97330	
ATTN: COMPUTERACC	4001	AUSTRALIA
ATTN: COMPUTING CENTER	80202	
ATTN: COMPUTING CENTRE	BN1 9QH	UNITED KINGDOM
ATTN: COMPUTING CENTRE	S-751 02	SWEDEN
ATTN: COVELL & HARWOOD CONSULTANTS	49007	
ATTN: DATEMA AB	S-171 21	SWEDEN
ATTN: DEPT. OF COMPUTER SCIENCE	SF-00250	FINLAND
ATTN: DEPT. OF MATHEMATICS	92521	
ATTN: DIRECTOR	2007	AUSTRALIA
ATTN: DIRECTOR OF COMPUTER SERVICES	68701	
ATTN: DOCUMENTATION CENTER	64108	

ATTN: DOCUMENTATION OFFICER		NEW ZEALAND
ATTN: DON T. HO	08854	
ATTN: EASTMAN KODAK CO.	14650	
ATTN: EDUCATION DEPT.	7001	AUSTRALIA
ATTN: ELIZABETH COMPUTER CENTRE	7000	AUSTRALIA
ATTN: GENERAL INSTRUMENT CORPORATION	21031	
ATTN: GERBER SCIENTIFIC EUROPE S.A.	B-1160	BELGIUM
ESELLSHAFT FUER SOFTWARE-ENGINEERING MB*	D-8000	GERMANY
ATTN: HAMPTON TECHNICAL CENTER	23666	
ATTN: IIMAS BIBLIOTECA		MEXICO
ATTN: INFORMATION SCIENCE CLUB	7005	AUSTRALIA
ATTN: INSTITUTO NAZIONALE DI OTTICA	I-50100	ITALY
ATTN: INTERMETRICS INC.	77546	
ATTN: IRIA BIBLIOTHEQUE	F-78150	FRANCE
ATTN: IRIA / BIBLIOTHEQUE	F-78150	FRANCE
ATTN: JEANNE L. TOULOUSE - LIBRARIAN	94301	
ATTN: JOHN G. CLEARY		NEW ZEALAND
ATTN: KENT TECHNICAL LIBRARY - B	98124	
ATTN: KENT TECHNICAL LIBRARY - C	98124	
ATTN: KENTRON (PTY) LTD	0001	SOUTH AFRICA
ATTN: K. MICHAEL - LIBRARIAN	90045	
ATTN: LIBRARY	T6G 2J8	CANADA
ATTN: LIBRARY		THE NETHERLANDS
ATTN: LIBRARY	02178	
ATTN: LIBRARY	75235	
ATTN: LIBRARY L-53 (COPY B)	94550	
ATTN: LIBRARY / SERIALS	94305	
LOVELACE CENTER FOR THE HEALTH SCIENCE*	87108	
ATTN: MICROPROCESSOR LABORATORIES INC.	77043	
ATTN: MICROSYSTEMS INC.	91107	
ATTN: MINI-COMPUTER SYSTEMS	3161	AUSTRALIA
NATIONAL CENTER FOR ATMOSPHERIC RESEARC*	80303	
ATTN: NEW ZEALAND MICROCOMPUTER CLUB		NEW ZEALAND
TN: NORTHWEST MICROCOMPUTER SYSTEMS INC*	97404	
ATTN: PASCAL DISTRIBUTION	80302	
ATTN: PAT MCCLAIN	98115	
ATTN: PERIODICALS LIBRARY (MATHS)	2001	SOUTH AFRICA
ATTN: PROCESSOR ENTERPRISES LTD.		NEW ZEALAND
ATTN: PROGRAMMERS	7001	AUSTRALIA
ATTN: PROGRAMMING MANAGER	3168	AUSTRALIA
ATTN: P.S. INC.	58107	
ATTN: RCS DATA SYSTEMS	48239	
ATTN: RECAU COPY 2	DK-8000	DENMARK
ATTN: RECAU COPY 1	DK-8000	DENMARK
ATTN: REFERENCE ROOM	K7L 3N6	CANADA
ATTN: RIJKS COMPUTERCENTRUM	7323 BA	THE NETHERLANDS
ATTN: SERIALS DEPT.	19104	
ATTN: SHATTOCK & ASSOCIATES	3131	AUSTRALIA
ATTN: SOFTWARE MANAGER	2092	SOUTH AFRICA
ATTN: SPCC LIBRARY 24EE	06602	
ATTN: SSRFC LIBRARY	55455	
ATTN: STATE ENERGY COMMISSION	6000	AUSTRALIA
ATTN: S86	20755	
ATTN: TECHNICAL ASSISTANCE	20052	
ATTN: TECHNICAL INFORMATION CENTER	91320	
ATTN: THE DIRECTOR		NEW ZEALAND
ATTN: THE DIRECTOR	5001	AUSTRALIA
ATTN: THE INFORMATION OFFICER	BA2 7AY	UNITED KINGDOM
ATTN: THE LIBRARY	TS1 3BA	UNITED KINGDOM
ATTN: THE MANAGER	5000	AUSTRALIA
ATTN: THE SECRETARY	SA2 8PP	UNITED KINGDOM
ATTN: USER SERVICES GROUP	93106	
ATTN: WASHINGTON STATE UNIV.	99164	
ATTN: W. H. GENTRY	22980	
ATTN: W. J. TAYLOR	NP44 1NX	UNITED KINGDOM
ATTN: 2900 PASCAL PROJECT	SO9 5NH	UNITED KINGDOM
CHUCK AUGUSTINE	15213	
E. B. AWUAH	RG6 2BG	UNITED KINGDOM
VANESSA AXELROD	20003	
JOSEPH AYERS	01908	
OTTO BAADE	55406	
DUANE W. BAILEY	01002	
LAWRENCE W. BAIN JR.	21040	
WILLIAM L. BAIRD	19090	
J. W. BAKER	M1J 2T1	CANADA
THOMAS BAKER	01862	
DAVID S. BAKIN	92680	
LAWRENCE E. BAKST	07044	
A. S. BALASUBRAMANYAM	560 020	INDIA
JON BANGS	10954	
WILLIAM BARABASH	01754	
THOMAS BARBARA	77072	
PAUL BARINA	94019	
A. BARKER	WC2	UNITED KINGDOM
NORMAN R. BARKER	95123	
DAN & ROBIN BARNES	95926	
G. DENNIS BARNES	91107	
LOUIS BARNETT	90274	
JOHN R. BARR	59812	
PAULA BARRETT	80027	
PEDRO BARROS	M3H 5S9	CANADA
ROBERT F. BASHFORD	19713	
ALEX J. BASKIN	90272	
DAVID BATES	94608	
HENRY R. BAUER III	82071	
JONATHAN BAUER	48104	
NEIL R. BAUMAN	19002	
T. BAYUS		UNITED KINGDOM
DWIGHT R. BEAN	92110	
BERT BEANDER	01876	
E. R. BEAUREGARD	17055	
GARY L. BECHTOLD	77074	
THOMAS L. BECK	53126	
MARTIN D. BEER	L69 3BX	UNITED KINGDOM
MICHAEL BEETNER	06484	
MICHAEL BEHAR	06880	
LARRY BEITCH	45219	
BERNHARD H. BEITINGER	D-8012	GERMANY
JOHN BELEW	90049	
KEITH BELLAIRS	56464	
LEE A. BENBROOKS	90403	
ALLEN E. BENDER	20852	
JOHN BENITO	95051	
WILLIAM G. BENTLEY	46755	
RANDOLPH BENTSON	07846	
A. C. BERESFORD	5064	AUSTRALIA
PAUL C. BERGMAN	21793	
THEODORE C. BERGSTROM	90631	
R. BHARATH	49855	
ALBRECHT BIEDL	D-1000	GERMANY
EARL BILLINGSLEY	01002	
C. BILLINGTON	3168	AUSTRALIA
RODNEY BLACK	06810	
PETER BLADWELL	2098	AUSTRALIA
COLIN R. BLANCHARD	N-3600	NORWAY
GERHARD BLANKE	D-6236	GERMANY
ALAN BLANNIN		UNITED KINGDOM
W. G. BLASDEL	20015	

LYNN BLICKENSTAFF	90065	
TIM BLUM	95008	
WILLIAM E. BLUM	94010	
W. L. BLUNDELL		UNITED KINGDOM
JAMES BLYTHE	48103	
JAMES BLYTHE	48105	
JEAN BOISVERT	G5L 3A1	CANADA
LARRY D. BOLES	37076	
EDWARD W. BOLTON	90066	
WALTER BOLTZ	A-1010	AUSTRIA
JEFF BONAR	01003	
KENNETH C. BONINE	92041	
RICHARD J. BONNEAU	01545	
GLENN A. BOOKOUT	95926	
GARY J. BOOS	69341	
L. BORRETT	3185	AUSTRALIA
ARNE BORTEMARK	S-581 83	SWEDEN
C. R. BOSWELL		NEW ZEALAND
DENNIS K. BOSWELL	85012	
F. D. BOSWELL	N2L 3G1	CANADA
F. G. BOTHA	2000	SOUTH AFRICA
MIKE BOUDRY	W14 OES	UNITED KINGDOM
HENRY J. BOWLDEN	15146	
CHRIS BOYLAN	55455	
ROBERT BOYLAN	08873	
DALE BRAINARD	44103	
M. A. BRAMER	MK7 6AA	UNITED KINGDOM
RICHARD C. BRANDT	84112	
DAVID C. BRAUGHT	61701	
R. BRENT	2600	AUSTRALIA
EDWARD BRITTAIN	2104	SOUTH AFRICA
FRANCIS A. BROGEN	78209	
HENRY C. BROM	55303	
C. BRON	7500 AE	THE NETHERLANDS
GEORGE BROOKE	D-8000	GERMANY
JERRY R. BROOKSHIRE	35805	
ALISON A. BROWN	14850	
ALLEN BROWN	12206	
A. C. BROWN	97201	
DALE BROWN	92663	
DAVID BROWN	SW7 2BX	UNITED KINGDOM
DAVID R. BROWN	N9B 3P4	CANADA
REID L. BROWN	01876	
R. A. BROWNELL	2070	AUSTRALIA
STEVE BRUELL	55455	
A. CHARLES BUCKLEY	08540	
WILLIAM E. BULLEY	48105	
FRANK BURGER	92686	
THOMAS M. BURGER	93017	
MIKE BURGHER	74601	
SUE D. BURKLUND	63188	
D. G. BURNLEY	E3B 5A3	CANADA
DAVID BURNS	SK11 6SR	UNITED KINGDOM
DAN BURROWS	55812	
JEAN W. BUTLER	98144	
EDWARD R. BYRNE	60540	
ROBERT L. BYRNE III	78704	
MARCUS L. BYRUCK	94133	
THOMAS E. BYTHER	04469	
GEORGE A. CACIOPPO JR.	11772	
KEVIN CADMUS	43201	
W. J. CAELLI	2902	AUSTRALIA
PHILLIP R. CALDWELL	75229	
ROBERT CALDWELL	92110	
D. J. CALVERT	SL6 1SL	UNITED KINGDOM
MAURICE CALVERT	CH-1216	SWITZERLAND
DAVID B. CAMERON	33620	
HARRY N. CAMPBELL	92686	
J. A. CAMPBELL	EX4 4QL	UNITED KINGDOM
MIKE CANADAY	92683	
B. CANTWELL	IP5 7RE	UNITED KINGDOM
D. CARACAPPA	08540	
JOHN CARPENTER	3145	AUSTRALIA
DANIEL CARROLL	94122	
PETER CARTWRIGHT	98115	
IAN J. CASEY	3191	AUSTRALIA
B. C. CASKEY	87185	
KARL J. CASPER	44115	
SIANLUIGI CASTELLI	I-20133	ITALY
PAT CAUDILL	97077	
KAREN CAVILEER	95051	
F. A. CELLINI	N2J 4G5	CANADA
JOSE R. CEN ZUBIETA		MEXICO
DOUG CHAMBERLIN	02173	
JOHN C. CHAN	98195	
AUSTIN CHANEY	44202	
TAIWAN CHANG	10010	
A. LYMAN CHAPIN	01581	
W. B. CHAPIN	55112	
GEORGE W. CHERRY	22090	
BABAK CHUBAK	53280	
ANTONIO CICU	I-20010	ITALY
COLIN CLIFFORD	D-8000	GERMANY
JOHN D. COATES	12308	
BARRY A. COLE	90291	
DAVID L. COLE	S4P 2H8	CANADA
JAMES A. COLE	11716	
DAVID E. COLGLAZIER	55410	
BETTY A. COLHOUN	20776	
MIKE COLLIGAN	60601	
ROGER A. COLLINS	92024	
KEVIN CONRY	95030	
P. COOKE	BN1 9PT	UNITED KINGDOM
M. CORBOULD	2601	AUSTRALIA
C. R. CORLES	85019	
MOSHA CORNFELD	90068	
DONALD R. COSCIA	11727	
M. J.J. COSTELLO	SE18 6PF	UNITED KINGDOM
P. COUNTY	3168	AUSTRALIA
M. MICHEL COURCHESNE	H1G 3S5	CANADA
BORDEN COVEL II	92037	
CARLIN R. COVEY	55409	
WILLIAM C. COX	90731	
CARL CRAFOORD	S-100 44	SWEDEN
PENNY CRANE	90036	
DAVID CRAWFORD	99501	
THOMAS W. CROSLEY	94087	
R. J. CROUZILLES	F-91710	FRANCE
TERRENCE R. CULLEN	01730	
M. CULLINAN	3068	AUSTRALIA
C. CURRAN	OX2 6PE	UNITED KINGDOM
JAMES A. CURTIS	03053	
JOSEPH CUSACK	08536	
C.N.S. DAMPNEY	2113	AUSTRALIA
ARTHUR W. DANA JR.	94025	
CHARLES A. DANIELS	98178	
ALEC DARA-ABRAMS	95064	
JEAN DARSIE	98107	

BARRY DASHER	V3T 1Y8	CANADA
DANIEL DASSOW	55016	
BRUCE DAVIDSON	M3C 1H7	CANADA
MELVIN DAVIDSON	98225	
CHARLES DAVIS	46628	
LEO R. DAVIS	20770	
RONALD DAWES	75006	
PAUL E. DAWSON	47401	
DAVID J. DE FANTI	02871	
H.J.J. DE GIER	2501 BD	THE NETHERLANDS
JEAN LOUIS DECOSTER	CH-2560	SWITZERLAND
ALAN DEEHR	99163	
EDWARD N. DEKKER III	60137	
DAVE DELAUTER	47401	
ALLAN B. DELFINO	94087	
DONALD C. DELONG	95112	
P. L. DEMPSEY	3122	AUSTRALIA
W. L. DENISON	2073	AUSTRALIA
RICHARD DEROSIER	01730	
PIERRE DESJARDINS	H3C 3J7	CANADA
JOE DEVITA	92663	
LARRY DI LULLO	85253	
J. F. DICKSON	07753	
RICHARD DIEVENDORFF	91724	
NORMAN DIMOCK	R3C 1P7	CANADA
FRASER G. DINGWALL	P09 2PE	UNITED KINGDOM
A. M. DINKELACKER	1610	SOUTH AFRICA
SANDRA DIRKS	90010	
J. SCOTT DIXON	10901	
JACK DODDS	B2Y 4A2	CANADA
REG DODDS	7600	SOUTH AFRICA
PETER E. DOLEMAN	94609	
J. E. DOLL	95070	
BOB DONAHUE	97403	
D. DONAHUE	94043	
M. F. DOORE	90813	
J. R. DORE	SK9 3BX	UNITED KINGDOM
DAN DORROUGH	14527	
W. S. DORSEY	92667	
FRANK D. DOUGHERTY	61008	
J. R. DOUGLAS		UNITED KINGDOM
R. H. DOUGLAS	85019	
R. H. DOURSON	93407	
GENE DREHER	90266	
KENNETH R. DRIESSEL	74102	
WENDY DUBOIS	95014	
DENNIS S. DUNCAN	87106	
FRANK DUNN	78704	
PEGGY DUNN	20760	
DAN EBBERTS	95817	
DANIEL EDGAR	98107	
PAUL R. EGGERT	90049	
RAFAEL E. EGLOFF	CH-8304	SWITZERLAND
E. M. EHLERS		SOUTH AFRICA
DANIEL A. EHMANN	14619	
DAVID EISENBERG	10006	
DAN L. EISNER	91792	
TOM EKBERG	75006	
MARVIN ELDER	75080	
BILL ELLIOTT	46240	
JAMES C. EMERY	08540	
LARRY ENGELHARDT	48176	
JIM ENGILES	97005	
JOHN ENGLAND	78753	
ROBERT J. ENNS	NON 1J0	CANADA
DONALD L. EPLEY	52242	
ROBERT A. EPPING	19101	
ALAN EPSTEIN	02194	
PIERRE-JEAN ERARD	CH-2000	SWITZERLAND
STANTON D. ERICSON	61107	
DANIEL ETHIER	55101	
V. L. EVANS	SO9 5NH	UNITED KINGDOM
NORMAN M. EVENSEN	10964	
BYRON G. EVERETT	57701	
BLAND EWING	94708	
JOSEPH FALETTI	94704	
SHAWN M. FANNING	92627	
RONALD J. FARMERY	B-1761	BELGIUM
G. FARR	4700	AUSTRALIA
FRANCIS FEDERIGHI	12309	
LUCIEN FEIEREISEN	D-7500	GERMANY
D. A. FEIGLIN	2000	AUSTRALIA
WERNER FERCH	HEH 2J8	CANADA
PAUL D. FIELD	30306	
ROBERT L. FILLMORE	KOA 2W0	CANADA
ALMUTH FISCHER	D-3000	GERMANY
DAVID FISH	95112	CANADA
LANCE K. FISHER	55317	
PAUL F. FITTS	12546	
CHARLES D. FOLEY III	10516	
WILLIAM FOLZ	53715	
WILLIAM H. FORD	95211	
JAMES A. FORGEY	98146	
CHUCK FORSBERG	97225	
DOUG FORSTER	95014	
W. BRUCE FOULKES	K1V 6N3	CANADA
JOHN TIMOTHY FRANKLIN	S-100 00	SWEDEN
K. FRANKOWSKI	55455	
JOHN C. FRANZINI	94903	
JOHN I. FREDERICK	10025	
R. A. FREEDMAN	01842	
DONALD D. FRENCH	02158	
JOHN FRENCH	92691	
WALT FRENCH	94707	
MARIAN FROBISH	61625	
WILLIAM Y. FUJIMOTO	91775	
TOSHIHIKO FUJIWARA	361	JAPAN
GLEN FULLMER	97077	
MARK FURTNEY	24501	
JOSEPH J. GAL	02110	
GENE GARBUTT	95818	
WILLIAM GARD	01776	
KEITH GARLAND	60174	
D. C. GARRATT	2602	AUSTRALIA
PATRICK D. GARVEY	90291	
G. W. GAUGHRAN	60196	
DALE GAUMER	46808	
C. V. GAYLORD	92705	
NARAIN GEHANI	08854	
RICHARD D. GEORGE	60439	
STEPHEN GERKE	22090	
G. W. GERRITY	2600	AUSTRALIA
LARRY GERTZOG	14620	
NICHOLAS R. GETI	06896	
JIM GILBERT	92677	
SHELLEY GILES	97301	
PAUL J. GILLIAM	99163	
MALCOLM GILLIS	35773	

BJORN GIMLE	S-126 12	SWEDEN
BRADLEY K. GJERDING	98199	
MOHENS GLAD	DK-3600	DENMARK
ALOIS GLANC	91330	
PER GOEBEL	DK-2800	DENMARK
G. GOERZ	D-8520	GERMANY
BILLIE S. GOLDSTEIN	11776	
DAVID A. GOMBERG	22102	
JULIAN GOMEZ	91103	
GASTON H. GONNET	22453	BRAZIL
RALPH S. GOODELL	01451	
PETE GOODEVE	94705	
JUDY GOODMAN	97077	
ADOLPH GOODSON	20771	
G. W. GORDON	2607	AUSTRALIA
RICHARD GORDON	R3E 0W3	CANADA
GEORGE S. GORDON JR.	02173	
JOHN R. GOTTHARDT	01775	
ARTHUR W. GOTTMAN	80222	
I. D. GRAHAM	WC1H 0PY	UNITED KINGDOM
RON GRAVES	21044	
DENNIS GRAY	85201	
D. L. GRAY	M21 1JF	UNITED KINGDOM
NORTON GREENFELD	02138	
STEVEN J. GREENFIELD	91604	
DAVID GREER	V6E 1P5	CANADA
DAVID J. GRIEP	90274	
GEOFFREY R. GRINTON		AUSTRALIA
W. J. GRODDE	D-5100	GERMANY
PETER GROGONO		CANADA
MICHAEL H. GROSS	94303	
G. G. GUSTAFSON	92110	
R. D. GUYON	2067	AUSTRALIA
CLAYTON HAAPALA	55057	
PETER H. HAAS	94086	
MICHAEL HADJIOANNOLL	90067	
FRANCIS B. HAJEK	73505	
PAUL H. HALENDA	80303	
DON B. HALES	84147	
K. HALEY	NE4 8EB	UNITED KINGDOM
DONALD HALFORD	80302	
JOHN L. HALL JR.	33601	
STEVEN B. HALL	44107	
ROBERT HALLORAN	07730	
RICHARD W. HAMILTON	98632	
WILLIAM G. HAMMER	99206	
LOTHAR HAMMERL	D-1000	GERMANY
WILLIAM J. HANKLEY	66506	
CHAD HANSEN	55101	
W. J. HANSEN	48104	
JON HANSON	55440	
JAMES HARGREAVES	45214	
JEFF HARLOW	58501	
BRYAN D. HAROLD	66506	
ROY HARRINGTON	94306	
DAVID HARRIS	V6H 1K8	CANADA
KIM R. HARRIS	94303	
K. J. HARRIS	92037	
TERRY HARRIS	02154	
A. J. W. HARRISON	7011	AUSTRALIA
DAVID J. HARRISON	K1V 9J1	CANADA
K. HARRISON	N2L 3G1	CANADA
STEVE HARRISON	92117	
CLEVE HART	94536	
STEVEN HARTLEY	97405	
HAROLD HARTMAN	52240	
AL HARTMANN	95051	
BURT E. HARTMANN	81501	
GEORGE W. HARVEY	96827	
WESTON W. HASKELL	77042	
DAVID HATCH	2120	AUSTRALIA
FINN-MOGENS S. HAUGI		NORWAY
DOUGLAS W. HAWKINS	85202	
MIKE HAYES	30341	
J. NIEL HAYNIE	33334	
TOM HEAD	99701	
LENNY HEATH	27605	
PATRICIA HEATH	PL4 8AA	UNITED KINGDOM
EGIL HEISTAD	N-2007	NORWAY
WILLIAM A. HEITMAN	95662	
PAUL D. HELVICK	75401	
NEAL A. HENDERSON	92128	
JAMES HENDRICKSON	85257	
JOHN HENNESSY	94305	
C. HENNICK	91320	
W. BRYAN HENNINGTON	92708	
JURGEN HENRICHS	2006	AUSTRALIA
L. S. HENSHAW	80401	
R. A. HENZEL	85019	
RICHARD W. HERMANSON	98031	
SCOTT HERR	61832	
MARK HERSEY	48105	
GEORGE C. HETRICK	02169	
P. F. HEWITT	SE1 9LU	UNITED KINGDOM
A. D. HEYES	NG7 2RD	UNITED KINGDOM
BRUCE HIBBARD	06484	
DAVID HICKOK	50158	
CURT HILL	68134	
LESLIE M. HINO	96822	
W. A. HINTON	53211	
ANDY HISGEN	15213	
STEVEN O. HOBBS	01886	
ALAN HOCHBERG	02090	
PAUL HOEFLING	97225	
CLAES HOJENBERG	S-752 51	SWEDEN
JAMES E. HOLBROOK	85281	
PAUL HOLBROOK	92715	
NICO HOLLEBEEK	2804 HS	THE NETHERLANDS
RICHARD HOLMES	01776	
RAY HOLT	94086	
GEORGE HOMER	92680	
MASAHIRO HONDA	94086	
C. H. HOOGENCIOORN	0001	SOUTH AFRICA
DAVID R. HOPPE	60196	
GREGORY L. HOPWOOD	92713	
PETER HORAN	3127	AUSTRALIA
DAVID HORNBAKER	80202	
THOMAS P. HOVEKE	60618	
K. B. HOWARD	93277	
CHARLES P. HOWERTON	80004	
HERBERT H. HOY	95008	
STANLEY J. HUBER	94510	
JAMES M. HUDSON	02181	
STEPHEN P. HUFNAGEL	78712	
MIKE HUGHES	57709	
GUSTAVO HUNG		PERU
LENN S. HUNT	61742	
J. C. HUNTINGTON	85019	
MICHAEL D. HURLEY	22312	
JOHN & BARBARA HUSEBY	97701	
BOB HUTCHINS	92663	
P. L. HUTCHISON	76101	
S. RAY HUTTON	68503	
MICHAL IGLEWSKI	00901	POLAND
ASHOK D. INGLE	75080	
DAVID L. IRVINE	84102	
F. L. IRVINE	4350	AUSTRALIA
R. L. IRWIN	77036	
MICHAEL ISTINGER	A-1000	AUSTRIA
CALVIN W. JACKSON	90025	
CHUCK JACKSON	94035	
CRAIG E. JACKSON	22302	
JOHN R. JACKSON	60542	
W. JACKSON	3168	AUSTRALIA
KENNETH R. JACOBS	20795	
FRED M. JACOBSON	53706	
ROBERT F. JAKOB	53210	
LOUIS B. JAMES	14609	
JUHANI JAMIA	SF-33500	FINLAND
W. JANSSEN	53092	
STJEPAN JARNJAK	YU-41000	YUGOSLAVIA
J. E. JEACOCKE	G12 8QQ	UNITED KINGDOM
RON JEFFRIES	93017	
JOHN P. JENKINSON	75006	
DAVID C. JENNER	98115	
JOHN JENSEN	79409	
AUTHOR R. JETER	85028	
K. M. JINKS	LA1 4YW	UNITED KINGDOM
BARTLEY C. JOHNSON	02116	
JOHN JOHNSON	52240	
MARK SCOTT JOHNSON	94132	
SUE JOHNSON	87545	
VICTOR A. JOHNSON	55404	
ED JOHNSTON	55901	
RICHARD A. JOKIEL	19518	
BRUCE JOLLIFFE	V6T 1W5	CANADA
A. J. JONES	WD1 1SA	UNITED KINGDOM
D. JONES	UB8 3PH	UNITED KINGDOM
RONDALL E. JONES	87185	
JOHN W. JORDAN	01890	
NIKI JORDAN	95051	
EDWARD JUDGE	01060	
CHRISTER JUREN	S-981 01	SWEDEN
MICHAEL KALICINSKY	D-4790	GERMANY
DENIS KALTHOFER	19147	
GILBERT KAPLAN	11415	
RICHARD A. KARHUSE	60201	
GUNNAR KARLSSON	S-411 35	SWEDEN
ROBIN KASCKOW	10017	
NORMAN R. KASHDAN	10029	
HEIKKI KASKELMA	SF-00400	FINLAND
ROBERT KAST	07876	
KOZAI KATSUTOSHI	30332	
ED KATZ	55112	
JEFFREY KATZ	06095	
ROBERT N. KAVANAGH	S7N 0W0	CANADA
JOSE I. KAZA		MEXICO
NEIL T. KEANE	S-175 86	SWEDEN
THORNTON KEEL	78751	
ROY KEELEY JR	36582	
RUSSELL B. KEGLEY	55112	
JAN KEISER	D-2800	GERMANY
P. KELLEY	2600	AUSTRALIA
GUY KELLY	92111	
PAUL L. KELLY	77092	
JANICE ANN KELSO	01532	
JEREMY KENAGHAN		UNITED KINGDOM
JIM KHALAF	92714	
RICHARD KIMBALL	01754	
LAURA L. KING	94114	
ROBERT S. KIRK	95051	
EDWARD E. KIRKHAM	53214	
RICHARD J.D. KIRKMAN		UNITED KINGDOM
KIM A. KIRKPATRICK	87701	
NED J. KISER	46514	
J. B. KLAHN	74004	
MARK KLEIN	03857	
ROGER KLOEPFER	49269	
REX KLOPFENSTEIN JR	43402	
WALTER J. KLOS	21031	
EDWARD W. KNUDSEN	21204	
SVEND ERIK KNUDSEN	CH-8092	SWITZERLAND
JACK KOCHER	61752	
DAVID A. KOHLER	95118	
D. KONIGSBACH	06897	
PETER KOOLISH	94086	
LEOB KOPF	14601	
G. A. KORN	85715	
CARY KORNFELD	94043	
DIETRICH KREKEL	D-5000	GERMANY
G. KREMBS	12401	
J. G. KRISHNAYA	411 001	INDIA
C. T. KROUSE	98662	
RICHARD KUBAT	55414	
GARY A. KUDIS	20024	
DAVID KUHLMAN	92122	
JAMES W. KUIPER	48103	
BENJAMIN KUIPERS	02155	
DUFF KURLAND	95051	
IVAR LABERG		NORWAY
RICHARD D. LADSEN	19422	
JACQUES LAFRANCE	74171	
DAN M. LALIBERTE	55812	
C. A. LANG	CB5 8BA	UNITED KINGDOM
KATHY LANG	B15 2TT	UNITED KINGDOM
LARRY LANGDON	20018	
LAURENCE R. LANGDON	83705	
GUY LAPALME	H3C 3J7	CANADA
CHARLES LARSON	52240	
STEVE LASSMAN	93017	
ROBERT LATHE	D-8046	GERMANY
ARTHUR L. Y. LAU	11973	
JAN LAUGESEN	DK-2500	DENMARK
PIERRE J. LAVELLE	22061	BRAZIL
LUC LAVOIE	H3C 3J7	CANADA
D. E. LAW	KT1 2EE	UNITED KINGDOM
RICHARD J. LAW	11767	
CHARLES LAYTON	2601	AUSTRALIA
PAUL LEBRETON	81212	
ROBERT J. LECHNER	02115	
ROBERT L. LEECH	10996	
ALLAN LEECHAN	L8S 4K1	CANADA
KEN LEESE	K2P 0G2	CANADA
KENNETH O. LELAND	92106	
IAN LEMAIR	85254	
DENIS LENIHAN	SW11	UNITED KINGDOM
R. KENT LEONARD	80222	
FRANK LEPERA	11973	
LANCE A. LEVENTHAL	92067	
J. A. LEVIN	92093	
ROBERT LEVINE	11020	
LEON S. LEVY	08034	
DAVID J. LEWIS	14850	
DON LEWIS	92806	
GEORGE LEWIS	94086	
J. A. LEWIS	2000	SOUTH AFRICA
SAM LIBAI	49512	ISRAEL
ALAN LILLICH	02154	
CARROLL R. LINDHOLM	90403	
BRUCE LINK	87115	
JORGE LINSKENS	RA-1069	ARGENTINA
RICHARD LINTON	53211	
KWEE TJOE LIONG	1012 VT	THE NETHERLANDS
MARTIN LIPELES	91320	
DAVID LIPPINCOTT	48104	
BILL LIPSKY	10013	
STEN LJUNGKVIST	S-603 78	SWEDEN
RICHARD LLEWELLYN	21045	
R. J. LOADER	RG6 2AX	UNITED KINGDOM
JOHN C. LOCKHART	08541	
J. J. LOGAN	22101	
R. J. LONG	4069	AUSTRALIA
LARRY LOOS	63701	
ERNST LOOSER	2119	AUSTRALIA
ROBERT E LORD	99163	
LARRY A. LOTITO	92663	
MICHEL LOUIS-SEIZE	H2Z 1A4	CANADA
TIM LOWERY	90278	
JOHN F. LUBIN	19104	
L. W. LUCAS	93555	
ROBERT LUCAS	97203	
PAUL C. LUSTGARTEN	53706	
RICHARD G. LYMAN	84116	
WILLIAM LYNN	75223	
STUART LYNNE		CANADA
KEN M. MA	06492	
J. P. MACCALLUM	03242	
B. C. MACDONALD	95410	
DAVE MACHART	55901	
GEORGE MACK	H9R 1G1	CANADA
HEATHER A. MACKAY	3181	AUSTRALIA
THOMAS MACKENZIE	K8A 3C5	CANADA
ROY MADDUX	20034	
TOSHINORI MAENO	177	JAPAN
HANK S. MAGNUSKI	94303	
RICHARD L. MAHN	48197	
KEVIN T. MAHONEY	01742	
RONALD MAK	95133	
MANUEL MALL	D-2000	GERMANY
EDWARD S. MALLINAK JR.	44092	
VERNON J. MALLU	77056	
W. J. MALTHUS		NEW ZEALAND
VINCENT MANIS	V6T 1W5	CANADA
MIKE MANTHEY	14226	
RICK L. MARCUS	55455	
J. P. MARKS	94304	
CHRIS W. MARTIN	S10 2TN	UNITED KINGDOM
DAVID P. MARTIN	90045	
PETER D. MARTIN	01450	
GEORGE A. MARTINEZ JR.	90023	
ERIC MARTINOT	94703	
GENE MARTINSON	55440	
GERALD MASPERD	F-92410	FRANCE
JOSEPH W. MAST	22801	
PRABHAKER MATETI	3052	AUSTRALIA
W. J. MATHER	2001	AUSTRALIA
TOM MATHIEU	99352	
DAVID MATTHEWS	48176	
NED N. MAYRATH	74128	
NED N. MAYRATH	74128	
D. W. MCCAMMISH	75080	
B. MCCRAE	3053	AUSTRALIA
GARY MCDONALD	64468	
JACK MCDONNELL	90503	
ROBERT L. MCGHEE	20015	
ALBERT F. MCGIRT	87544	
DANIEL R. MCGLYNN	10549	
STEPHEN S. MCGRANE	55117	
M. L. MCGRAW	30328	
CHARLES W. MCKAY	77058	
MICHAEL MCKENNA	02154	
S. BROOKS MCLANE	16802	
JEREMY MCLUCKIE	2192	SOUTH AFRICA
COLIN MCMASTER	94301	
STUART J. MCRAE	SW7 2AZ	UNITED KINGDOM
PHILIP F. MEADS JR.	94611	
JACK R. MEAGHER	49008	
BASIL MEDDINGS	T6H 3X1	CANADA
BRIAN A. E. MEEKINGS	LA1 4YN	UNITED KINGDOM
PAUL MEILLEUR	95466	
MICHAEL ROBERT MEISSNER	55455	
HAROLD MELAMED	55116	
MONTE JAY MELDMAN	60016	
MONTE J. MELDMAN	60016	
WARREN K. MELHADO	11020	
L. F. MELLINGER	91405	
YVES MENARD	H3C 3P8	CANADA
BERT MENDELSON	01063	
STEPHEN F. MERSHON	98055	
JOHN J. MERTZ	53151	
BOB METZGER	48640	
D. P. METZGER	85019	
KURT MEYLE	19454	
MARK MICHELSON	84115	
KATHLEEN S. MICKEN	23185	
REIJO MIEMINEN	SF-33720	FINLAND
MARK M. MILLARD	97216	
CHARLES E. MILLER	17257	
LESLIE J. MILLER	03051	
MIKE MILLER	66506	
PAUL MILLER	94087	
ROGER E. MILLER	55112	
TERRENCE C. MILLER	92093	
VICTOR S. MILLER	10598	
RICHARD B. MILLWARD	02912	
WENDY MILNE	NR4 7TJ	UNITED KINGDOM
PAUL MINKIN	55426	
S. M. MINTON	33143	
STEVEN L. MITCHELL	10003	
WILLIAM A. MITCHELL	77024	
E. N. MIYA	91103	
V. L. MOBERG	92021	
MORTEN MOEN	N-3290	NORWAY
STEVE MOLES	CM17 9NA	UNITED KINGDOM

JAMES MOLONEY	06902	
FRANK MONACO	30060	
ANNE MONTGOMERY	80230	
CHARLIE MONTGOMERY	97077	
JOE B. MONTGOMERY	62906	
ALLAN MOORE	14215	
H. W. MOORE	92626	
JUNE B. MOORE	94960	
R. T. MOORE	K2H 8R6	CANADA
T. S. MORAN	PE19 3LS	UNITED KINGDOM
RAYMOND MOREL	CH-1204	SWITZERLAND
RAYMOND G. MORETZ JR.	18015	
CARROLL MORGAN	2072	AUSTRALIA
CHRISTINE MORRIS	95050	
GREG MORRIS	01581	
THOMAS M. MORRISETTE	18104	
CHARLES Y. MORROW	15213	
H. R. MORSE	03031	
JOHN A. MORSE	01754	
RICHARD D. MOSAK	14627	
PAUL J. MOTZ	N2G 4E5	CANADA
I. A. MOULTRIE		SOUTH AFRICA
T. MOWCHANUK	3042	AUSTRALIA
ARNOLD H. MUECKE	75235	
ERIK T. MUELLER	02139	
GEORGE H. MUELLER	55435	
M. SHAHID MUJTABA	94305	
GLEN R. J. MULES	10804	
MAURICE R. MUNSIE	2000	AUSTRALIA
GENE MURROW	91367	
LARRY MUSBACH	63045	
BOB MYERS	45429	
GENE MYLES	J9H 6K2	CANADA
PHILIP R. MYLET	22206	
JOHN NAGLE	95051	
GEORGE NAGY	68588	
ROBERT NARAD	13069	
ISAAC R. NASSI	01754	
DAVE NAUMAN	55455	
JOHN NAUMAN	55455	
THOMAS M. NEAL	92634	
DAVID NEDLAND-SLATER	GU14 80H	UNITED KINGDOM
ROBERT NEELY		UNITED KINGDOM
ROBERT D. NELL	S4P 2H8	CANADA
CRAIG NELSON	32901	
BRUCE NERASE	55104	
CHARLES NEUMANN	63045	
MALCOLM C. NEWEY	2600	AUSTRALIA
H. W. NEWLAND	SE1 7NA	UNITED KINGDOM
DENNIS NEWTON	94611	
JAMES NICHOLS	03801	
JEREMY S. NICHOLS	55440	
MARTIN NICHOLS	07801	
DENNIS NICKOLAI	92037	
KELVIN B. NICOLLE	5001	AUSTRALIA
J. F. NIEBLA	90803	
CARL F. NIELSEN	92123	
JAN HOJLUND NIELSEN	DK-1606	DENMARK
PEDER NEDEBOL NIELSEN	DK-8200	DENMARK
JOHN A. NIERENGARTEN	54601	
MARY NOERENBERG	55409	
HANS NORDSTROM	S-195 00	SWEDEN
RON NORMAN	N2C 2E0	CANADA
ROBERT NORRIS	10965	
BILL NORTON	53115	
DICK NORTON	61801	
PAULA OCHS	97077	
MICHAEL OLFE	10028	
ARI OLIVEIRA	91303	
MARK L. OLSON	45701	
P. B. ORCHARD	SO22 4LD	UNITED KINGDOM
BOB ORR	78766	
FARREL OSTLER	84601	
ROBERT M. OTTOSEN	48197	
HUGH OUELLETTE	55987	
WAYNE N. OVERMAN	21202	
JOHN D. OWENS	10304	
ALAN OYAMA	99352	
STEVEN OYANAGI	55455	
JOSEPH A. O'BRIEN	90274	
MARK T. O'BRYAN	49007	
MAURICE O'FLAHERTY	BT36 8LF	UNITED KINGDOM
STEVE O'KEEFE	20229	
JOSEPH O'ROURKE	19104	
HARM PAAS	9700 AV	THE NETHERLANDS
BILL PAGE	01876	
GARRETT PAINE	91011	
THOMAS J. PALM	98199	
JEFF PALMER	67203	
KURT PAPKE	55101	
JEFFRY L. PARKER	94086	
RODNEY PARKIN	2042	AUSTRALIA
KEVIN A. PARKS	21234	
ROSS R. W. PARLETTE	94088	
WALT PARRILL	62025	
JOHN PARRY	7005	AUSTRALIA
J. RICHARD PEARSON	80004	
JOHN PEATMAN	30332	
JOHN PEMBERTON	94131	
RUSSELL J. PEPE	08854	
JIM PERCHIK	02139	
G. PEREZ	2000	SOUTH AFRICA
DAVID PERLMAN	55427	
ARTHUR PERLO	06520	
PETER G. PERRY	5109	AUSTRALIA
MIKE D. PESSONEY	35805	
DAVID PETERSON	01776	
ERVING S. PFAU	70118	
GERALD PFEIFFER	75075	
WILLIAM F. PHILLIPS	92807	
T. L. PHINNEY	85019	
D. T. PIELE	53141	
ROBERT PIERCE	78746	
DOUG PIHL	55440	
I. PIRIE	2580	AUSTRALIA
ALAIN PIROTTE	B-1170	BELGIUM
STEPHEN M. PLATT	19104	
SCOTT PLUNKETT	96821	
JEFF L. POMEROY	55414	
P. C. POOLE	3052	AUSTRALIA
T. D. POPPENDIECK	55104	
LUCIEN POTVIN	K2H 8S9	CANADA
WARREN G. POWELL	19144	
GENE POWERS	94596	
JACK POWERS	95193	
KARL PRAGERSTORFER	A-4020	AUSTRIA
DAVID L. PRESSBERG	01880	
MICHAEL PRIETULA	55455	
CHARLES PRINDLE	10550	
MEL PRUIS	49503	
LEO PUTCHINSKI	75075	
DOUGLAS H. QUEBBEMAN	47150	
E. H. RACHLIN	85019	
J. E. RADUE	4001	SOUTH AFRICA
JUAN RADULOVIC	10016	
JOHN RAE		NEW ZEALAND
RICK RAGER	92714	
SUNDAR RAJARATNAM	560 012	INDIA
ROBERT J. RAKER	94104	
STEVEN R. RAKITIN	07110	
N. RAMACHANDRAN	20036	
THEO RAMAKERS	13502	
JAYASHREE RAMANATHAN	77025	
PETER M. RAMSTAD	55113	
LAURENCE L. RAPER	48076	
CHARLES RAPIN	CH-1007	SWITZERLAND
ERNST WALTER RASCHNER	D-4790	GERMANY
WALTER J. RATAJ	01824	
ROGER RATHBUN	K7L 3N6	CANADA
BRUCE W. RAVENEL	94109	
BRUCE K. RAY	80307	
LINDA LEA RAY		
PAUL MICHAEL REA	92625	
GERHARD RECHEL	D-7000	GERMANY
CHARLES E. REED	06608	
C. EDWARD REID	32308	
ROBERT REINHARDT	YU-61001	YUGOSLAVIA
ROBERT RESS	95826	
CRAIG W. REYNOLDS	94087	
HONOR REYNOLDS	12305	
SAMUEL M. REYNOLDS	91103	
SAM E. RHOADS	96910	
ROBERT L. RHODES	91761	
L. RIANHARD	07960	
LLOYD RICE	90404	
DAN C. RICHARD	67226	
CARL RICHARDS	T2V 0H5	CANADA
GARY A. RICHARDSON	91303	
CHARLES RIDER	91326	
JOHN E. RIEBER	97005	
E. H. RIGBY	2500	AUSTRALIA
DONALD H. RINGLER	20601	
DAVID RIPLEY	08540	
H. RISTITS	L7P 1W9	CANADA
KEN RITCHIE	68005	
C. ROADS	94025	
RALEIGH ROARK	98133	
CARROLL B. ROBBINS JR.	28704	
F. ERIC ROBERTS	06856	
IAN ROBERTS	2006	AUSTRALIA
J. D. ROBERTS	RG6 2AX	UNITED KINGDOM
MARK L. ROBERTS	90274	
KEN ROBINSON	SO9 5NH	UNITED KINGDOM
PETER ROBINSON	CB2 3QG	UNITED KINGDOM
STEVEN ROGERS	45433	
RONALD A. ROHRER	04469	
FRED ROMEO	11725	
MICHAEL ROONEY	02154	
BOB ROOSTH	90245	
ROBERT ROSE	22043	
BRIAN ROSEN	15213	
J. BEN ROSEN	55455	
CAROLYN A. ROSENBERG	90266	
J. ROSENBERG	3168	AUSTRALIA
MICHAEL ROSENBERG	10020	
ALAN ROSENFELD	97223	
DAVID A. ROSSER	96274	
RICHARD ROSS-LANGLEY	AL3 6BL	UNITED KINGDOM
RICHARD L. ROTH	06468	
H. J. ROWE	LE1 7RH	UNITED KINGDOM
LAWRENCE A. ROWE	94720	
DAVID ROWLAND	97201	
STUART W. ROWLAND	44124	
PETER ROWLEY	H9R 1T9	CANADA
CHARLES A. ROYNTON	M4S 1J7	CANADA
OSCAR RTOS	92713	
IRA RUBEN	19002	
LOUIS V. RUFFINO	20854	
FRANK RUSKEY	V8W 2Y2	CANADA
JOHN L. RUTIS	97106	
P. E. RUTTER	07733	
V. RYBACKI	WC1	UNITED KINGDOM
ODD W. RYDEN	01851	
DAVID J. RYPKA	60540	
JOHN RYZLAK	07430	
D. E. SAARELA	55424	
TIM J. SALO	55455	
ARTHUR E. SALWIN	22209	
WILLIAM SAMAYOA	55901	
PAUL SAMSON	98033	
ROBERT E. SANDERSON	98846	
TOM SANDERSON	91311	
TOM SANDERSON	87002	
WAYNE A. SANDERSON	55112	
FRODE SANDVIK	N-7034	NORWAY
GEORGE SARGENT	48804	
NEIL SARNAK	2195	SOUTH AFRICA
JAMES B. SAXE	15213	
TOM SCALLY	11040	
JOSEPH F. SCHAUB JR.	10577	
WERNER SCHENK	14580	
DONALD E. SCHLUTER	90302	
G. MICHAEL SCHNEIDER	55455	
CONRAD SCHNEIKER	95051	
H. JAMES SCHNELKER	80123	
RICHARD SCHROEDEL	55427	
KENT SCHROEDER	55435	
JAY SCHUMACHER	80302	
ROLF SCHUMACHER	D-2000	GERMANY
BEN SCHWARTZ	07821	
FRANK SCHWARTZ	02173	
DAVID T. SCOTT	87112	
WILLIAM H. SEAVER	92626	
DUANE W. SEBEM	55372	
MARK J. SEBERN	53012	
JERRY W. SEGERS	30332	
MARK SEIDEN	10598	
LARRY SEILER	91125	
MARK SENN	47905	
A. SEWARDS	K2K 1N8	CANADA
JERRY SEWELL JR.	97229	
GEORGE M. SHANNON	02173	
IAN SHANNON	2010	AUSTRALIA
JOSEPH C. SHARP	94303	
R. J. SHARPE	2601	AUSTRALIA
D. E. SHAW	V7R 4L6	CANADA
JEFFRY G. SHAW	94088	
JOHN M. SHAW	20014	
ASHOK SHENOLIKAR	11725	
AL SHEPPARD	30313	
THOMAS E. SHIELDS	22304	
P. L. SHIMER-ROWE	93021	
KERRY SHORE	55107	
KEN SIBERZ	90046	
LINDA SIENER	95014	
STEFAN M. SILVERSTON	03060	
BILL SIMMONS	55440	
DENNIS SIMMS	80221	
THOMAS W. SKELTON	48823	
JAMES K. SKILLING	01740	
F. R. SKILTON	L2S 3A1	CANADA
C. R. SKUTT	97034	
LES SLATER	01862	
CAROL SLEDGE	15229	
IRA SLODODIEN	94104	
BARRY SMITH	91107	
BROOKS DAVID SMITH	53211	
DAN SMITH	65211	
JAMES A. SMITH	N2L 3G1	CANADA
JAMES E. SMITH	02178	
KENNETH G. SMITH	KOA 3G0	CANADA
LAWTHER O. SMITH	18936	
M. G. SMITH	2600	AUSTRALIA
RICHARD SNODGRASS	15213	
PAT SNYDER	68025	
REGIS B. SNYDER JR	60164	
JAMES SOLDERITSCH	19085	
N. SOLNTSEFF	L8S 4K1	CANADA
SAMUEL SOLON	94087	
MANFRED SOMMER	D-8000	GERMANY
LEE L. C. SORENSEN	90604	
THOMAS J. SOUCY	01905	
J. B. SOUTHCOTT	5001	AUSTRALIA
JOHN R. SOUVESTRE	70005	
TERRY L. SPEAR	80302	
RICHARD SPELLERBERG	55440	
LUTHER SPERBERG	10010	
JOHN SPIKER	91364	
RICHARD D. SPILLANE	07666	
ROB SPRAY	75240	
D. SPRIDGEON	HU6 7RX	UNITED KINGDOM
ALLEN SPRINGER	02139	
LEONARD SPYKER	3173	AUSTRALIA
M. A. SRIDHAR	560 003	INDIA
G. J. STAALMAN		THE NETHERLANDS
BRIAN T. STACEY	2193	SOUTH AFRICA
BILL STACKHOUSE	94903	
RICHARD STADTMILLER	22091	
KENDALL STAMBAUGH	98225	
J. DENBIGH STARKEY	99164	
MICHAEL K. STAUFFER	94062	
GARY B. STEBBINS	98370	
E. L. STECHMANN	55112	
CHARLES A. STEELE JR.	01854	
GREG STEELE	55435	
HEINZ STEGBAUER	A-2340	AUSTRIA
MARK STEPHENS	99123	
NIGEL STEPHENS	GU7 2DP	UNITED KINGDOM
JACK STEVE	83814	
DAVE STEVENS	V5A 1A6	CANADA
ROBERT K. STEVENS	33432	
MAUREEN J. STILLMAN	02173	
R. D. STINAFF	60004	
A. I. STOCKS	33319	
JERRY STODDARD	55440	
RICHARD A. STONE	55435	
ENGELBERT STORK	S-442 00	SWEDEN
ROBERT STRADER	44139	
ALAN STRELZOFF	02062	
B. STRONG	2146	SOUTH AFRICA
JAMES F. SULLIVAN	92707	
R. K. SUMMIT	94304	
MARKKU SUNI	SF-20500	FINLAND
SILVIA SUSSMAN	NN7 3LJ	UNITED KINGDOM
A. J. SUTTON	27101	
MARY SUTTON	H4T 1N1	CANADA
STANLEY M. SUTTON	77092	
EDGAR N. SVENDSEN	45840	
LARS Y. SVENSSON	S-440 74	SWEDEN
STANLEY M. SWANSON	77843	
E. G. SWARTZMEYER	30303	
S. D. SWIERSTRA		THE NETHERLANDS
RICHARD TABOR	95014	
A. E. TADASHI	730	JAPAN
S. TAKAGI	244	JAPAN
KEN TAKAHASHI	01730	
RAMON TAN	10016	
HIDEHIKO TANAKA	10031	
ANDREW S. TANENBAUM	1007 MC	THE NETHERLANDS
BRADLEY M. TATE	75240	
BRUCE TAYLOR	2064	AUSTRALIA
DAVID K. TAYLOR	55812	
RICHARD N. TAYLOR	98055	
S. TAYLOR-REED	LN12 1NQ	UNITED KINGDOM
F. TEMPEREAU	92127	
R. D. TENNENT	K7L 3N6	CANADA
MICHAEL TEPPER	D-1000	GERMANY
S. S. THAKKAR	M13 9PL	UNITED KINGDOM
PRAKASH THATTE	60164	
RICK THOMAS	20012	
RON THOMAS	55435	
CHARLES THOMPSON	V6P 5S2	CANADA
LADONNA THOMPSON	55424	
PAUL THOMPSON	55441	
JIM THOMSON	95540	
DENNIS K. THORSON	07922	
LAVINE THRAILKILL	40506	
COYT C. TILLMAN JR.	02139	
PATRICIA TIMPANARO	22003	
HERVE TIREFORD	CH-1211	SWITZERLAND
ROBERT TISCHER	DK-2000	DENMARK
KEITH TIZZARD	EX4 4PU	UNITED KINGDOM
JEFFREY TOBIAS	2232	AUSTRALIA
NOBUKI TOKURA	560	JAPAN
THOMAS TOLLEFSEN	95442	
MASAYUKI TOMIMURO	99164	
ROGER TOREN	V5A 1S6	CANADA
SCOTT R. TRAPPE	97206	
MARIUS TROOST	92713	
JAY TROW	78704	
TAZUYKI TSUNEZUMI	95051	
RICHARD L. TUCKER	45324	
JIM TURLEY	80020	
J. TURNBULL	M1 7ED	UNITED KINGDOM

PRESCOTT TURNER	02162	
ROGER I. TURNER		UNITED KINGDOM
ROBERT W. TUTTLE	06520	
P. J. TYERS	3168	AUSTRALIA
GORDON UBER	10591	
CHOI UISIK	95129	
C. G. URMSON	4001	SOUTH AFRICA
TOM URSIN	55440	
KAZUO USHIJIMA	812	JAPAN
LAURIE DAVIES VALLENTINE	04-01	MALAYSIA
DICK VAN DEN BURG	1183 AV	THE NETHERLANDS
P. J. VAN DER HOFF	2651 VN	THE NETHERLANDS
S. VAN ERP	78731	
DICK VAN LEER	94545	
PIERRE VAN NYPELSTEER	B-1050	BELGIUM
JOHN VAN ROEKEL	48176	
DWIGHT VANDENBERGHE	98133	
MICHAEL W. VANNIER	63110	
W. VAUGHN	85019	
JAMES A. VELLENGA	55440	
P. VERBAETEN	B-3030	BELGIUM
M. H. VERHAART		NEW ZEALAND
JIM VERNON	55440	
VINCENT VIGUS	92634	
RICHARD C. VILE JR.	48106	
JOHN V. VILKAITIS	06787	
ADRIAN VILLANUSTRE	RA-1425	ARGENTINA
RICHARD VILMUR	60104	
ROBERT VINCENT	01880	
JOHN S. WADDELL	45387	
C. J. WADDINGTON	55455	
BOB WALLACE	98007	
C. S. WALLACE	3168	AUSTRALIA
DAVE WALLACE	94598	
DAVID R. WALLACE	85021	
BRUCE D. WALSH	91301	
DAVID P. WALSH	20901	
MARIE WALTER	92714	
DAVID WARD	K1A 0R6	CANADA
MIKE WARDALE	M1R 5A6	CANADA
SCOTT K. WARREN	77005	
ALLEN A. WATSON	07602	
JOHN L. WEAVER	79604	
L. KIRK WEBB	78712	
NEIL W. WEBRE	93407	
WALTER WEHINGER	D-7000	GERMANY
DAVID M. WEIBLE	60680	
KEVIN WEILER	15213	
RUTH WEINBERG		ISRAEL
LEN WEISBERG	94304	
JOE WEISMAN	95452	
RAY WEISS	90801	
J. R. WEISTART	62563	
TOM WEISZ	48103	
ANTHONY B. WELLER	WC1H 0AH	UNITED KINGDOM
PETER WENTWORTH	6000	SOUTH AFRICA
BOB WERNER	52333	

JOHN P. WEST	30327	
DANA WHEELER	94707	
NORM WHEELER	90230	
DONALD E. WHILE	44141	
C. G. WHITAKER	MK43 0AL	UNITED KINGDOM
N. WHITE	ST5 5BG	UNITED KINGDOM
L. P. WHITEHEAD	3131	AUSTRALIA
P. WHITEHEAD	SW7 2BX	UNITED KINGDOM
AKE WIKSTROM	S-402 20	SWEDEN
D. M. WILBORN	90746	
ALAIN D. D. WILLIAMS	NW11 8DP	UNITED KINGDOM
C. J. WILLIAMS	M3A 1M3	CANADA
KIM WILLIAMS	V3N 4N8	CANADA
NIGEL WILLIAMS	7007	AUSTRALIA
H. WILLMAN	01730	
ROY A. WILSKER	02114	
FRED WILSON	91602	
JAN R. WILSON	70808	
ROBERT WILSON	L6T 3Y3	CANADA
JIM WINSALLER	93111	
LEESON J. I. WINTER	01720	
DAVID S. WISE	47401	
MARK WOLCOTT	48130	
CHARLES A. WOLFE	91342	
TOM WOLFE	91107	
HENRY WOOD	08540	
STEPHEN E. WOODBRIDGE	32905	
ANDREW S. WOYAK	55405	
DON M. WRATHALL	85704	
H. R. WRIGHT	19102	
TOM WRIGHT	55420	
RUDOLF F. WROBEL	66216	
JOHN C. WYMAN	13206	
MICHAEL T. WYMAN	02154	
MINEO YAMAKAWA	73190	
EARL M. YARNER	02168	
D. J. YATES	4067	AUSTRALIA
BRADLEY N. YEARWOOD	90024	
KIET T. YEN	55455	
FRANCIS W. YEUNG	18976	
PERTTI YLINEN	SF-33900	FINLAND
CHRISTOPHER YORK	10028	
JAMES YORK	92805	
H. YOSHIDA	02173	
R. M. YOUNG	J8X 1C6	CANADA
COLEMAN YOUNGDAHL	95476	
ALEXANDER YUILL-THORNTON II	94941	
PETER H. ZECHMEISTER	55455	
R. ZECTZER	3000	AUSTRALIA
ERWIN ZEDNIK	D-8000	GERMANY
FRED ZEISE	95051	
PETE ZIEBELMAN	77036	
STEPHEN N. ZILLES	95030	
ANDREW HARRIS ZIMMERMAN	95132	
PHILIP R. ZIMMERMAN JR.	80302	
DONALD A. ZOCCHI	97005	
TOM ZWITTER	44022	

****

Introduction

The application notes introduced a few issues ago continue to flourish. However we do have some problems at PN headquarters in checking the quality of programs submitted, and therefore we welcome any comment or certification of correctness by readers.

This section has elicited much favourable comment. Our thanks to those members who wrote in to let us know what they thought, and especially to those who submitted programs.

Applications

News

Business Packages available

Cyber-Score Inc, Software Dept, Suite 406 - The Riker Building, 35 West Huron Street, Pontiac, Michigan 48058 (313-338-6317) have advertised Pascal-written software that includes Depreciation, Interest, Checking, Metric, Base2816, Sort1, Sort2, Form1040, Stocks, Handicap, Calculator, Decision, and Vol 2 for Business soon to be released.

NorthWest Microcomputer Systems, 121 East Eleventh Street, Eugene, Oregon 97401 (503-485-0626) have vintage turnkey business systems, including Accounts Receivable, Word Processing, Client Information Management, General Ledger, Fuel Dispensing & Accounting.

P.S.Inc, Fargo, North Dakota have Pascal business accounting packages including a general ledger, accounts payable, accounts receivable, inventory control, order entry. All seem to be linked together into a single comprehensive system.

Interactive Technology Inc, 14350 NW Science Park Drive, Portland, Oregon 97229 (503-644-0111) are "simply ecstatic over recent articles and the general enthusiasm that is growing for Pascal." In a recent letter, they gave us a lot of information on their plans (see Open Forum).

This happily matches up with the requests from James A. Anderson, Arnold Bob, Ken Leese, Monte Jay Meldman and Nield Overton, who are all looking for business-applications software. (See Here and There (Tidbits) Section except for Ken.)

Data-Base Management Systems

Wilhelm Burger in Texas is working on a DBMS system in Pascal. Its seems he is working with the AAEC IBM 360/370 Pascal, and has a Parser Generator, but is now working on the Data Base Manager.

Boeing Computer Services in Seattle, Washington is developing a sophisticated data base management system in Pascal.

Interpreters

An APL interpreter written in Pascal won the first prize in the "Great APL Contest" of Byte Magazine. The authors were Alan Kaniss, Vincent DiChristofaro & John Santini of 1327 McKinley Street, Philadelphia PA 19111. The program is described in Byte, June 1979, for those interested.

A portable LISP interpreter has been developed under Contract W-7405-ENG-48 for the US Department of Energy by L.A.Cox and W.P.Taylor. The Report is available from NTIS as Order Number #UCRL-52417 at $4.00 per paper copy. The title is "A Portable LISP Interpreter", and the complete interpreter (in Pascal) is given. Cox & Taylor worked for UC Lawrence Livermore Laboratory, Livermore, CA.

Inter-language translators

Roy Freak at the University of Tasmania has written a Fortran to Pascal translator which has successfully translated over 170 Fortran programs into Pascal, including some difficult examples from Ed Yourdon's books and some Fortran test programs that found their way into the Pascal Validation Suite (for testing the accuracy of sin, cos, etc).

The translator makes an extensive analysis of the Fortran text, and is about the size of a large compiler. It is designed both to preserve equivalence in its transformations and to produce as good Pascal as can be achieved. It analyses expressions to see where Pascal's precedence rules require extra parentheses, analyses the control flow structure to try to produce whiles, ifs, cases, etc from Fortran's constructs, and analyses the call structure

Applications

so that it can nest procedure subprograms as deeply as their usage allows. It also handles COMMON and EQUIVALENCE by making some assumptions about Pascal representation mapping. These extensive analyses make the translation a relatively slow process for some of those very large complicated Fortran programs one sees sometimes, but most programs or subprograms are translatable in a reasonable time (limited by lexical analysis and other factors).

The translator does not handle Fortran I/O (because it needs run-time information to do a complete job, or knowledge of intent), nor does it handle adjustable arrays completely (because the facility is not in Pascal). Outside these restrictions however, the translated Pascal version should be ready to compile, or to be massaged by hand should the user have to cope with non-standard Fortran or wish to improve the program. Unfortunately the translator runs only on Burroughs B6700 computers (and compatible machines) because it is written in Burroughs Algol and uses random-access disk files to store its program blocks.

Bits & Pieces

William G Hutchison wins our "PUG Friend of the Month" award. With all the interesting information received, a virtual Captain Pascal Magic Ring is on its way. Bill writes:

"1. Glad you liked the LLL Lisp system. It looks like a very clean and extendable system.

"2. It appears that the Kernighan & Plauger "Software Tools" may soon be available in Pascal. See the writeup from the Ratfor Newsletter - "Rat Informant". Names like PUG and RAT are so bad they give me MUMPS!

"3. Newman & Sproull "Principles of Interactive Computer Graphics" Second Edition McGraw-Hill 1979 uses Pascal to "publish" graphics algorithms. Unfortunately, they merely left out the hidden line program listings, rather than be bothered to translate them from SAIL to Pascal. So the new edition is streamlined, but less complete.

"4. I would like to use the programs published in the PN, but I can't use any of them. They all use Standard Pascal or extension features not available in the P4 subset, which is all that I have at my disposal."

{ P4 is neither a subset of Pascal, nor an acceptable standard. We encourage PUG members to implement all of Pascal. }

{ The extract from Rat Informant reads: "Several people have attempted translations from Ratfor to other languages including Pascal, C, Algol, BCPL, and Basic (yes, even Basic ...)." This may not mean what Bill thinks, but it is intriguing to speculate on what might happen if all the Software Tools were to be pascalized, perhaps by the Fortran to Pascal translator. }

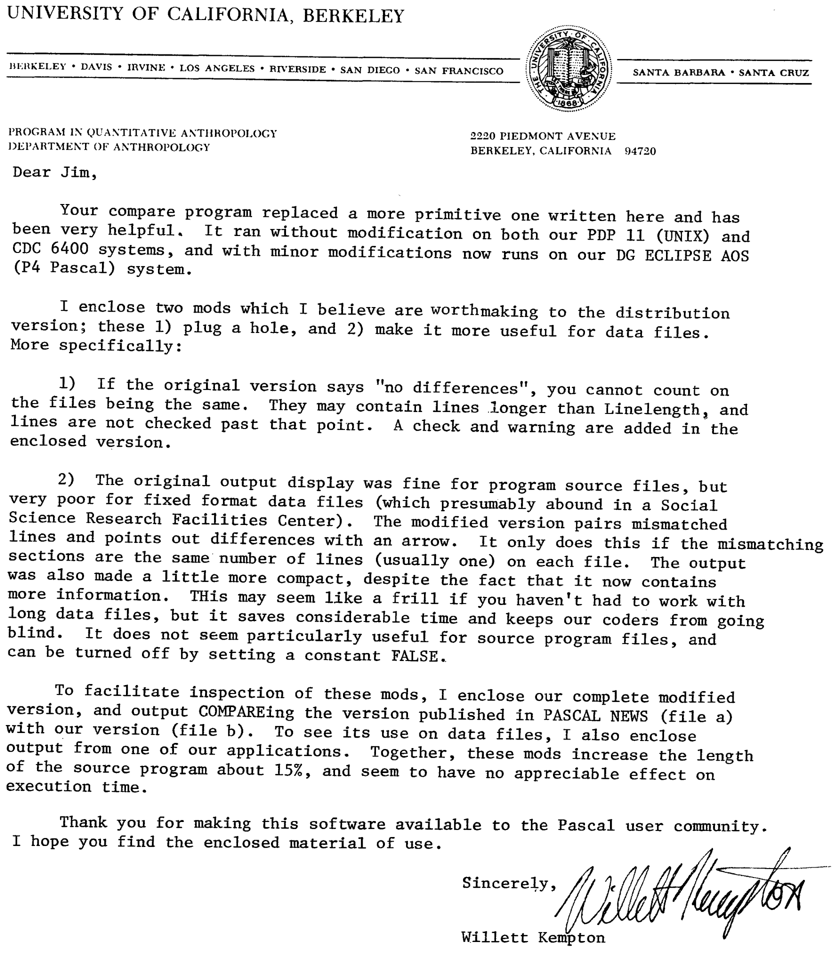
Donald Knuth has developed a system called TEX (Tau Epsilon Xi -- rhymes with "Tech") for producing beautiful typography for programs and programmers (including mathematicians as a subset of the above). See the article "Mathematical Typography" in the Bulletin of the American Mathematical Society, Vol 1 No 2 March 1979 (New Series). We understand that the original program, written in SAIL (or MAINSAIL, we're not too sure) is being translated into Pascal and this version will be the eventually published one. All Pascalers will applaud using Pascal to bootstrap more elegance into our systems.

Rich Cichelli reports that ANPA/RI are close to having an enhanced version of the North American Philips conformity checker for Pascal. He says it is a priority project at ANPA/RI.

Software Tools

Changes to S-1 "Compare" (See PN#12, June 1978, page 20.)

Willett Kempton has certified use of Compare (Software Tool S-1), and sent in some corrections to fix up a bug and improve the product. We are publishing the comparison output of Compare run on itself and on its enhanced brother below together with the letter. Readers will undoubtedly note that the version of Compare used to produce the listing has a few (no doubt machine-dependent) features not in the standard-conforming version. The letters "a" and "b" at the left margin indicate the source of the lines, and the '^' marks the line changes where these are minor. We have heard of many other places where Compare has been used successfully.



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

Dear Jim,

Your compare program replaced a more primitive one written here and has been very helpful. It ran without modification on both our PDP 11 (UNIX) and CDC 6400 systems, and with minor modifications now runs on our DG ECLIPSE AOS (P4 Pascal) system.

I enclose two mods which I believe are worthmaking to the distribution version; these 1) plug a hole, and 2) make it more useful for data files. More specifically:

1) If the original version says "no differences", you cannot count on the files being the same. They may contain lines longer than Linelength, and lines are not checked past that point. A check and warning are added in the enclosed version.

2) The original output display was fine for program source files, but very poor for fixed format data files (which presumably abound in a Social Science Research Facilities Center). The modified version pairs mismatched lines and points out differences with an arrow. It only does this if the mismatching sections are the same number of lines (usually one) on each file. The output was also made a little more compact, despite the fact that it now contains more information. THis may seem like a frill if you haven't had to work with long data files, but it saves considerable time and keeps our coders from going blind. It does not seem particularly useful for source program files, and can be turned off by setting a constant FALSE.

To facilitate inspection of these mods, I enclose our complete modified version, and output COMPAREing the version published in PASCAL NEWS (file a) with our version (file b). To see its use on data files, I also enclose output from one of our applications. Together, these mods increase the length of the source program about 15%, and seem to have no appreciable effect on execution time.

Thank you for making this software available to the Pascal user community. I hope you find the enclosed material of use.

Sincerely, Willett Kempton

Willett Kempton

```
    compare.  version 1.3   (7 Nov 78)

match criterion = 3 lines.

filea: compare.origin
fileb: compare.new
            ************************************
extra text:  on fileb,  between lines 46 and 47 of filea

  b   47   *      Another program parameter (constant), "Markunequalcolumns",
  b   48   *      specifies that when unequal lines are found, each line from
  b   49   *      Filea is printed next to its corresponding line from Fileb,
  b   50   *      and unequal columns are marked.  This option is particularly
  b   51   *      useful for fixed-format data files.  Notes: Line pairing is
  b   52   *      not attempted if the mismatching sections are not the same
  b   53   *      number of lines on each file. It is not currently very smart
  b   54   *      about ASCII control characters like tab. (W.Kempton, Nov 78)
  b   55   *

            ************************************
mismatch:    filea, line 63   not equal to    fileb, line 72:

  a   63      version = '1.2p  (78/03/01)';
  b   72      version = '1.3   (7 Nov 78)';
                           ^^    ^^^^^^^

            ************************************
extra text:  on fileb,  between lines 66 and 67 of filea

  b   76      markunequalcolumns = true;     { IF UNEQUAL LINES ARE TO BE PAIRED,  }
  b   77                                     {  AND UNEQUAL COLUMNS MARKED         }

            ************************************
extra text:  on fileb,  between lines 78 and 79 of filea

  b   90          name : char;

            ************************************
extra text:  on fileb,  between lines 98 and 99 of filea

  b  111    linestoolong : boolean;          { FLAG IF SOME LINES NOT COMPLETELY CHECKED }

            ************************************
extra text:  on fileb,  between lines 151 and 152 of filea

  b  165          if not eoln(filex) then  linestoolong := true;

            ************************************
mismatch:    filea, lines 285 thru 292   not equal to    fileb, lines 299 thru 316:

  a  285    procedure writetext(p, q : linepointer);
  a  286    begin { WRITETEXT }
  a  287      writeln;
  a  288      while (p <> nil) and (p <> q) do
  a  289        begin  write(' * ');
  a  290          if p^.length = 0 then writeln
  a  291          else writeln(p^.image : p^.length);
  a  292          p := p^.nextline

  b  299    procedure writeoneline(name : char;  l : integer;  p : linepointer);
  b  300    begin  { WRITEONELINE }
  b  301          write('    ', name, l:5,'  ');
  b  302          if p^.length = 0 then writeln
  b  303          else writeln(p^.image : p^.length);
  b  304    end;  { WRITEONELINE  }
  b  305
  b  306    procedure writetext(var x : stream);
  b  307      { WRITE FROM X.HEAD TO ONE LINE BEFORE X.CURSOR }
  b  308      var
  b  309        p, q : linepointer;  lineno : integer;
  b  310    begin { WRITETEXT }
  b  311      p:=x.head;  q:=x.cursor;   lineno:=x.headlineno;
  b  312      while (p <> nil) and (p <> q) do
  b  313        begin
  b  314          writeoneline( x.name, lineno, p);
  b  315          p := p^.nextline;
  b  316          lineno := lineno + 1;

            ************************************
extra text:  on fileb,  between lines 297 and 298 of filea

  b  322    procedure writepairs( pa, pb : linepointer;  la, lb : integer);
  b  323      { THIS WRITES FROM THE HEAD TO THE CURSOR, LIKE PROCEDURE WRITETEXT. }
  b  324      { UNILKE PROCEDURE WRITETEXT, THIS WRITES FROM BOTH FILES AT ONCE,   }
  b  325      { COMPARES COLUMNS WITHIN LINES, AND MARKS UNEQUAL COLUMNS     }
  b  326    var
  b  327      tempa, tempb : array [1..linelength] of char;
  b  328      col, maxcol  : integer;
  b  329    begin  { WRITEPAIRS }
  b  330      repeat
  b  331        writeoneline('a', la, pa);   writeoneline('b', lb, pb);
  b  332        unpack(pa^.image,tempa,1);   unpack(pb^.image,tempb,1);
  b  333        if  pa^.length > pb^.length
  b  334                then maxcol := pa^.length else maxcol := pb^.length;
  b  335        write(' ': 11);  {11 spaces used for file name and line number }
  b  336        for col := 1 to maxcol do
  b  337             if tempa[col] = tempb[col] then write(' ') else write('^');
  b  338        writeln;  writeln;
  b  339        pa := pa^.nextline;  la := la + 1;
  b  340        pb := pb^.nextline;  lb := lb + 1;
  b  341      until (pa = a.cursor) or (pa = nil);
  b  342    end;  { WRITEPAIRS }
  b  343

            ************************************
mismatch:    filea, line 305   not equal to    fileb, line 351:

  a  305      else write('s ', f:1, ' to ', l:1);
  b  351      else write('s ', f:1, ' thru ', l:1);
                                       ^^^^^^^^^^^^

            ************************************
mismatch:    filea, lines 309 thru 319   not equal to    fileb, lines 355 thru 366:

  a  309    procedure printextratext(var x : stream; xname : char;
  a  310                             var y : stream; yname : char);
  a  311    begin { PRINTEXTRATEXT }
  a  312      write(' extra text on file', xname, ', ');
  a  313      writelineno(x);  writeln;
  a  314      if y.head = nil then
  a  315        writeln(' before eof on file', yname)
  a  316      else
  a  317        writeln(' between lines ', y.headlineno-l:1, ' and ',
  a  318                y.headlineno:1, ' of file', yname);
  a  319      writetext(x.head, x.cursor)

  b  355    procedure printextratext(var x, y : stream);
  b  356
  b  357    begin { PRINTEXTRATEXT }
  b  358      write(' extra text:  on file', x.name, ', ');
  b  359
  b  360      if y.head = nil then
  b  361        writeln(' before eof on file', y.name)
  b  362      else
  b  363        writeln(' between lines ', y.headlineno-1:1, ' and ',
  b  364                y.headlineno:1, ' of file', y.name);
  b  365      writeln;
  b  366      writetext(x)

            ************************************
mismatch:    filea, line 323   not equal to    fileb, line 370:

  a  323      writeln(' *************************************');
  b  370      writeln(' ':11, '*************************************');
                        ^^^^^^^                               ^^^^^^^^^^

            ************************************
mismatch:    filea, lines 327 thru 335   not equal to    fileb, lines 374 thru 386:

  a  327        if emptya then printextratext(b, 'b', a, 'a')
```

```
a  328        else printextratext(a, 'a', b, 'b')
a  329      else
a  330        begin
a  331          writeln(' mismatch:');  writeln;
a  332          write(' filea, ');  writelineno(a);  writeln(':');
a  333          writetext(a.head, a.cursor);
a  334          write(' fileb, ');  writelineno(b);  writeln(':');
a  335          writetext(b.head, b.cursor)

b  374        if emptya then printextratext(b, a)
b  375        else printextratext(a, b)
b  376      else
b  377        begin
b  373          write(' mismatch:   ');
b  379          write(' filea, ');  writelineno(a);  write('   not equal to    ');
b  380          write(' fileb, ');  writelineno(b);  writeln(':');  writeln;
b  381          if  markunequalcolumns    and
b  382              ((a.cursorlineno - a.headlineno) = (b.cursorlineno - b.headlineno))
b  383          then
b  384            writepairs(a.head, b.head, a.headlineno, b.headlineno)
b  385          else
b  386           begin writetext(a); writetext(b) end

        *************************************
extra text:  on fileb,  between lines 374 and 375 of filea

  b  425    a.name := 'a';  b.name := 'b';
  b  427    linestoolong := false;

        *************************************
extra text:  on fileb,  between lines 393 and 394 of filea

  b  447        if linestoolong then
  b  448          begin    writeln;
  b  449            writeln(' WARNING:  some lines were longer than ',
  b  450                                 linelength:1, ' characters.');
  b  451            writeln('             they were not compared past that point.'):
  b  452          end;
```

S-2 "Augment" and "Analyze" (See PN#12, June 1978, page 23.)

Sam Hills, Crescent City Computer Club, New Orleans, has prepared a machine-dependent version of Augment and Analyze for the Zurich dialect of the Dec-10 Pascal, and is working on a similar modification to accept a new dialect from the University of Texas. The program is available presumably, with documentation, from Sam Hills, 3514 Louisiana Avenue Parkway, New Orleans, LA 70125 (79 Apr 16).

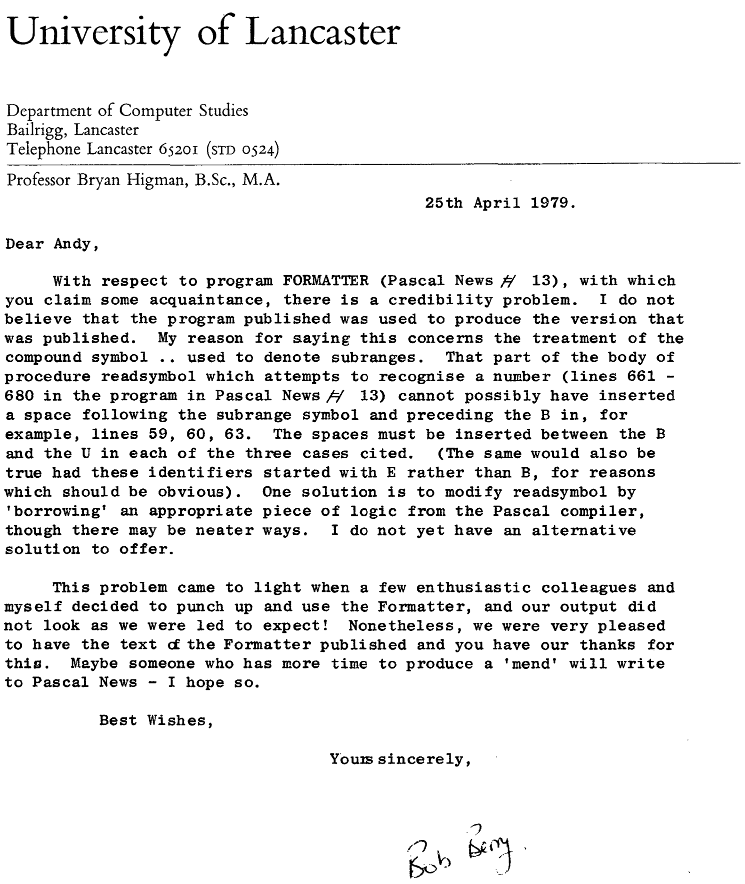
{ Note that this version is ONLY useful to DEC-10 users; it accepts non-standard statements as input and has various "chaining" features. }

S-3 "Prettyprint" (See PN#13, December 1978, page 34.)

Unfortunately, we've misplaced a letter from an eagle-eyed reader which complained about a conflict in the documentation for PRETTY. Indentation Rule 3 clearly states the style for IF-THEN-ELSE. However, lines 336-356 of the source program clearly show that Prettyprint processing itself can produce different results. The reason is that General Pretty printing rule 1 overrides all other rules. In a sense, then, blank lines and blanks are directives to the pretty printer.

S-4 "Format" (See PN#13, December 1978, page 45.)

We received many reports (unfortunately) of bugs in Format. For example, George Gonzales has sent a corrected though heavily modified version, fixing more than a dozen problems. We plan to print a list of corrections as soon as we can find the time. Bob Berry sent the nice letter below:



University of Lancaster

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

Professor Bryan Higman, B.Sc., M.A.

25th April 1979.

Dear Andy,

With respect to program FORMATTER (Pascal News # 13), with which you claim some acquaintance, there is a credibility problem. I do not believe that the program published was used to produce the version that was published. My reason for saying this concerns the treatment of the compound symbol .. used to denote subranges. That part of the body of procedure readsymbol which attempts to recognise a number (lines 661 - 680 in the program in Pascal News # 13) cannot possibly have inserted a space following the subrange symbol and preceding the B in, for example, lines 59, 60, 63. The spaces must be inserted between the B and the U in each of the three cases cited. (The same would also be true had these identifiers started with E rather than B, for reasons which should be obvious). One solution is to modify readsymbol by 'borrowing' an appropriate piece of logic from the Pascal compiler, though there may be neater ways. I do not yet have an alternative solution to offer.

This problem came to light when a few enthusiastic colleagues and myself decided to punch up and use the Formatter, and our output did not look as we were led to expect! Nonetheless, we were very pleased to have the text of the Formatter published and you have our thanks for this. Maybe someone who has more time to produce a 'mend' will write to Pascal News - I hope so.

Best Wishes,

Yours sincerely,

Bob Berry.

TRUE CONFESSIONS

I (Andy) shamefacedly admit to having edited the ".." symbol in several places. What happened was this: as I was preparing the source of Format for publication I noticed several bothersome rough places. One of these was no blank preceding some occurrences of "..". Because this appeared in both the source and the result of Format run on itself, I edited the result not thinking that this was an ingrained symptom of Format being continually run across itself (well before I received it). Another rough spot I confess to "fixing" was the ugly breaking upon wraparound of several expressions in assignment statements. I'm very sorry.

S-5 ID2ID

Recoding a Pascal Program Using ID2ID

Andy Mickel
University Computer Center
University of Minnesota
Minneapolis, MN 55455 USA

Copyright (c) 1979.

What ID2ID Does

ID2ID is a program designed to quickly and accurately edit the text of a Pascal program by substituting new identifiers for existing ones. A typical use might be to recode a program with longer, more descriptive identifiers to enhance the program’s readability.

Ordinary text editors are not necessarily good to use for this purpose because each identifier substitution requires one pass through the entire text of the source program. Also many text editors do not easily provide the means to distinguish whole identifiers from those identifiers which happen to contain other identifiers (for example, "int" versus "integer").

How ID2ID Works

ID2ID accepts two input files: "SOURCE", a text file consisting of a Pascal source program, and "IDPAIRS" a text file consisting of pairs of identifiers in the form: OLDID,NEWID one pair to a line.

An identifier in a Pascal program consists of a letter followed by zero or more letters or digits. ID2ID imposes a practical maximum length of 25 characters for any identifier. This means that ID2ID will not distinguish between two identifiers which do not differ in their first 25 characters.

ID2ID reads the file of identifier pairs and builds a search tree which is then used to look up identifiers during the scanning of the source program. Two output files are generated: "TARGET", a text file consisting of the edited source of the Pascal program with new identifiers and "REPORT", a text file consisting of warning and error messages accumulated during editing.

Several situations can pose problems to the process of identifier substitution:

1. An "oldid" may appear more than once in the IDPAIRS file. This prevents a unique substitution, and ID2ID halts and displays the message:. "DUPLICATE OLDID: ___ ".

2. A warning message is issued in the case of duplicate "newid’s". This is just to let you know that you may not have intended to rename two "oldid’s" to the same "newid".

3. A warning message is issued if ID2ID encounters a program "sourceid" which is the same as a "newid". You may not have realized that you picked a "newid" which already existed as an identifier in the source program.

Of course an "oldid" in one "oldid,newid" pair may have the same spelling as a "newid" in a different "oldid,newid" pair.

In scanning the source program, ID2ID recognizes all identifiers including Pascal reserved words. Of course, identifiers within comments and strings are unchanged. The "E" used to specify exponents in real numbers is distinguished from an ordinary identifier spelled "E".

How to Use ID2ID

ID2ID is available as an operating-system control statement on CDC 6000/Cyber 70,170 computer systems. The general form of the control statement is:

```
ID2ID(SOURCE,TARGET,IDPAIRS,REPORT)
```

Assuming SOURCE and IDPAIRS are local files, ID2ID will produce results on files TARGET and REPORT. For example:

Suppose SOURCE is:

```
PROGRAM EXAMPLE(OUTPUT);
 VAR VARA, VARX, VARY: INTEGER;
BEGIN
 VARX := 24;
 VARY := 80;
 VARA := VARX * VARY;
 WRITELN(’CHARACTERS = ’, VARA)
END.
```

and IDPAIRS is:

```
VARA,CHARACTERS
VARX,LINES
VARY,CHARSPERLINE
```

then the TARGET produced by ID2ID is:

```
PROGRAM EXAMPLE(OUTPUT);
 VAR CHARACTERS, LINES, CHARSPERLINE: INTEGER;
BEGIN
 LINES := 24;
 CHARSPERLINE := 80;
 CHARACTERS := LINES * CHARSPERLINE;
 WRITELN(’CHARACTERS = ’, CHARACTERS)
END.
```

ID2ID uses an AVL-balanced binary tree of identifiers, so it is not affected by the order in which the identifier pairs are presented on the IDPAIRS file. The above program was processed in 0.043 seconds by ID2ID on a Cyber 172 computer using Pascal-6000 Release 3. A program consisting of 891 identifiers on 400 lines was processed with ID2ID with 58 pairs of identifier substitutions in 1.624 seconds on a 172 using Release 3.

History

ID2ID was originally designed and written by John T. Easton and James F. Miner at the Social Science Research Facilities Center in 1976 to provide a reliable means of transforming poorly coded Pascal programs into tolerable ones. Subsequent refinements were added by Andy Mickel and Rick L. Marcus at the University Computer Center in 1978 to improve its ease of use and its error processing.

ID2ID was redesigned in 1979 by James F. Miner and Andy Mickel to incorporate a better identifier table and secure error processing. This necessitated a complete rewrite of the program. ID2ID has now joined a long list of other Pascal software-writing tools.

```
{*        ID2ID - Rename Identifiers In a Pascal Program.
 *
 *        James F. Miner    79/06/01.
 *           Social Science Research Facilities Center.
 *        Andy Mickel        79/06/28.
 *           University Computer Center
 *        University of Minnesota
 *        Minneapolis, MN 55455 USA     Copyright (c) 1979.
 *
 *           (Based on an earlier version by John T. Easton and
 *            James F. Miner, 76/11/29, as modified by Andy Mickel
 *            and Rick L. Marcus, 78/12/08)
 *
 *        THE NAMES AND ORGANIZATIONS GIVEN HERE MUST NOT BE DELETED
 *        IN ANY USE OF THIS PROGRAM.
 *
 *        See the PTOOLS writeup for external documentation.
 *
 *
 **       ID2ID - Internal documentation.
 *
 *           ID2ID reads a file of IDPAIRS and builds an AVL-balanced
 *        binary tree of identifiers while checking for duplicates.  It
 *        then reads the SOURCE program and edits it to a TARGET file by
 *        substituting identifiers found in the tree.  A final check is
 *        made for new identifiers which were already seen in the
 *        SOURCE, and a REPORT may be generated.
 }

program ID2ID(Source, Target, IdPairs, Report);

  label
    13 { FOR FATAL ERRORS };

  const
    MaxLength = 25;
       Blanks = '                          { MUST BE MaxLength LONG };

  type
      CharSet = set of Char;
     IdLength = 1 .. MaxLength;
       IdType = record
                    Name: packed array [IdLength] of Char;
                  Length: IdLength
                end;
      Balance = (HigherLeft, Even, HigherRight);
      NodePtr = ↑ Node;
         Node = record
                         Id: IdType;
                       Left,
                      Right: NodePtr;
                        Bal: Balance;
                    IdIsNew: Boolean;
                case
                    IdIsOld: Boolean of
                  True:
                    (NewPtr: NodePtr);
                  False:
                    (SeenInSource: Boolean)
                end;
  var
       IdTable: NodePtr { SYMBOL TABLE };

       IdPairs,
        Source,
      Target,
      Report: Text;

     Letters,
      Digits,
      LettersAndDigits: CharSet;


procedure Initialize;

begin
  Rewrite(Report);
  Letters := ['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',
              '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 := ['0' .. '9'];
  LettersAndDigits := Letters + Digits;
end { Initialize };


procedure ReadId(var InFile: Text; var Ident: IdType);

  var
    ChCount: 0 .. MaxLength;

begin
  Ident.Name := Blanks;   ChCount := 0;
  repeat
    ChCount := ChCount + 1;   Ident.Name[ChCount] := InFile↑;   Get(InFile)
  until not (InFile↑ in LettersAndDigits) or (ChCount = MaxLength);
  Ident.Length := ChCount
end { ReadId };


procedure ReadIdPairsAndCreateSymbolTable;

  type
    IdKind = (OldKind, NewKind);

  var
      OldId,
      NewId: IdType;
       Link: NodePtr { REMEMBER NewId POINTER };
    LineNum: Integer;
    IncrHgt: Boolean;


  procedure Error;

  begin
    WriteLn(Report, 'on line number ': 29, LineNum: 1,
            ' of the "IdPairs" file.');
  end { Error };


  procedure Enter(var Identifier: IdType; Kind: IdKind; var P: NodePtr;
                  var IncreasedHeight: Boolean);

{ Enter USES AN AVL-BALANCED TREE SEARCH ALGORITHM BY NIKLAUS WIRTH. }
{      (SEE SECTION 4.4 IN "ALGORITHMS + DATA STRUCTURES = PROGRAMS" }
{       PRENTICE HALL, 1976, PP. 215-222.)                           }

    var
      P1,
      P2: NodePtr;
  begin
    if P = nil then
      begin { Id NOT FOUND IN TREE; INSERT IT. }
        New(P);   IncreasedHeight := True;
        with P↑ do
          begin
            Id := Identifier;
            IdIsNew := Kind = NewKind;   IdIsOld := Kind = OldKind;
            Left := nil;   Right := nil;   Bal := Even;
            if IdIsNew then begin Link := P;  SeenInSource := False end
            else NewPtr := Link
          end
      end
    else
      if Identifier.Name < P↑.Id.Name then
        begin
          Enter(Identifier, Kind, P↑.Left, IncreasedHeight);
          if IncreasedHeight then { LEFT BRANCH HAS GROWN HIGHER }
            case P↑.Bal of
              HigherRight:
                begin P↑.Bal := Even;   IncreasedHeight := False end;
              Even:
                P↑.Bal := HigherLeft;
              HigherLeft:
                begin { REBALANCE }
                  P1 := P↑.Left;
                  if P1↑.Bal = HigherLeft then
                    begin { SINGLE LL ROTATION }
                      P↑.Left := P1↑.Right;   P1↑.Right := P;
                      P↑.Bal := Even;   P := P1
                    end
                  else
                    begin { DOUBLE LR ROTATION }
                      P2 := P1↑.Right;   P1↑.Right := P2↑.Left;
                      P2↑.Left := P1;   P↑.Left := P2↑.Right;
                      P2↑.Right := P;
                      if P2↑.Bal = HigherLeft then P↑.Bal := HigherRight
                      else P↑.Bal := Even;
                      if P2↑.Bal = HigherRight then P↑.Bal := HigherLeft
                      else P1↑.Bal := Even;
                      P := P2
                    end;
                  P↑.Bal := Even;   IncreasedHeight := False;
                end;
            end { CASE }
        end
      else
        if Identifier.Name > P↑.Id.Name then
          begin
            Enter(Identifier, Kind, P↑.Right, IncreasedHeight);
            if IncreasedHeight then { RIGHT BRANCH HAS GROWN HIGHER }
              case P↑.Bal of
                HigherLeft:
                  begin P↑.Bal := Even;   IncreasedHeight := False end;
                Even:
                  P↑.Bal := HigherRight;
                HigherRight:
                  begin { REBALANCE }
                    P1 := P↑.Right;
                    if P1↑.Bal = HigherRight then
                      begin { SINGLE RR ROTATION }
                        P↑.Right := P1↑.Left;   P1↑.Left := P;
                        P↑.Bal := Even;   P := P1
                      end
                    else
                      begin { DOUBLE RL ROTATION }
                        P2 := P1↑.Left;   P1↑.Left := P2↑.Right;
                        P2↑.Right := P1;   P↑.Right := P2↑.Left;
                        P2↑.Left := P;
                        if P2↑.Bal = HigherRight then P↑.Bal := HigherLeft
                        else P↑.Bal := Even;
                        if P2↑.Bal = HigherLeft then P1↑.Bal := HigherRight
                        else P1↑.Bal := Even;
                        P := P2
                      end;
                    P↑.Bal := Even;   IncreasedHeight := False
                  end;
              end { CASE }
          end
        else
          begin { Identifier IS ALREADY IN TREE }
            IncreasedHeight := False;
            with P↑ do
              begin
                if IdIsOld then
                  if Kind = OldKind then { DUPLICATE OldId'S }
                    begin
                      WriteLn(Report, '*** Duplicate OldId''s encountered: ',
                         Identifier.Name);
                      Error;   goto 13
                    end
                  else begin IdIsNew := True;   Link := P end
                else
                  if Kind = NewKind then
                    begin
                      WriteLn(Report, '--- WARNING:  ', Identifier.Name,
                         ' has also appeared as another NewId');   Error;
                      Link := P
                    end
                  else begin IdIsOld := True;   NewPtr := Link end
              end
          end
  end { Enter };


  procedure Truncation(var Ident: IdType);

  begin
    WriteLn(Report, '--- WARNING:  Truncation for identifier, ', Ident.Name);
    WriteLn(Report, 'Extra characters ignored.': 39);   Error;
    repeat Get(IdPairs) until not (IdPairs↑ in LettersAndDigits);
  end { Truncation };


begin { ReadIdPairsAndCreateSymbolTable }
  IdTable := nil;   Reset(IdPairs);   LineNum := 1;   IncrHgt := False;
  while not EOF(IdPairs) do
    begin
      while (IdPairs↑ = ' ') and not EOLn(IdPairs) do Get(IdPairs);
      if IdPairs↑ in Letters then
        begin
          ReadId(IdPairs, OldId);
          if IdPairs↑ in LettersAndDigits then Truncation(OldId);
          while (IdPairs↑ in [' ', ',']) and not EOLn(IdPairs) do Get(IdPairs);
          if IdPairs↑ in Letters then
            begin
              ReadId(IdPairs, NewId);
              if IdPairs↑ in LettersAndDigits then Truncation(NewId);
              Enter(NewId, NewKind, IdTable, IncrHgt);
              Enter(OldId, OldKind, IdTable, IncrHgt);
            end
          else
            begin WriteLn(Report, '--- WARNING:  Malformed IdPair');   Error end
        end
      else
        begin WriteLn(Report, '--- WARNING:  Malformed IdPair');   Error end;
      ReadLn(IdPairs);   LineNum := LineNum + 1
    end
end { ReadIdPairsAndCreateSymbolTable };


procedure EditSourceToTarget;

  var
          SourceId: IdType;
           DigitsE,
    ImportantChars: CharSet;


  procedure Substitute(var Identifier: IdType; P: NodePtr);


    procedure WriteSourceId;

    begin
      with SourceId do Write(Target, Name: Length);
      while Source↑ in LettersAndDigits do
        begin Write(Target, Source↑);   Get(Source) end
    end { WriteSourceId };


  begin { Substitute }
    if P = nil then { Identifier NOT IN TREE, ECHO } WriteSourceId
    else
      if Identifier.Name < P↑.Id.Name then Substitute(Identifier, P↑.Left)
      else
        if Identifier.Name > P↑.Id.Name then Substitute(Identifier, P↑.Right)
        else { FOUND }
          with P↑ do
            if IdIsOld then
              begin
                with NewPtr↑.Id do Write(Target, Name: Length);
                while Source↑ in LettersAndDigits do Get(Source)
              end
                else begin SeenInSource := True;   WriteSourceId end
  end { Substitute };


begin { EditSourceToTarget }
  Reset(Source);   Rewrite(Target);
  ImportantChars := LettersAndDigits + ['(', '_', ''''];
  DigitsE := Digits + ['E', 'e'];
  while not EOF(Source) do
    begin
      while not EOLn(Source) do
        if Source↑ in ImportantChars then
          case Source↑ of
            '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',
            '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 ReadId(Source, SourceId);   Substitute(SourceId, IdTable)
              end;
            '0', '1', '2', '3', '4', '5', '6', '7', '8', '9':
              repeat Write(Target, Source↑);   Get(Source)
              until not (Source↑ in DigitsE);
            '''':
              begin
                repeat Write(Target, Source↑);   Get(Source)
                until (Source↑ = '''') or EOLn(Source);
                if EOLn(Source) then
                  WriteLn(Report, '--- WARNING:  Unclosed string found ',
                                  'in source program.');
                Write(Target, Source↑);   Get(Source)
              end;
            '(':
              begin
                Write(Target, Source↑);   Get(Source);
                if Source↑ = '*' then { COMMENT }
                  begin
                    repeat
                      Write(Target, Source↑);   Get(Source);
                      while Source↑ <> '*' do
                        begin
                          if EOLn(Source) then WriteLn(Target)
                          else Write(Target, Source↑);
                          Get(Source)
                        end;
                      Write(Target, Source↑);   Get(Source)
                    until Source↑ = ')';
                    Write(Target, Source↑);   Get(Source)
                  end
              end;
            '{':   { STDCOMMENT }
              begin
                repeat
                  if EOLn(Source) then WriteLn(Target)
                  else Write(Target, Source↑);
                  Get(Source)
                until Source↑ = '}';
                Write(Target, Source↑);   Get(Source)
              end
          end { CASE }
        else { OTHER CHARACTERS }
          begin  Write(Target, Source↑);   Get(Source) end;
      ReadLn(Source);   WriteLn(Target)
    end
end { EditSourceToTarget };


procedure CheckSeenInSource(P: NodePtr);

begin
  if P <> nil then
    begin
      CheckSeenInSource(P↑.Left);
      with P↑ do
        if IdIsNew and not IdIsOld then
          if SeenInSource then
            begin
              WriteLn(Report, '--- WARNING:  ', Id.Name: Id.Length,
                      ' was specified as a new identifier ');
              WriteLn(Report, 'and was also seen in the source ': 46,
                      'program unchanged.');
            end;
      CheckSeenInSource(P↑.Right)
    end
end { CheckSeenInSource };


begin { ID2ID }
  Initialize;
  ReadIdPairsAndCreateSymbolTable;
  EditSourceToTarget;
  CheckSeenInSource(IdTable);
  13:
end { ID2ID }.
```

S-6 Prose

Disclaimer:
The editors are not completely happy with the portability of this program, and several problems were noted in preparing it for publication. In particular, there is insufficient information about the Control Data conventions to help people to convert it to other systems. The pecularities of the 76B character escape and the segmented files are examples. Nevertheless, there is considerable demand for Prose to be released, and it is better than the other text-formatters we have seen.

Prose Instruction Manual

John P. Strait
University Computer Center
University of Minnesota

Copyright 1978

Abstract

Preparation and editing of prose (such as computer oriented documentation) is a tedious process. This process can be made somewhat easier through the use of computerized text processing tools such as text editors and formatters. This writeup describes a text formatting program named Prose. Prose and this instruction manual are oriented toward the preparation of computer oriented documentation, and so this writeup assumes basic knowledge of computer-related text processing tools.

Contents

****

The text examples in this manual have been extracted from Alice's Adventures in Wonderland by Lewis Carroll.

****

Historical Notes

Most of the text formatting programs available today descend from one of several original programs. Among these is RUNOFF which was developed on the Dartmouth Time-Sharing System in the 196Øs. Later, the Call-a-Computer system provided a RUNOFF version called EDIT RUNOFF as a text editor command. In 1972, Michael Huck, working on the University of Minnesota's MERITSS system (a CDC 64ØØ running the KRONOS operating system), began to develop a version of EDIT RUNOFF that he called TYPESET. TYPESET went through many developmental changes, and stabilized somewhat in early 1977 at version 5.Ø, which is written in CDC COMPASS assembly language. Prose is written in the programming language Pascal, and was developed over a year's time starting in the spring of 1977. The design of Prose was influenced heavily by TYPESET and so Prose is one of the many descendants of RUNOFF.

Philosophy, Goals, and Abilities

Prose is intended primarily for the preparation of machine retrievable documentation, and this has influenced the choice of its repertoire of abilities. TYPESET was intended as a "versatile text information processor commonly used to typeset theme papers, term papers, essays, letters, reports, external documentation ..., and almost any other typewritten text" [Typeset 5.Ø Information, Copyright 1977 by Michael Huck]. In spite of these aspirations, no program can be all things to all people, and so it is with Prose. It was intended that Prose be able to do most of the things that are needed to produce high quality computerized text.

The design of Prose was influenced by several goals. First, it should be possible to produce high quality results, with a minimum number of directives. Prose should have about 9Ø% of the abilities that you think are useful, and the 1Ø% it doesn't have should be the ones that are so esoteric that they are non-essential. Some text formatters take the approach of providing a minimum set of built-in abilities, along with a "general and powerful" feature such as macros. The idea is that you can accomplish anything you want (no matter how much effort it will take) by defining appropriate macros. The problem with this approach is that the user is forced to learn a complicated feature in order to produce any but the most trivial results.

Prose's philosophy is that the user should not be overwhelmed by a large number of complicated directives. That the syntax of the directives should be consistent. That the text should stand out, not the directives. Because of this desire for simplicity, Prose may or may not be the tool for a given application. The following two tables should aid in deciding whether or not to use Prose.

Prose ...

a. Prose has a small number of commands, which provide a learnable set of basic formatting abilities.
b. Prose can do underlining and discretionary hyphenation.
c. Prose can remember and restore the text processing environment.
d. Prose can produce mixed-case or upper-case-only output from either mixed-case or upper-case-only input.
e. Prose can accumulate and produce a sorted index, referring to page numbers.
f. Prose can print selected pages on request.
g. Prose can format text in pages with headers, footers, and other frills.
h. Prose can fill and justify text to specified margins.
i. Prose is an extremely portable program, written in standard Pascal, and it uses ASCII as its internal character code. It is written to encourage transportation between computers with different hardware and different operating systems.

... and Cons

a. Prose cannot control photo-typesetting machines.
b. Prose cannot do graphics.
c. Prose does not have multi-column ability.
d. Prose does not have macros, variables, or other programming language-like features.
e. Prose does not have the ability to store text and retrieve it later, with the exception of the special purpose indexing ability.
f. Prose does not have tabs.
g. Prose does not have directives to do everything you always wanted to.

Basic Units of Text

Some of the basic units of natural language are the word, the phrase, the sentence, and the paragraph. In text formatting, the word, the line, and the paragraph are the basic units. A word is defined as any non-blank string of characters, with a blank on either side. Thus, for the purposes of formatting, a punctuation character is part of the word it is next to. By default, Prose reformats its input by filling words into lines, adding blanks to justify the lines to left and right margins, and printing lines together to make paragraphs. In filling lines, Prose does not pay attention to the original positions of the words, but instead fills as many words as possible into the output lines, preserving the original order. The following example illustrates this process of filling and justifying.

Input to Prose:

```
     "When we were little," the Mock Turtle went on at last,
more calmly, though still sobbing a little now and then,
"we went to school in the sea.  The master was an old
Turtle--we used to call him Tortoise--"
     "Why did you call him Tortoise, if he wasn't one?"
Alice asked.
     "We called him Tortoise because he taught us," said the
Mock Turtle angrily.  "Really you are very dull!"
     "You ought to be ashamed of yourself for asking such a
simple question," added the Gryphon; and then
they both sat silent and looked at Alice, who felt ready to
sink into the earth.
```

Output from Prose:

```
     "When we were little," the Mock Turtle went on at last,
more calmly, though still sobbing a little now and then, "we
went to school in the sea.  The master was an old Turtle--we
used to call him Tortoise--"
     "Why did you call him  Tortoise,  if  he  wasn't  one?"
Alice asked.
     "We called him Tortoise because he taught us," said the
Mock Turtle angrily.  "Really you are very dull!"
     "You ought to be ashamed of yourself for asking such  a
simple  question," added the Gryphon; and then they both sat
silent and looked at Alice, who felt ready to sink into  the
earth.
```

Most of text formatting is filling and justifying. In the absence of special instructions to Prose (called directives), it will fill all of the input words into output lines, and justify all of those lines.

The distinction between one paragraph and the next is defined by a justification break, which causes Prose to stop filling the current output line, and print it without justifying. Since the break is one of the most frequently used instructions (as well as one of the simplest), it can be indicated in many ways. Paragraphs can be separated (broken) by one or more blank lines, by leading blanks typed on an input line (a paragraph indentation), or by the Prose ".BREAK" directive. The following example demonstrates these three methods.

Input to Prose

```
At last the Gryphon said to the Mock Turtle "Drive on,
old fellow!  Don't be all day about it!" and he went
on in these words:--
```

```
     "Yes, we went to school in the sea, though you mayn't
     believe it--"
     .BREAK
     "I never said I didn't!" interrupted Alice.
     .BREAK
     "You did," said the Mock Turtle.
          "Hold your tongue!" added the Gryphon, before Alice could
     speak again.
```

Output from Prose:

```
     At  last  the Gryphon said to the Mock Turtle "Drive on, old
     fellow!  Don't be all day about it!" and he went on in these
     words:--

     "Yes,  we  went  to  school  in  the  sea, though you mayn't
     believe it--"
     "I never said I didn't!" interrupted Alice.
     "You did," said the Mock Turtle.
          "Hold your tongue!" added the  Gryphon,  before  Alice
     could speak again.
```

When you use one of these methods to create a paragraph, Prose only does a justification break. That is, Prose will not skip lines or indent unless blank lines or indentations explicitly appear on the input file. There is a way to do fancier things by using the ".PARAGRAPH" directive, but that will be introduced later.

A General Look at Directives

In its default mode, Prose automatically fills and justifies output lines, and formats the output in pages. Directives are needed to instruct Prose to do anything more fancy. There are directives to change the margins, to control options, and to define the type of output device you intend to use.

A line of directives is indicated by typing the directive escape character in the first column of an input line. The period was chosen as the default directive escape character (although you can change it if you wish) because it seems very unlikely that anyone would want to type a period in the first column of a line of text. The entire line is scanned for directives. Several directives can be typed on the same line, provided that they are separated by the directive escape character. For example:

```
          .BREAK.SKIP 2.MARGIN( L5 R65 )
```

Some directives, however, take the remainder of the line as their parameter, and so no other directives can follow these. Long directives may extend to several lines. Continuation lines are indicated by a plus sign (+) typed in column one. The continuation may be made anywhere that a blank is allowed. For example:

```
          .FORM( [ /// L58 // #73 'PAGE' P /// ]
          +      [ /// L58 //     'PAGE' P /// ] )
```

Although the examples in this writeup will usually show directives typed entirely in upper case, upper and lower case letters may be intermixed.

Every directive begins with the name of the command, for instance "MARGIN". The name can always be abbreviated to three letters, and in fact, only the first three letters are examined by Prose. The name may be followed by a parameter, but in the absence of a parameter, default values are used. There are four forms for the parameter:

1) The absence of any parameter.
2) A single numeric value.
3) The remainder of the directive line.
4) A specification enclosed in parentheses, which consists of descriptors defined by the directive itself.

When a numeric value is required (for a parameter or as part of a descriptor), an explicit positive integer may be given. In many directives, a relative value may be used. This is indicated by a plus or minus sign before the integer, and indicates that the old value should be incremented or decremented by a certain amount. In the following example, the left margin is set to 10 and the right margin to 70. Then, the margins are squeezed together by 5 characters on both sides.

```
          .MARGIN( L10 R70 )
          .MARGIN( L+5 R-5 )
```

Controlling the Formatting Environment

The formatting environment is defined to be all the options and specifications that direct Prose as it produces formatted output from unformatted input. The concepts that make up the formatting environment can be loosely grouped into six areas, and there are directives to control each one:

1) INPUT controls the meaning and treatment of characters on the input file.
2) OUTPUT describes the type of output device for which the formatted result is intended.
3) FORM specifies the format of the page into which the running text will be inserted. This includes where to print titles, footers, and the like.
4) MARGIN sets the left and right margins.
5) PARAGRAPH describes special actions for the beginning of each paragraph.
6) OPTION controls the rest of the miscellaneous options that affect the text formatting process.

Of these six groups, the INPUT, MARGIN, OPTION, and PARAGRAPH settings are likely to be changed often throughout the text. There will probably be a small number of different settings, and it will be convenient to be able to resume old settings. To accomodate these needs, a simple device is available for these four directives.

When setting the options controlled by these directives, the following syntax is used:

```
          .directivename( parameters )
```

where the parameters consist of a key letter followed by option settings. For instance:

```
          .MARGIN( L5 R60 )
```

sets the left margin to 5 and the right to 60. Each time one of these four directives is processed, Prose saves the new values in a keep buffer. There are ten keep buffers (numbered 0 through 9) associated with each of these directives. A keep parameter may be used to specify which buffer to use, but if not specified, the values are saved in the numerically next buffer.

Old values may be recalled by using the following form:

```
          .directivename number
```

For example:

```
          .MARGIN 5
```

sets the margins to the values that were stored in keep buffer 5.

If no parameter is specified, the values are set to those that were stored in the numerically previous keep buffer. Since the keep number is automatically incremented when the parenthesis form is used and automatically decremented when no parameter is given, the keep buffers can be used as a stack.

```
          .MARGIN( L0 R70 )

             ...

          .MARGIN( L10 R60 )

             ...

          .MARGIN
```

In the previous example, the last MARGIN directive resets the margins to their previous values: left 0 and right 70.

Short Directive Table

Directive	Meaning (action)	Break	Parameter type
BREAK	break justification	*	-none-
COMMENT	no action		remainder of line
COUNT	set page count		numeric
FORM	define page format	*	(...)
INDENT	indent following line	*	numeric
INPUT	set input parameters	*	(...) or numeric
INX	store index entry		remainder of line
LITERAL	print literal text		remainder of line
MARGIN	set margins	*	(...) or numeric
OPTION	set options	*	(...) or numeric
OUTPUT	set output parameters		(...)
PAGE	eject to top of page	*	numeric
PARAGRAPH	set paragraphing params		(...) or numeric
RESET	reset directive defaults	*	(...)
SELECT	select pages to print	*	(...)
SKIP	skip output lines	*	numeric
SORTINDEX	sort and print index	*	(...)
SUBTITLE	set the subtitle		remainder of line
TITLE	set the main title		remainder of line
UNDENT	undent following line	*	numeric
WEOS	write end of section	*	-none-

The directives marked with an asterisk (*) cause a justification break before they are processed, since they affect the filling and justifying environment.

(...) indicates that the parameter is enclosed in parentheses and is described in detail along with the description of the directive itself.

BREAK

Causes a justification break.

COMMENT

Prose treats the remainder of the directive line as a comment, i.e. it is ignored. The COMMENT directive allows you to include in the source of your document information that will not be printed on the formatted copy.

COUNT number
COUNT

Sets the page counter. The numeric parameter can be relative. For example, ".COUNT +1" increments the page number by one. In the absence of a parameter the default is to set the page number to one.

FORM (parameters)
FORM

Defines the page format, including titles, footers, date/time, and the top and bottom of the page. The argument consists of parameters, followed by (if appropriate) an optional field width. For example "T:30" prints the title in a field of 30 characters. Text lines are built by the FORM directive from left to right, starting in the first printable column, although the tabbing specification may be used to alter that. The following table describes the FORM specifications that are available.

key char	meaning		default field width
C	24 hour clock as hh.mm.ss (15.37.58)		8
D	raw date as yy/mm/dd (78/02/13)		8
E	nice date as dd Mmm yy (13 Feb 78)		9
Ln	fill in n lines of running text		
Pf	current page number, f selects the form		3
	N or n	arabic numerals (default)	[the field
	L	upper case letter	width will
	l	lower case letter	be expanded
	R	upper case roman numerals	if needed]
	r	lower case roman numerals	
S	subtitle		its length
T	main title		its length
W	wall clock as hh:mm AM (3:37 PM) or hh:mm PM		8
#n	tab forward or backward to absolute column n		
"..."	print literal text		
'...'	print literal text		
/	print an end of line		
/n	print -n- ends of lines		

```
[       define top of page
]       define bottom of page
```

default form:

```
.FORM( [ // T #62 E /// L56 // #33 '- ' PN:1 ' -' /// ] )
```

The FORM directive is processed interpretively. This means that the format is re-scanned as each page of output is produced, so changing one of the title buffers with the TITLE or SUBTITLE directives will change the title or subtitle on the next page.

The top of page definition is used for several things. By using the OUTPUT directive, you can request Prose to send a page eject to the output device when it reaches the top of a page. You can also request Prose to pause at the top of each page to allow you to change paper. At the end of the document, Prose does one last page eject, interpreting the FORM specification until it reaches the top of page.

The bottom of page specification is where Prose increments the page number, so if you print the page number both before and after the bottom of page definition, you will get two different numbers.

It is easy (once you understand the FORM directive) to produce fancy page formats. For example, you can design a FORM that will print the page number at the right of odd numbered pages, and at the left of even pages. This is done with a FORM that defines two pages with two "["s and two "]"s:

```
.FORM( [ // T #62 E /// L56 // #63 'PAGE' P /// ]
+      [ // T #62 E /// L56 //     'PAGE' P /// ] )
```

In the absence of a parameter, no special page formatting is done. This is similar to a FORM consisting of a single L specification defining an infinite number of lines per page. In this mode, the PAGE directive acts as though there are 5 lines left on the page.

INDENT number
INDENT

Indents the following line by a certain number of spaces. In the absence of a parameter, the default is 5.

INPUT (parameters)
INPUT number
INPUT

The INPUT directive is used to define the input environment, that is, the interpretation of characters on the input file. The parameters can be given in any order, and consist of a key letter followed by a value. The following table summarizes the parameters.

key letter	meaning	type	default	relative
B	explicit blank character	character	nul	
C	case shift character	character	nul	
D	directive escape character	character	.	
H	hyphenation character	character	nul	
K	keep	number	next	no
U	underline character	character	nul	
W	input width	number	150	no

If a specification is not given, its value is not changed. The default value is the one that will be set if the key letter is given by itself, and is also the value that is assigned when Prose begins processing.

B: The explicit blank character indicates a blank that Prose should not tamper with. Thus, if the cross hatch (#) is specified as the explicit blank:

```
.INPUT( B# )
```

then two words that are separated by an explicit blank:

```
Mr.#Smith
```

will never be split from one line to the next, and Prose will never fill blanks in between the words to justify a line.

C: The case shift character must be used to create mixed-case output from upper-case-only input. When a case shift character is specified, Prose automatically shifts all upper case letters to lower case. To specify an upper case letter, one of two methods may be used. The first method is to surround letters with the case shift characters, causing a shift-up and shift-down. Since most upper case letters are at the beginning of a word (following a blank), the second method, called stuttering, is to double the first character of the word. The following example demonstrates the production of mixed-case output from upper-case-only input.

Input to Prose:

```
.INPUT( C^ )
TTHE MMOCK TTURTLE WENT ON.
     "^W^E HAD THE BEST OF EDUCATIONS--IN FACT, WE WENT TO
SCHOOL EVERY DAY--"
     ^"I'VE^ BEEN TO A DAY-SCHOOL, TOO," SAID AALICE.  "^Y^OU
NEEDN'T BE SO PROUD AS ALL THAT."
     "^W^ITH EXTRAS?" ASKED THE MMOCK TTURTLE, A
LITTLE ANXIOUSLY.
     "^Y^ES," SAID AALICE: "WE LEARNED FFRENCH AND MUSIC."
     "^A^ND WASHING?" SAID THE MMOCK TTURTLE.
     "^C^ERTAINLY NOT" SAID AALICE, INDIGNANTLY.
     "^A^H TTHEN YOURS WASN'T A REALLY GOOD SCHOOL," SAID THE
MMOCK  TTURTLE  IN A TONE OF GREAT RELIEF.  "^N^OW, AT ^OURS^,
THEY
HAD, AT THE END OF THE BILL, '^F^RENCH, MUSIC, ^AND
WASHING--^ EXTRA.'"
```

Output from Prose:

```
The Mock Turtle went on.
     "We had the best of educations--in fact, we went  to
school every day--"
     "I'VE  been to a day-school, too," said Alice.  "You
needn't be so proud as all that."
     "with extras?"  asked  the  Mock  Turtle,  a  little
anxiously.
     "Yes," said Alice: "we learned French and music."
     "And washing?" said the Mock Turtle.
     "Certainly not" said Alice, indignantly.
     "An  Then  yours  wasn't a really good school," said
the Mock Turtle in a tone  of  great  relief.   "Now,  at
OURS,  they  had, at the end of the bill, 'French, music,
AND WASHING-- extra.'"
```

At first glance, the stuttering method may seem clumsy, but experience shows that it is reasonably easy to get used to. To enter words that already have a double letter at the beginning (like llama and oops), merely precede the word with two case shift characters, causing a shift-up/shift-down (^^LLAMA and ^^OOPS). Keep in mind that the case shift character does not need to be used unless you want to create mixed-case output from upper-case-only input. It is recommended that if possible, you use mixed-case input to create mixed-case output.

D: The directive escape character is the character you type in the first column of an input line to flag it as a directive line.

H: The hyphenation character is used to define hyphenation points within words. Sometimes a long word will cause many blanks to be inserted to justify the preceding line. Prose will hyphenate such a word if you have defined the syllable boundries within that word. Of course, not all the syllable boundries need be specified, only those where you want Prose to be able to split a word. For example, if the hyphenation character is set to the slash (/), you might type "syncopation" as "syn/co/pa/tion". Prose will insert a hyphen (-) only when the characters on both sides of the hyphenation point are letters. You might type "hyper-active" as "hyper-/active", and Prose will split the word, if necessary, without adding a superfluous hyphen. If Prose is forced to insert more blanks than a certain threshold (set with the OPTION directive), it will issue a message suggesting that you insert hyphenation characters.

K: The keep parameter explicitly specifies which keep buffer should be used to store the new input options. The default is to use the numerically next buffer.

U: Text surrounded by the underline character will be underlined. Blanks are not underlined, but explicit blanks are.

W: The input width is used to specify how many characters will be read from each input line. If your input lines have sequencing information at the right of each line, you will need to set the width to an appropriate value.

INX text

Enters the remainder of the line together with the current page number as an index entry. This means that as the formatted text migrates from page to page, the resulting index will always be correct.

LITERAL text

Prints the remainder of the line on the output file. The special processing for upper/lower case, underlining, and literal blanks is performed on the text of the parameter, and then it is printed as a single output line. This output line is printed independently of filling and justifying and page formatting processes; it is transparent to the usual Prose formatting and is not counted as an output line. The LITERAL directive is useful for producing special printer control characters. For example,

```
.LITERAL T
```

sets a print density of 8 lines per inch on some CDC line printers.

MARGIN (parameters)
MARGIN number
MARGIN

The margin directive is used to set the left and right margins for filling and justifying. The left margin is the number of leading spaces before the first printed character, and the right margin is the column number of the last printed character. Thus subtracting the left margin from the right margin gives the number of printed columns. The parameters may be given in any order, and consist of a key letter followed by a value. The following table lists the parameters.

key letter	meaning	type	default	relative allowed
K	keep	number	next	no
L	left margin	number	0	yes
R	right margin	number	70	yes

If a specification is not given, its value is not changed. The default value is the one that will be set if the key letter is given by itself, and is also the value that is assigned when Prose begins processing.

The keep parameter explicitly specifies which keep buffer should be used to store the new margins. The default is to use the numerically next buffer.

OPTION (parameters)
OPTION number
OPTION

All the miscellaneous options that affect the text formatting process are gathered together in the OPTION directive. These options are summarized in the following table. For switch options, "+" is on and "-" is off.

key letter	meaning	type	default	relative allowed
E	print error messages	switch	+	
F	fill output lines	switch	+	
J	justification limit	numeric	3	no
K	keep	numeric	next	no
L	left justify	switch	+	
M	multiple blanks	switch	+	
P	2 blanks after periods	switch	+	
R	right justify	switch	+	
S	spacing	numeric	1	no
U	shift to upper case	switch	-	

If a specification is not given, its value is not changed. The default value is the one that will be set if the key letter is given by itself, and is also the value that is assigned when Prose begins processing.

E: Error messages are printed on the main output file, interspersed in the formatted text. These may be entirely suppressed by setting the E option to "E-".

F: Output lines are automatically filled and justified as described in the section "Basic Units of Text". If the fill switch is turned off, Prose will print the input lines as they are, without reformatting to fill up the output lines. In effect, a justification break is done after each input line.

J: In justifying the left and right margins of an output line, Prose has to insert blanks that are not explicitly on the input file. The justification limit controls the point at which Prose will attempt to hyphenate a word. If, for instance, the justification limit is three, then the hyphenation process will be invoked when Prose inserts enough blanks to bring the number between any adjacent words to three. If hyphenation is not possible, or Prose is not able to bring the number of inserted blanks below the limit, an error message is printed.

K: The keep parameter explicitly specifies which keep buffer should be used to store the new options. The default is to use the numerically next buffer.

L:
R: The left and right justify switches work together to determine what kind of justification is done. If both switches are on, output lines are justified to both the left and right margins. If both switches are off, lines are centered between the two margins. If one is on and one is off, the result is one straight margin (either left or right) and one ragged margin. The following demonstrates these four options.

.OPTION(L+ R+) :

"You couldn't have wanted it much," said Alice; "living at the bottom of the sea."

"I couldn't afford to learn it," said the Mock Turtle with a sign. "I only took the regular course."

"What was that?" inquired Alice.

"Reeling and Writhing, of course, to begin with," the Mock Turtle replied; "and then the different branches of Arithmetic--Ambition, Distraction, Uglification, and Derision."

"I never heard of 'Uglification,'" Alice ventured to say. "What is it?"

The Gryphon lifted up both its paws in surprise. "Never heard of uglifying!" it exclaimed. "You know what to beautify is, I suppose?"

.OPTION(L- R-) :

"Yes," said Alice doubtfully: "it
means--to--make--anything--prettier."

"Well, then," the Gryphon went on, "if you don't know
what to uglify is, you are a simpleton."

Alice did not feel encouraged to ask any more questions
about it: so she turned to the Mock Turtle, and said
"What else had you to learn?"

"Well, there was Mystery," the Mock Turtle replied,
counting off the subjects on his flappers--"Mystery,
ancient and modern, with Seaography: then Drawling--the
Drawling-master was an old conger-eel, that used to come
once a week: he taught us Drawling, Stretching, and
Fainting in Coils."

.OPTION(L+ R-) :

"What was that like?" said Alice.

"Well, I ca'n't show it you, myself," the Mock Turtle said "I'm too stiff. And the Gryphon never learnt it."

"Hadn't time," said the Gryphon: "I went to the Classical master, though. He was an old crab, he was."

"I never went to him," the Mock Turtle said with a sign. "He taught Laughing and Grief, they used to say."

"So he did, so he did," said the Gryphon, sighing in turn; and both creatures hid their faces in their paws.

"And now many hours a day did you do lessons?" said Alice, in a hurry to change the subject.

.OPTION(L- R+)

"Ten hours the first day," said the Mock Turtle: "nine
the next, and so on."

"What a curious plan!" exclaimed Alice.

"That's the reason they're called lessons," the Gryphon
remarked: "because they lessen from day to day."

This was quite a new idea to Alice, and she thought it
over a little before she made her next remark. "Then the
eleventh day must have been a holiday?"

"Of course it was," said the Mock Turtle.

"And now did you manage on the twelfth?" Alice went on
eagerly.

"That's enough about lessons," the Gryphon interrupted in
a very decided tone. "Tell her something about the games
now."

M: If the multiple blanks switch is on, multiple blanks on the input file are considered to be significant. That is, if there are several blanks between two words on the input file, there will be at least that many on the output file, but Prose may add more blanks during the justification process. If the switch is off, multiple blanks will be changed into a single blank.

P: If the 2 blanks after periods option is selected, then Prose will make sure that each period which is already followed by at least one blank will be followed by at least two blanks. Prose will not add blanks before justifying if there are already two. This makes it easy to have sentences separated by two blanks without requiring you to be extremely careful about typing the original text.

S: By setting the spacing option, you can easily produce single, double, or triple spaced output. Simply set the spacing option to 1, 2, or 3.

U: Since some output devices are not able to handle mixed-case files, you can cause Prose to shift all lower case letters to upper case by selecting the shift to upper case option. This is of particular interest to CDC users for whom lower case letters are interpreted as two characters when sent to certain output devices. This option is also handy for printing large sections, such as sample programs, all in upper case.

OUTPUT (terminal-type parameters)

The OUTPUT directive defines important aspects of the output device that is the destination of the formatted text. The OUTPUT directive may be used only once, and must appear before any lines are printed on the output device or immediately following the directive ".RESET(OUTPUT)".

Terminal-type may be one of the following; the default is ASC:

ASC ASCII terminal, using carriage return for overprinting and form feed for page eject. A teletype is called an ASC terminal although the form feed will not cause a page eject. This is not a problem if the eject option (see below) is not selected.

LPT Line printer, using "+" for overprinting and "1" for page eject. Carriage control is supplied automatically by Prose, and so like any other terminal, column 1 is the first printing column.

AJ Anderson/Jacobson terminal, using 1/60th of inch increments for justification. ASC may be specified for an AJ terminal, but the result will not have as high quality. If AJ is selected, however, the output will be printed more slowly. For this reason, it is recommended that ASC be used for drafts, and AJ only for the final version. The AJ may be followed by a number specifying the desired pitch (in characters per inch), e.g. "AJ 10".

The parameters define further characteristics of the output device, and several global output options. The parameters may be given in any order, and are selected from the following table.

key letter	meaning	type	default
E	page eject at top of page ("[" in FORM description)	switch	-
P	pause at top of page	switch	-
S	shift output lines to the right	numeric	0
U	underlining is available	switch	+

E: If the page eject option is selected, a form feed or "1" will be printed every time the "[" is encountered in the FORM specification.

P: If the pause option is selected, every time the "[" is encountered in the FORM specification, Prose will stop printing and wait for some operator acknowledgement. On an ASC or AJ terminal, Prose will sound the bell, and wait for a carriage return to be entered.

For an LPT terminal, the processing is dependent on the operating system. This option is handy for using an AJ terminal with non-fan-fold paper, allowing you to roll paper in for each page. For the CDC version, any single character (not just carriage return) will cause Prose to resume printing on an ASC or AJ terminal. For a CDC LPT terminal, Prose will print a PM message containing the Prose control statement.

S: All output that Prose produces can be shifted to the right by any number of spaces up to 50. This makes it easy to center the output on a wide printer page.

U: If the destination terminal does not have underlining ability and your input does underlining, the underlining available option should be turned off to prevent Prose from trying to generate overprinted underlines.

PAGE number
PAGE

Causes a page eject if there are fewer than the specified number of lines remaining on the current page. If no parameter is given, PAGE does an unconditional page eject.

PARAGRAPH (parameters)
PARAGRAPH number
PARAGRAPH

Paragraphs can be indicated by any of the methods introduced in the section "Basic Units of Text". The PARAGRAPH directive provides a more versatile method of creating paragraphs.

The PARAGRAPH directive specifies what is done when a new paragraph is signalled by typing a special character (called the paragraph flag character) in the first column of an input line. An automatic indent or undent can be selected, an automatic skip and/or automatic page eject can be specified, and you can even have Prose automatically number the paragraphs.

key letter	meaning	type	default	relative
F	paragraph character	character	nul	
I	automatic indent	number	0	no
K	keep	number	next	no
N	number generator		none	
P	automatic page eject	number	0	no
S	automatic skip	number	0	no
U	automatic undent	number	0	no

If a specification is not given, its value is not changed. The default value is the one that will be set if the key letter is given by itself, and is also the value that is assigned when Prose begins processing.

F: The paragraph flag character is used to invoke this collection of paragraphing actions by typing it in the first column of an input line. Note that this character should be set in at least one PARAGRAPH directive, or none of these actions will work.

I:
U: The automatic indent or automatic undent is applied to the first line of the paragraph (see the description of INDENT and UNDENT). If the number generator is used, the indent or undent is applied after the number is generated.

N: If the number generator is specified, a new number (or letter) will be generated for each occurrance of the paragraph flag character. The number generator is initialized to 1 each time new PARAGRAPH settings go into effect, but resuming an old setting will also resume the old numbering. The number replaces the paragraph flag character when the line is formatted. The number generator parameter has the form: Nfn .

f selects the numeric form:

-blank-	no numbering
N or n	arabic numerals
L	upper case letter
l	lower case letter
R	upper case roman
r	lower case roman

n is the field width, which will be expanded if needed.

P: The automatic page eject is used to simulate the effect of the directive

```
.PAGE number
```

before the first line of the paragraph. If this parameter is set to 4, for instance, it will ensure that at least four lines are left on the page. If there are fewer lines than specified, a page eject is done. This is applied after the automatic skip.

S: The automatic skip is done before the first line of the paragraph, and functions the same as a SKIP directive.

K: The keep parameter explicitly specifies which keep buffer should be used to store the new paragraph options. The default is to use the numerically next buffer.

RESET
RESET `(parameters)
RESET (EXCEPT parameters)

The RESET directive is used to set directives to their default values. If you have changed the values of many directives (such as FORM, MARGIN, or OPTION), the simple command

```
.RESET
```

resets the values of all directives to their defaults. Directives may be reset selectively by using the second form of the command. For example,

```
.RESET( MARGIN OPTION )
```

only resets the MARGIN and OPTION directives. Directives may also be excluded selectively. For example,

```
.RESET( EXCEPT FORM OUTPUT )
```

resets all directives with the exception of FORM and OUTPUT.

Parameters for RESET are selected from the following list of directive names.

COUNT	FORM	INPUT	INX
MARGIN	OPTION	OUTPUT	PAGE
PARAGRAPH	SELECT	SUBTITLE	TITLE

The values of parameters for most directives are set to their defaults (which are listed with the description of each directive) with the exception of the keep parameters which are set to "K0". For the COUNT, INX, and PAGE directives, however, the action is different. Resetting COUNT sets the page counter to 1, resetting INX deletes all index entries that have been accumulated, and resetting PAGE causes a page eject. In addition, since resetting FORM or OUTPUT directly affects the printed result, resetting either of these directives also causes a page eject.

SELECT (parameters)

As documentation is revised, not every page changes. The SELECT directive may be used to print only certain pages. The entire text will be formatted, but only selected pages will be printed. Thus the central processor time used will not be reduced very much, but printing time will be. The descriptor consists simply of page numbers separated by spaces. To select a span of pages, two numbers are typed together, separated by a colon (:). The second page number may be specified relative to the first. The following example selects pages 3, 5, 10 through 15, and 20 through 25 to be printed.

```
.SELECT( 3 5 10:15 20:+5 )
```

The default is to select all pages to be printed.

SKIP number
SKIP

Skips a certain number of output lines, i.e. prints blank lines. SKIP will never print blank lines at the top of a page, so to skip lines at the top of a page, at least one actual blank line must precede the SKIP directive. In the absence of a parameter, 5 lines are skipped.

SORTINDEX (parameters)
SORTINDEX

The index entries that are accumulated by INX directives can be sorted either alphabetically or by page number, and then printed in a fairly flexible manner. The SORTINDEX directive allows you to specify what column is to be considered the first significant column for alphabetical sorting, how many leading blanks to print at the left of each index line, where to insert the page number in each line, and how to format the page number. The parameters may be given in any order, and are selected from the following.

key letter	default	meaning
L	2	left width of page number (field width for number)
M	0	margin (left margin before index line)
P	0	column (in index entry) to insert page number
R	2	right width of page number (blanks printed after)
S	1	sorting option. if this is numeric, it is the first significant column for alphabetical sorting. if it is the letter "P", it selects sorting by page number.

In the absence of parameters, the defaults are used.

SUBTITLE text

Enters the remainder of the directive line into the subtitle buffer. The subtitle buffer is used by the FORM directive.

TITLE text

Enters the remainder of the directive line into the main title buffer. The title buffer is used by the FORM directive.

UNDENT number
UNDENT

Undents the following line a certain number of spaces. The undent is sometimes known by the name "outdent" or "hanging indent". A line can never be undented past the leftmost column of the printer page, and so a large number is adjusted to a smaller value. In the absence of a parameter, the default is to undent to the leftmost printable column.

WEOS

Write an end-of-section on the output file. This directive is useful for creating multiple section writeups under systems with utilities that manipulate multiple section files. In the CDC version of Prose, WEOS writes a CDC end-of-record mark. Specifically, this directive is used to create indexed writeups at the University of Minnesota.

CDC KRONOS and NOS

At the University of Minnesota, Prose is available through the PROSE control statement, which has three order-dependant file parameters. The prototype call (which includes the default file names) is:

```
PROSE(INFILE,OUTPUT,INPUT)
```

INFILE - file containing text in Prose form.
OUTPUT - file to receive the formatted result.
INPUT - file used for the pause option of the OUTPUT directive.

The following control statement will format a file named DOC and write the result to OUTPUT:

```
PROSE(DOC)
```

and to format the same file but write the result to LIST:

```
PROSE(DOC,LIST)
```

```
{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

          PROSE - A TEXT FORMATTING TOOL.
          J. P. STRAIT.     77/06/05.
          COPYRIGHT (C) 1977, 1979.
          ALL RIGHTS RESERVED.




          PROSE IS A FORMATTING PROGRAM, DESIGNED FOR DOCUMENT
PREPARATION.  IT IS WRITTEN IN PASCAL AND IS IMPLEMENTED IN SUCH
A WAY AS TO ENCOURAGE TRANSPORTATION BETWEEN DIFFERENT HARDWARE AND
DIFFERENT OPERATING SYSTEMS.

          PROSE WAS DEVELOPED IN THE SPRING OF 1977, AND DRAWS
VERY HEAVILY FROM TYPESET, A FORMATTING PROGRAM WRITTEN BY MICHAEL
HUCK.  TYPESET, WRITTEN IN COMPASS (THE CDC 6000/CYBER SERIES
ASSEMBLY LANGUAGE), WAS IN TURN BASED ON EDIT-RUNOFF.  THUS PROSE
IS ONE OF THE MANY DESCENDANTS OF RUNOFF.

          COMPLETE EXTERNAL DOCUMENTATION IS AVAILABLE,. AND IT IS
MAINTAINED IN PROSE FORM.  REFER TO THAT FOR AN OVERVIEW OF PROSE.

          IN STRIVING FOR PORTABILITY, THE DECISION WAS MADE TO
REPRESENT TEXT INTERNALLY IN ASCII.  THIS MEANS THAT TO TRANSPORT
THIS PROGRAM, ONE INPUT ROUTINE AND ONE OUTPUT ROUTINE MUST BE
REWRITTEN TO TRANSLATE BETWEEN THE HOST CHARACTER SET AND ASCII.
OTHER SYSTEM DEPENDANT DETAILS SUCH AS THE DATE AND CLOCK FUNCTIONS
MUST BE CONSIDERED FOR TRANSPORTATION OF THIS PROGRAM.  ALL AREAS
WHICH NEED ATTENTION WHEN CONVERTING THIS PROGRAM FROM THE CDC 6000
VERSION ARE MARKED WITH NULL COMMENTS IN COLUMNS 69-72.

          THIS VERSION OF PROSE READS AND WRITES THE CDC ASCII 63
CHARACTER SET.  IF YOU RUN SOME OTHER CHARACTER SET, YOU MAY WISH
TO CONVERT THE INPUT/OUTPUT ROUTINES TO PROCESS THAT CHARACTER SET.

  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }


program prose(infile,output+,input/);                              {  }



const

 infinity        = 1000;     {  LARGEST NUMBER + 1  }
 maintitle       = true;     {  TITLE INDICATOR  }
 maxinxlength    = 60;       {  MAX LENGTH OF INDEX ENTRIES  }
 maxiwidth       = 200;      {  MAXIMUN INPUT WIDTH  }
 maxkeep         = 9;        {  MAXIMUM KEEP VALUE  }
 maxmargin       = 200;      {  LARGEST RIGHT MARGIN  }
 maxnumberwidth  = 20;       {  MAX NUMBER OF DIGITS IN A NUMBER  }
 maxowidth       = 200;      {  MAXIMUM OUTPUT WIDTH  }
 maxpage         = 999;      {  MAXIMUM PAGE NUMBER  }
 maxshift        = 50;       {  MAX OUTPUT SHIFT  }
 maxskip         = 100;      {  MAXIMUM SKIP COUNT  }
 maxsplit        = 20;       {  MAXIMUM NUMBER OF SPLIT POINTS  }
 maxstringlength = 222;      {  MAX LENGTH OF TEXT LINES  }
 min             = 10;       {  GENERAL REASONABLE MIMIMUM  }
 subtitle        = false;    {  SUBTITLE INDICATOR  }

{  CERTAIN CONSTRAINTS ARE APPLIED TO THE MIN/MAX VALUES,       }
{  TO ELIMINATE ARRAY OVERFLOW CHECKS AND OTHER ERROR CHECKS:   }
{                                                               }
{       MAXSTRINGLENGTH >= MAXIWIDTH + MAXNUMBERWIDTH + 2       }
{       MAXMARGIN       <= MAXSTRINGLENGTH - 2                  }
{       (EVERYTHING)     <  INFINITY                            }
{       (EVERYTHING)     >  (REASONABLE)                        }

{  THE ASCII CHARACTER SET:  }

 nul =    0;       blank =   32;          at =   64;        grav =   96;
 soh =    1;     exclaim =   33;           a =   65;      smalla =   97;
 stx =    2;      dquote =   34;           b =   66;      smallb =   98;
 etx =    3;        hash =   35;           c =   67;      smallc =   99;
 eot =    4;      dollar =   36;           d =   68;      smalld =  100;
 enq =    5;     percent =   37;           e =   69;      smalle =  101;
 ack =    6;   ampersand =   38;           f =   70;      smallf =  102;
 bel =    7;      squote =   39;           g =   71;      smallg =  103;
  bs =    8;      lparen =   40;           h =   72;      smallh =  104;
  ht =    9;      rparen =   41;           i =   73;      smalli =  105;
  lf =   10;        star =   42;           j =   74;      smallj =  106;
  vt =   11;        plus =   43;           k =   75;      smallk =  107;
  ff =   12;       comma =   44;           l =   76;      smalll =  108;
  cr =   13;       minus =   45;           m =   77;      smallm =  109;
  so =   14;      period =   46;           n =   78;      smalln =  110;
  si =   15;       slash =   47;           o =   79;      smallo =  111;
 dle =   16;        zero =   48;           p =   80;      smallp =  112;
 dc1 =   17;         one =   49;           q =   81;      smallq =  113;
 dc2 =   18;         two =   50;           r =   82;      smallr =  114;
 dc3 =   19;       three =   51;           s =   83;      smalls =  115;
 dc4 =   20;        four =   52;           t =   84;      smallt =  116;
 nak =   21;        five =   53;           u =   85;      smallu =  117;
 syn =   22;         six =   54;           v =   86;      smallv =  118;
 etb =   23;       seven =   55;           w =   87;      smallw =  119;
 can =   24;       eight =   56;           x =   88;      smallx =  120;
  em =   25;        nine =   57;           y =   89;      smally =  121;
 sub =   26;       colon =   58;           z =   90;      smallz =  122;
 esc =   27;   semicolon =   59;    lbracket =   91;      lbrace =  123;
  fs =   28;        less =   60;   backslash =   92; verticalbar =  124;
  gs =   29;       equal =   61;    rbracket =   93;      rbrace =  125;
  rs =   30;     greater =   62;       caret =   94;       tilde =  126;
  us =   31;    question =   63;  underscore =   95;         del =  127;



 type

  ascii = 0.. 127;

asciix = 0.. 255;  {  THE TYPE ASCII IS EXTENDED FOR INTERNAL USE    }
                   {  IN THE FOLLOWING MANNER:                       }
                   {                                                 }
                   {  C + 200B      INDICATES THAT C IS UNDERLINED.  }

ascii2host = packed record                                                 {  }
 c              : char;      {  EXTERNAL CHARACTER  }                      {  }
 chr74          : boolean;   {  IF A 74 ESCAPE IS REQUIRED  }              {  }
 chr76          : boolean    {  IF A 76 ESCAPE IS REQUIRED  }              {  }
 end;                                                                      {  }

charclass = packed record
 digit          : boolean;   {  ZERO..NINE  }
 formchar       : boolean;   {  C,D,E,L,P,S,T,W,HASH,LBRACKET,RBRACKET,
                                SLASH,DQUOTE,SQUOTE,RPAREN,BLANK  }
 inputchar      : boolean;   {  B,C,D,H,K,U,W,BLANK  }
 letter         : boolean;   {  A..Z,SMALLA..SMALLZ  }
 marginchar     : boolean;   {  K,L,R,BLANK  }
 numform        : boolean;   {  N,SMALLN,L,SMALLL,R,SMALLR,BLANK  }
 optionchar     : boolean;   {  E,F,J,K,L,M,P,R,S,U,BLANK  }
 outputchar     : boolean;   {  E,P,S,U,W,BLANK  }
 paragraphch    : boolean;   {  C,I,K,N,P,U,BLANK  }
 plusorminus    : boolean;   {  PLUS,MINUS  }
 quote          : boolean;   {  DQUOTE,SQUOTE  }
 sortinxchar    : boolean;   {  L,M,P,R,S,BLANK  }
 end;

ch3 = packed array[1..3] of asciix;

ch10 = packed array[1..10] of asciix;

direct = (bre,               {  BREAK  }
          com,               {  COMMENT  }
          cou,               {  COUNT  }
          frm,               {  FORM  }
          ind,               {  INDENT  }
          inp,               {  INPUT  }
          inx,               {  INX  }
          lit,               {  LITERAL  }
          mar,               {  MARGIN  }
          opt,               {  OPTION  }
          out,               {  OUTPUT  }
          pag,               {  PAGE  }
          par,               {  PARAGRAPH  }
          res,               {  RESET  }
          sel,               {  SELECT  }
          ski,               {  SKIP  }
          sor,               {  SORTINDEX  }
          sbt,               {  SUBTITLE  }
          ttl,               {  TITLE  }
          und,               {  UNDENT  }
          weo,               {  WEOS  }
          exc,               {  EXCEPT (USED BY RESET)  }
          ill,               {  ILLEGAL  }

{  THE FOLLOWING ARE NOT DIRECTIVES, BUT IT IS CONVENIENT  }
{  TO INCLUDE THEM IN THIS TABLE.                          }

          ast,               {  ASCII TERMINAL  }
          lpt,               {  LINE PRINTER  }
          ajt,               {  ANDERSON/JACOBSON TERMINAL  }
          ilt);              {  ILLEGAL  }

 dirset = set of direct;

 inputsettings = packed record
  defined    : boolean;
  b,c,d,h,u  : ascii;
  w          : 0..infinity
  end;

 pinxentry = ↑inxentry;
 inxentry = record
  x    : packed array[1..maxinxlength] of asciix;
  xl   : integer;              {  LENGTH OF ENTRY  }
  xp   : integer;              {  PAGE NUMBER  }
  next : pinxentry
  end;

 marginsettings = packed record
  defined : boolean;
  l,r     : 0..infinity
  end;

 numberform = (numeric,upperalpha,loweralpha,upperroman,lowerroman,
               nonumbering);

 optionsettings = packed record
  defined         : boolean;
  e,f,l,m,p,r,u   : boolean;
  j,s             : 0..infinity
  end;

 paragraphsettings = packed record
  defined : boolean;
  c       : 0..infinity;
  f       : ascii;
  i       : -infinity..infinity;
  n       : numberform;
  p       : 0..infinity;
  s       : 0..infinity;
  w       : 0..infinity
  end;

 remember = 0..maxkeep;

 splitpoint = packed record
  point : 0..infinity;  {  POSITION OF SPLIT POINT WITHIN WORD  }
  inpnt : 0..infinity;  {  POSITION OF SPLIT POINT WITHIN INLINE  }
```

```
   hypnt : boolean   {  SPLIT POINT REPRESENTS POSSIBLE HYPHEN  }
   end;

  pstring = packed array[1..maxstringlength] of asciix;

  string = array[1..maxstringlength] of  {  STR[1].C ALWAYS = ' '  }
   packed record
    c   : asciix;      {  CHARACTER  }
    nbl : 0..infinity  {  IF C=' ', NUMBER OF BLANKS, ELSE CHARWIDTH  }
    end;



 var


  asc            : array[char] of ascii;                          {  }
                              {  CONVERT DISPLAY CODE TO ASCII  }  {  }
  asc74          : array[char] of ascii;                          {  }
                              {  CONVERT 74 ESCAPE CODE TO ASCII  }  {  }
  asc76          : array[char] of ascii;                          {  }
                              {  CONVERT 76 ESCAPE CODE TO ASCII  }  {  }
  badjustify     : integer;  {  J OPTION  }
  blankcount     : integer;  {  ACCUMULATED BLANK OUTPUT LINE COUNTER  }
  blankline      : boolean;  {  BLANK OUTPUT LINE INDICATOR  }
  carriagecontrol : ascii;   {  FOR LINE PRINTER OUTPUT  }
  casech         : ascii;    {  C INPUT  }
  class          : array[ascii] of charclass;
                             {  CHARACTER CLASSIFICATIONS  }
  charwidth      : integer;  {  CHAR WIDTH IN PRINTER UNITS  }
  dirch          : ascii;    {  D INPUT  }
  directline     : boolean;  {  INPUT LINE IS A DIRECTIVE  }
  directs        : array[direct] of ch3;
                             {  DIRECTIVE NAMES  }
  eject          : boolean;  {  E OUTPUT  }
  endofinput     : boolean;  {  INTERNAL EOF INDICATOR  }
  ensure2        : boolean;  {  P OPTION  }
  errornl        : integer;  {  ERROR IN NUMBER  }
  errorn2        : integer;  {  ERROR IN NUMBER  }
  errors         : boolean;  {  ERRORS IN THIS PROSE RUN  }
  errorsmall     : boolean;  {  NUMBER IS TOO SMALL  }
  errorl         : asciix;   {  ERROR TEXT  }
  error10        : ch10;     {  ERROR TEXT  }
  eol            : boolean;  {  INTERNAL EOLN INDICATOR  }
  explicitblank  : ascii;    {  B INPUT  }
  fill           : boolean;  {  F OPTION  }
  firsterror     : boolean;  {  FIRST ERROR ON THIS LINE  }
  form           : pstring;  {  FORM BUFFER  }
  formindex      : integer;  {  CURRENT FORM POSITION  }
  formlength     : integer;  {  FORM LENGTH  }
  formnext       : pstring;  {  FORM FOR NEXT PAGE  }
  formnlength    : integer;  {  LENGTH OF FORMNEXT  }
  gaps           : array[0..maxstringlength] of 1..maxstringlength;
                             {  POINTERS TO WORD GAPS  }
  host           : array[ascii] of ascii2host;                    {  }
                             {  CONVERT ASCII TO DISPLAY CODE  }    {  }
  hyphen         : ascii;    {  H OPTION  }
  inchar         : asciix;   {  CURRENT INPUT CHARACTER  }
  incolumn       : integer;  {  CURRENT INPUT COLUMN  }
  infile         : text;     {  PROSE SOURCE INPUT FILE  }
  inlength       : integer;  {  LENGTH OF CURRENT INPUT LINE  }
  inline         : string;   {  CURRENT INPUT LINE  }
  inwidth        : integer;  {  W INPUT  }
  inxbase        : pinxentry;{  BASE OF INDEX ENTRY LIST  }
  inxlast        : pinxentry;{  LAST INDEX ENTRY  }
  keepinp        : integer;  {  CURRENT INPUT KEEP BUFFER  }
  keepmar        : integer;  {  CURRENT MARGIN KEEP BUFFER  }
  keepopt        : integer;  {  CURRENT OPTION KEEP BUFFER  }
  keeppar        : integer;  {  CURRENT PARAGRAPH KEEP BUFFER  }
  leftjustify    : boolean;  {  L OPTION  }
  leftmargin     : integer;  {  L MARGIN  }
  linecount      : integer;  {  OUTPUT LINE COUNT (WITHIN PAGE)  }
  linenumber     : integer;  {  INPUT LINE COUNT (FOR ERROR MESSES)  }
  linenums       : boolean;  {  LINE NUMBERS EXIST ON INPUT FILE  }
  lockeddent     : integer;  {  I/U PARAGRAPH  }
  lowercase      : boolean;  {  FOR UPPER TO LOWER CASE CONVERSION  }
  lowerdir       : boolean;  {  LOWERCASE FLAG IN DIRECTIVES  }
  months         : array[1..12] of ch3;
                             {  MONTH NAMES  }
  moreonleft     : boolean;  {  INDICATOR FOR JUSTIFYING  }
  multipleblanks : boolean;  {  M OPTION  }
  nblanks        : integer;  {  BLANK COUNT ON INPUT  }
  nchars         : integer;  {  WIDTH OF OUTPUT LINE  }
  newinline      : boolean;  {  BEGIN INPUT LINE INDICATOR  }
  newoutline     : boolean;  {  BEGIN OUTPUT LINE INDICATOR  }
  newparagraph   : boolean;  {  BEGIN PARAGRAPH INDICATOR  }
  ngaps          : integer;  {  NUMBER OF WORD GAPS  }
  nicedate       : ch10;     {  DATE AS YY MMM DD  }
  nsplits        : integer;  {  NUMBER OF SPLIT POINTS IN WORD  }
  nwords         : integer;  {  NUMBER OF WORDS IN OUTPUT LINE  }
  numbering      : numberform;
                             {  N PARAGRAPH  }
  numberwidth    : integer;  {  N PARAGRAPH  }
  outlength      : integer;  {  LENGTH OF OUTPUT LINE  }
  outline        : string;   {  OUTPUT LINE  }
  outwidth       : integer;  {  W OUTPUT  }
  pagenumber     : integer;  {  CURRENT PAGE NUMBER  }
  parachar       : ascii;    {  F PARAGRAPH  }
  paracount      : integer;  {  PARAGRAPH COUNTER  }
  parapage       : integer;  {  P PARAGRAPH  }
  paraskip       : integer;  {  S PARAGRAPH  }
  pause          : boolean;  {  P OUTPUT  }
  printerrors    : boolean;  {  E OPTION  }
  rawclock       : ch10;     {  CLOCK TIME AS HH:MM:SS  }
  rawdate        : ch10;     {  DATE AS YY/MM/DD  }
  rightjustify   : boolean;  {  R OPTION  }
  rightmargin    : integer;  {  R MARGIN  }
  saveinp        : array[remember] of inputsettings;
                             {  INPUT STACK  }
  savemar        : array[remember] of marginsettings;
                             {  MARGIN STACK  }
  saveopt        : array[remember] of optionsettings;
                             {  OPTION STACK  }
  savepar        : array[remember] of paragraphsettings;
                             {  PARAGRAPH STACK  }
  selection      : packed array[0..maxpage] of boolean;
                             {  SELECT DIRECTIVE SETTING  }
  shift          : integer;  {  S OUTPUT  }
  shiftup        : boolean;  {  U OPTION  }
  space          : integer;  {  S OPTION  }
  splits         : array[1..maxsplit] of splitpoint;
                             {  SPLIT POINTS WITHIN WORD  }
  terminaltype   : direct;   {  OUTPUT TERMINAL TYPE  }
  text           : string;   {  FOR BUILDING FORM SPECIFICATIONS  }
  textindex      : integer;  {  CURRENT TEXT POSITION  }
  textlength     : integer;  {  LENGTH OF TEXT  }
  title          : array[boolean] of pstring;
                             {  TITLE AND SUBTITLE BUFFERS  }
  titlelength    : array[boolean] of integer;
                             {  TITLE AND SUBTITLE LENGTHS  }
  underavail     : boolean;  {  U OUTPUT  }
  underchar      : ascii;    {  U INPUT  }
  underlining    : boolean;  {  UNDERLINING FLAG  }
  underdir       : boolean;  {  UNDERLINING FLAG IN DIRECTIVES  }
  wallclock      : ch10;     {  CLOCK TIME AS HH:MM AM  }
  word           : string;   {  CURRENT WORD  }
  wordlength     : integer;  {  LENGTH OF WORD  }











 procedure error( n : integer ); forward;
 procedure validate( var num : integer;
                     min,max,err : integer ); forward;
 procedure reinitialize( which : dirset ); forward;








{ ---------------------------------------------------------------- }
{                                                                  }
{                        GENERAL UTILITY                           }
{                        ------- -------                           }
{                                                                  }
{ ---------------------------------------------------------------- }




{         ASCIICHAR - CONVERT LITERAL HOST CHARACTER TO ASCII.
 }

 function asciichar( ch : char ) : ascii;
 begin {  ASCIICHAR  }
 asciichar := asc[ch]                                              {  }
 end {  ASCIICHAR  };




{         UPPER - CONVERT ALPHABETIC CHARACTERS TO UPPER CASE.
 }

 function upper( ch : asciix ) : asciix;
 begin {  UPPER  }
 if class[ch].letter
  then if ch >= smalla
   then upper := ch -  32
   else upper := ch
  else upper := ch
 end {  UPPER  };




{         LOWER - CONVERT TO LOWER CASE IF ALPHABETIC.
 }

 function lower( ch : asciix ) : asciix;
 begin {  LOWER  }
 if class[ch].letter
  then if ch <= z
   then lower := ch +  32
   else lower := ch
  else lower := ch
 end {  LOWER  };




{         NUMFORM - DETERMINE THE NUMERIC FORM.
 *
 *        PARAM  CH = N, SMALLN, L, SMALLL, R, SMALLR.
 *               ERR = ERROR IF BAD NUMERIC FORM.
 }

 function numform( ch : ascii; err : integer ) : numberform;
 begin {  NUMFORM  }
 if class[ch].numform
```

```
  then case ch of
   n,
   smalln : numform := numeric;
   l      : numform := upperalpha;
   smalll : numform := loweralpha;
   r      : numform := upperroman;
   smallr : numform := lowerroman;
   blank  : numform := nonumbering
   end
  else begin errorl := ch; error(err); numform := numeric end
 end {  NUMFORM  };




{          CONVERTNUMBER - CONVERT NUMBER FROM BINARY TO TEXT.
 *
 *          PARAM  STR - OUTPUT STRING.
 *                 LEN - LENGTH OF OUTPUT STRING.
 *                 NUM - NUMBER TO CONVERT.
 *                 FW  - FIELD WIDTH OF NUMBER.
 *                 FORM- FORM OF CONVERSION.
  }

 procedure convertnumber( var str : string; var len : integer;
                          num,fw : integer; form : numberform );
 var
  digit          : array[1..maxnumberwidth] of ascii;
                           {  DIGIT ARRAY  }
  nextnum        : integer;  {  FOR DECOMPOSITION  }
  x1,x2          : integer;  {  LOOP INDECES  }




  {         SEND1 - SEND ONE DIGIT.
   *
   *         PARAM  DIG - DIGIT TO SEND.
    }

  procedure send1( dig : ascii );
  begin {  SEND1  }
  if x1 < maxnumberwidth
   then begin x1 := x1 + 1;
    digit[x1] := dig
    end
  end {  SEND1  };




 begin {  CONVERTNUMBER  }
 x1 := 0;
 case form of
   numeric     : repeat nextnum := num div 10;
                  send1(num - 10 * nextnum + zero);
                  num := nextnum
                 until num = 0;
   loweralpha,
   upperalpha  : repeat num := num - 1;
                  nextnum := num div 26;
                  send1(num - 26 * nextnum + a);
                  num := nextnum
                 until num = 0;
   lowerroman,
   upperroman  : begin while num >= 1000 do
                  begin send1(m); num := num - 1000 end;
                 if num >= 900
                  then begin send1(d); send1(m); num := num - 900 end
                  else if num >= 500
                   then begin send1(d); num := num - 500 end
                   else if num >= 400
                    then begin send1(c); send1(d); num := num - 400 end;
                 while num >= 100 do
                  begin send1(c); num := num - 100 end;
                 if num >= 90
                  then begin send1(x); send1(c); num := num - 90 end
                  else if num >= 50
                   then begin send1(l); num := num - 50 end
                   else if num >= 40
                    then begin send1(x); send1(l); num := num - 40 end;
                 while num >= 10 do
                  begin send1(x); num := num - 10 end;
                 if num >= 9
                  then begin send1(i); send1(x); num := num - 9 end
                  else if num >= 5
                   then begin send1(v); num := num - 5 end
                   else if num >= 4
                    then begin send1(i); send1(v); num := num - 4 end;
                 while num >= 1 do
                  begin send1(i); num := num - 1 end
                 end;
   nonumbering:
  end;
 if len + fw > maxstringlength then fw := maxstringlength - len;
 for x2 := x1+1 to fw do
  begin len := len + 1;
  with str[len] do
   begin c := blank;
   nbl := charwidth
   end
  end;
 if len + x1 > maxstringlength then x1 := maxstringlength - len;
 if form in [numeric,loweralpha,upperalpha]
  then for x2 := x1 downto 1 do
   begin len := len + 1;
   with str[len] do
    begin if form = loweralpha
     then c := digit[x2] +  32
     else c := digit[x2];
    nbl := charwidth
    end
   end
  else for x2 := 1 to x1 do
   begin len := len + 1;
   with str[len] do
    begin if form = lowerroman
     then c := digit[x2] +  32
     else c := digit[x2];
    nbl := charwidth
    end
   end
 end {  CONVERTNUMBER  };




{          SHIFTSTRING - CONVERT STRING TO UPPER/LOWER CASE,
 *                       CONSIDERING STUTTERING AND CASE SHIFT.
  }

 procedure shiftstring( var str : string; var len : integer;
                        var lcs : boolean );
 var
  intch          : ascii;    {  INTERNAL CHARACTER  }
  oldch          : ascii;    {  PREVIOUS INTERNAL CHARACTER  }
  oldoldch       : ascii;    {  PREVIOUS PREVIOUS CHARACTER  }
  x1,x2          : integer;  {  LOOP INDICES  }
 begin {  SHIFTSTRING  }
 oldch := blank;
 oldoldch := blank;
 x1 := 0;
 x2 := 1;
 if len >= 1
  then if str[1].c = parachar
   then begin x1 := 1; x2 := 2 end;
 for x2 := x2 to len do
  begin intch := lower(str[x2].c);
  if intch = casech
   then lcs := not lcs
   else if intch = oldch
    then if (oldoldch = blank) and class[intch].letter
     then begin str[x1].c := upper(intch);
      lcs := true
      end
     else begin x1 := x1 + 1;
      if lcs
       then str[x1].c := intch
       else str[x1].c := upper(intch)
      end
    else begin x1 := x1 + 1;
     if lcs
      then str[x1].c := intch
      else str[x1].c := upper(intch)
     end;
   oldoldch := oldch;
   oldch := intch
   end;
  len := x1
  end {  SHIFTSTRING  };




 {          UNDERSTRING - SET UNDERLINED CHARACTERS IN STRING,
  *                       CONSIDERING UNDERLINE CHARACTER.
  *                       THIS IS ALSO DONE IN READPSTRING.
   }

  procedure understring( var str : string; var len : integer;
                         var uln : boolean );
  var
   intch          : ascii;    {  INTERNAL CHARACTER  }
   x1,x2          : integer;  {  LOOP INDICES  }
  begin {  UNDERSTRING  }
  x1 := 0;
  for x2 := 1 to len do
   begin intch := str[x2].c;
   if intch = underchar
    then uln := not uln
    else begin x1 := x1 + 1;
     if (intch <> blank) and uln
      then str[x1].c := intch +  128
      else str[x1].c := intch
     end
   end;
  len := x1
  end {  UNDERSTRING  };




  {          JUSTIFY - LEFT JUSTIFY, RIGHT JUSTIFY, AND/OR CENTER
   *                   AN OUTPUT LINE.
    }

   procedure justify;
   const
    floor          = 0.0;      {  MAKES TRUNC DO FLOOR  }
    cieling        = 0.9999;   {  MAKES TRUNC DO CIELING  }
   var
    fc             : real;     {  TO SELECT FLOOR OR CIELING  }
    ib             : integer;  {  INSERT BLANKS  }
    nb             : integer;  {  NUMBER BLANKS (TOTAL)  }
    ng             : integer;  {  NUMBER GAPS (ACTUAL)  }
   begin {  JUSTIFY  }
   ng := ngaps - 1;
   nb := (rightmargin - nchars) * charwidth;
   if leftjustify
    then begin if rightjustify
   then begin if moreonleft
     then fc := floor
     else fc := cieling;
    for ng := ng downto 1 do
     begin ib := trunc(fc + nb / ng);
     with outline[gaps[ng]] do nbl := nbl + ib;
     nb := nb - ib
     end
    end
  end
 else with outline[gaps[0]] do
  if rightjustify
   then nbl := nbl + nb
   else nbl := nbl + trunc(nb / 2);
moreonleft := not moreonleft
end {  JUSTIFY  };








{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -   }
{                                                                       }
{                               OUTPUT                                  }
{                               ------                                  }
{                                                                       }
{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -   }




{         WRITE1 - WRITE ONE CHARACTER, DO CONVERSION FROM ASCII
 *                 TO THE HOST CHARACTER SET.
 *
 *        PARAM  CH = CHARACTER TO WRITE.
  }

procedure write1( ch : asciix );
begin {  WRITE1  }
with host[ch mod  128] do                                           { }
 begin if chr74                                                     { }
  then write(chr( 60))                                              { }
  else if chr76                                                     { }
   then write(chr( 62));                                            { }
 write(c)                                                           { }
 end                                                                { }
end {  WRITE1  };




{         ENDLINE - TERMINATE AND COUNT AN OUTPUT LINE.
  }

 procedure endline;
 begin {  ENDLINE  }
 if selection[pagenumber]
  then if blankline
   then blankcount := blankcount + 1
   else writeln;
 if linecount <> infinity then linecount := linecount - 1
 end {  ENDLINE  };




{         WRITEBLANKLINES - WRITE ACCUMULATED BLANK LINES.
  }

 procedure writeblanklines;
 begin {  WRITEBLANKLINES  }
 blankline := false;
 if terminaltype = lpt
  then while blankcount >= 2 do
   begin if selection[pagenumber] then write('0');
   blankcount := blankcount - 2;
   if linecount <> infinity then linecount := linecount + 1;
   endline
   end;
 while blankcount > 0 do
  begin blankcount := blankcount - 1;
  if linecount <> infinity then linecount := linecount + 1;
  endline
  end
 end {  WRITEBLANKLINES  };




{         WRITESTRING - WRITE A STRING TO THE OUTPUT FILE.
 *
 *        PARAM  STR = STRING TO WRITE.
 *               LEN = LENGTH OF STR.
  }

 procedure writestring( var str : string; len : integer );
 var
  x1,x2,x3        : integer;  {  GENERAL INDEX VARIABLES  }
  understr        : string;   {  UNDERLINING FOR THIS STRING  }
  lunderchar      : ascii;    {  LOCAL UNDERCHAR  }
 begin {  WRITESTRING  }
 if selection[pagenumber]
  then begin while (str[len].c = blank) and (len > 1) do
    len := len - 1;
   if str[len].c = blank then len := 0;
   blankline := (len = 0) and (carriagecontrol = blank),
   if not blankline
    then begin writeblanklines;
     if underchar <> nul
      then begin x2 := 0;
       for x1 := 1 to len do with str[x1] do
        if odd(c div  128)
         then begin understr[x1].c := underscore;
          understr[x1].nbl := charwidth;
          c := c -  128;
          x2 := x1
          end
         else begin understr[x1].c := blank;
          understr[x1].nbl := nbl
          end;
       if (x2 <> 0) and underavail
        then begin lunderchar := underchar;
         underchar := nul;
         writestring(understr,x2);
         underchar := lunderchar;
         case terminaltype of
          ajt,
          ast : write1(cr);
          lpt : begin writeln; carriagecontrol := plus end
          end
         end
       end;
     str[1].nbl := str[1].nbl + shift;
     if terminaltype = lpt then write1(carriagecontrol);
     if explicitblank <> nul
      then for x1 := 1 to len do with str[x1] do
       if c = explicitblank
        then begin c := blank; nbl := charwidth end;
     if shiftup
      then for x1 := 1 to len do
       str[x1].c := upper(str[x1].c);
     if terminaltype = ajt
      then begin x2 := 0;
       for x1 := 1 to len do
       with str[x1] do
        if c <> blank
         then begin if x2 <> 0
           then begin x3 := x2 div charwidth;
            if (x2 mod charwidth = 0) and (x3 < 5)
             then for x3 := 1 to x3 do write1(blank)
             else begin write1(esc); write1(x);
              write1(x2 div 100 + zero);
              write1(x2 div 10 mod 10 + zero);
              write1(x2 mod 10 + zero)
              end
            end;
          x2 := 0;
          write1(c)
          end
         else x2 := x2 + nbl
       end
      else for x1 := 1 to len do
       with str[x1] do
        if c = blank
         then for x2 := 1 to nbl do
          write1(blank)
         else write1(c);
     carriagecontrol := blank;
     str[1].nbl := str[1].nbl - shift
     end
   end
  else blankline := false
 end {  WRITESTRING  };




{         ADVANCEFORM - ADVANCE FORM TO NEXT L SPECIFICATION.
  }

 procedure advanceform;
 var
  ch              : ascii;    {  KEY CHARACTER  }
  formch          : asciix;   {  CURRENT FORM CHARACTER  }
  fw              : integer;  {  FIELD WIDTH OF CURRENT ITEM  }
  tl              : integer;  {  LOCAL TITLE LENGTH  }
  which           : boolean;  {  WHICH TITLE (MAIN,SUB)  }
  x1              : integer;  {  GENERAL INDEX  }




 {         NEXTCH - ADVANCE TO NEXT FORM CHARACTER.
   }

  procedure nextch;
  begin {  NEXTCH  }
  formindex := (formindex mod formlength) + 1;
  formch := form[formindex]
  end {  NEXTCH  };




 {         NUMBER - READ A NUMBER FROM THE FORM.
  *
  *        PARAM  DEF = DEFAULT NUMBER.
   }

  function number( def : integer ) : integer;
  var
   num            : integer;  {  NUMBER BEGIN BUILT  }
  begin {  NUMBER  }
  if class[formch].digit
   then begin num := 0;
    repeat num := num * 10 + formch - zero;
     if num >= infinity then num := infinity-1;
   nextch
  until not class[formch].digit;
  number := num
  end
 else number := def
end {  NUMBER  };




{        FIELDWIDTH - READ OPTIONAL FIELD WIDTH SPECIFICATION.
 *
 *        PARAM  DEF = DEFAULT FIELD WIDTH.
 *               MIN = MINIMUM FIELD WIDTH.
  }

procedure fieldwidth( def,min : integer );
begin {  FIELDWIDTH  }
fw := def;
if formch = colon
 then begin nextch;
  fw := number(def)
  end;
if fw < min then fw := min
end {  FIELDWIDTH  };




{        SEND1 - SEND ONE CHARACTER TO THE TEXT LINE.
 *
 *        PARAM  CH = CHARACTER TO BE SENT.
  }

procedure send1( ch : asciix );
begin {  SEND1  }
textindex := textindex + 1;
if textindex + shift > maxowidth
 then begin textindex := 1; error(-1) end;
text[textindex].c := ch;
text[textindex].nbl := charwidth;
if textindex > textlength then textlength := textindex
end {  SEND1  };




{        SEND10 - SEND UP TO 10 CHARACTERS TO THE TEXT LINE,
 *                DETERMINING FIELD WIDTH.
 *
 *        PARAM  CH = 10 CHARACTERS.
 *               DEF = DEFAULT FIELD WIDTH.
 *               MIN = MINIMUM FIELD WIDTH.
  }

procedure send10( ch : ch10; def,min : integer );
var
 x1              : integer;  {  INDEX INTO CH  }
begin {  SEND10  }
fieldwidth(def,min);
if fw < def
 then {  SEND RIGHTMOST FW CHARACTERS  }
  for x1 := def-fw+1 to def do send1(ch[x1])
 else {  SEND LEADING BLANKS AND ALL DEF CHARACTERS  }
  begin for x1 := 1 to fw-def do send1(blank);
  for x1 := 1 to def do send1(ch[x1])
  end
end {  SEND10  };




{        WRITETEXT - WRITE TEXT BUFFER.
  }

procedure writetext;
begin {  WRITETEXT  }
writestring(text,textlength);
endline;
textlength := 1;
textindex := 1
end {  WRITETEXT  };




{        WAIT - WAIT FOR OPERATOR ACKNOWLEDGEMENT.
 *               HEAVILY SYSTEM DEPENDANT.
  }

procedure wait;                                                   {  }
type ch80       = packed array[1..80] of char;                    {  }
var  cs        : ch80;      {  CURRENT CONTROL STATEMENT  }       {  }
                                                                  {  }
 procedure csimage( var cs : ch80 ); extern;                      {  }
                                                                  {  }
begin {  WAIT  }                                                  {  }
if terminaltype = lpt                                             {  }
 then begin csimage(cs);                                          {  }
  writeln('PM ',cs)                                               {  }
  end                                                             {  }
 else begin writel(bel);                                          {  }
  writeln(chr(0),chr( 11));                                       {  }
  writeln(chr(0),chr(  6),chr(0),chr(  1));                       {  }
  readln                                                          {  }
  end                                                             {  }
end {  WAIT  };
```

```

begin {  ADVANCEFORM  }
ch := upper(form[formindex]);
if not class[ch].quote then nextch;
if class[ch].formchar
 then case ch of
  c : send10(rawclock,8,0);
  d : send10(rawdate,8,0);
  e : send10(nicedate,9,0);
  l : begin if textlength > 1 then writetext;
      linecount := number(1)
      end;
  p : begin if (formch = colon) or (formch = blank)
       then ch := n
       else begin ch := formch; nextch end;
      fieldwidth(3,0);
      convertnumber(text,textindex,pagenumber,fw,numform(ch,-4));
      if textindex > textlength then textlength := textindex
      end;
  s,
  t : begin which := (ch = t) or (ch = smallt);
      tl := titlelength[which];
      fieldwidth(tl,0);
      if fw < tl
       then {  SEND LAST FW CHARACTERS  }
        for x1 := tl-fw+1 to tl do send1(title[which][x1])
       else {  SEND LEADING BLANKS AND ALL TL CHARACTERS  }
        begin for x1 := 1 to fw-tl do send1(blank);
        for x1 := 1 to tl do send1(title[which][x1])
        end
      end;
  w : send10(wallclock,8,0);
  hash : begin x1 := number(1);
      while textindex < x1 do send1(blank);
      textindex := x1
      end;
  lbracket : begin if textlength > 1 then writetext;
      if selection[pagenumber]
       then begin if eject
         then begin blankcount := 0;
          if terminaltype = lpt
           then carriagecontrol := one
           else writel(ff)
          end
         else if terminaltype <> lpt
          then writeblanklines;
        if pause then wait
        end;
      if formnlength > 0
       then begin form := formnext;
        formlength := formnlength;
        formindex := 0;
        repeat nextch until formch = lbracket;
        nextch;
        formnlength := 0
        end
      end;
  rbracket : begin if textlength > 1 then writetext;
      pagenumber := pagenumber + 1;
      validate(pagenumber,0,infinity-1,-3)
      end;
  slash : for x1 := 1 to number(1) do writetext;
  dquote,
  squote : repeat nextch;
       while formch <> ch do
        begin send1(formch);
        nextch
        end;
       nextch;
       if formch = ch then send1(ch)
      until formch <> ch;
  blank :
  end
 else begin error1 := ch; error(-2) end
end {  ADVANCEFORM  };





{        BEGINLINE - BEGIN OUTPUT LINE, ADVANCE FORM AS NECESSARY.
 }

procedure beginline;
var
 fix            : integer;  {  LOCAL COPY OF FORMINDEX  }
 fnl            : integer;  {  LOCAL COPY OF FORMNLENGTH  }
begin {  BEGINLINE  }
if linecount <= 0
 then {  MAKE LINECOUNT > 0  }
  begin fix := formindex;
   fnl := formnlength;
   repeat
    if fnl <> formnlength
     then begin fix := formindex;
      fnl := formnlength
      end;
    advanceform
   until (linecount > 0) or ((fix = formindex) and (fnl = 0));
   if linecount <= 0
    then {  BAD FORM  }
     begin error(-5);
      linecount := infinity
      end
   end;
blankline := true
end {  BEGINLINE  };
```

```
{         WRITENULL - WRITE A NULL LINE.
 }

procedure writenull;
begin {  WRITENULL  }
beginline;
writestring(outline,1);
endline
end {  WRITENULL  };




{          SKIP - SKIP OUTPUT LINES.
 }

procedure skip( n : integer );
var x1 : integer;
begin {  SKIP  }
if n > linecount then n := linecount;
for x1 := 1 to n do writenull
end {  SKIP  };




{          WRITELINE - WRITE THE OUTPUT LINE.
 }

procedure writeline;
begin {  WRITELINE  }
beginline;
writestring(outline,outlength);
endline;
if space <> 0 then skip(space);
outlength := 1;
outline[1].nbl := leftmargin * charwidth;
nchars := leftmargin;
nwords := 0;
ngaps := 0;
gaps[0] := 1;
newoutline := true
end {  WRITELINE  };




{          PAGE - CONDITIONALLY PRODUCE A PAGE EJECT.
 }

procedure page( n : integer );
begin {  PAGE  }
if linecount < n
 then repeat while linecount > 0 do writenull;
   while (form[formindex] <> lbracket) and (linecount <= 0) do
    advanceform
  until form[formindex] = lbracket
 else if linecount = infinity then
  if 5 < n then skip(5)
end {  PAGE  };








{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
{                                                                       }
{                             INPUT                                     }
{                             -----                                     }
{                                                                       }
{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }




{          NEXTCHAR - ADVANCE TO THE NEXT INPUT CHARACTER, AND
 *                    CONVERT FROM HOST CHARACTER SET TO ASCII.
 }

procedure nextchar;




 {         READLINE - READ AN INPUT LINE, CONVERT INTO ASCII,
  *                   CONSIDERING CASE SHIFT AND UNDERLINING.
  }

 procedure readline;
 var
  extch         : char;      {  EXTERNAL CHARACTER  }
  intch         : ascii;     {  INTERNAL CHARACTER  }
  x1,x2         : integer;   {  GENERAL INDEX VARIABLES  }
 begin {  READLINE  }
 newinline := true;
 x1 := 0;
 while not eoln(infile) and (x1 < inwidth) do
  begin read(infile,extch);                                         {  }
  x1 := x1 + 1;                                                     {  }
  if not eoln(infile)                                               {  }
   then if ord(extch) =  60                                         {  }
    then begin intch := asc74[infile↑];                             {  }
     get(infile)                                                    {  }
     end                                                            {  }
    else if ord(extch) =  62                                        {  }
     then begin intch := asc76[infile↑];                            {  }
      get(infile)                                                   {  }
      end                                                           {  }
     else intch := asc[extch]                                       {  }
   else intch := asc[extch];                                        {  }
  inline[x1].c := intch
  end;
 inline[x1+1].c := blank;
 for x2 := 1 to x1+1 do inline[x2].nbl := charwidth;
 if inline[1].c = dirch
  then begin directline := true; lowerdir := true end
  else directline := directline and (inline[1].c = plus);
 if casech <> nul
  then if directline
   then shiftstring(inline,x1,lowerdir)
   else shiftstring(inline,x1,lowercase);
 if x1 > 1
  then while (inline[x1].c = blank) and (x1 > 1) do
   x1 := x1 - 1;
 if x1 = 1
  then if inline[x1].c = blank
   then x1 := 0;
 inlength := x1;
 readln(infile);
 firsterror := true;
 end {  READLINE  };




begin {  NEXTCHAR  }
incolumn := incolumn + 1;
if incolumn > inlength
 then if eol
  then if eof(infile)
   then endofinput := true
   else begin readline;
    incolumn := 1;
    if linenums
     then begin if class[inline[1].c].digit
      then begin linenumber := 0;
       repeat linenumber := linenumber * 10 +
                            inline[incolumn].c - zero;
        incolumn := incolumn + 1
       until not class[inline[incolumn].c].digit
       end;
      incolumn := incolumn + 1
      end
     else linenumber := linenumber + 1;
    eol := incolumn > inlength;
    if eol
     then inchar := blank
     else inchar := inline[incolumn].c
    end
  else begin eol := true;
   inchar := blank
   end
 else inchar := inline[incolumn].c
end {  NEXTCHAR  };






{          NEXTLINE - ADVANCE TO BEGINNING OF NEXT INPUT LINE.
 }

procedure nextline;
begin {  NEXTLINE  }
incolumn := inlength + 1;
eol := true;
nextchar
end {  NEXTLINE  };








{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
{                                                                       }
{                       DIRECTIVE PROCESSING                            }
{                       --------- ----------                            }
{                                                                       }
{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }




{          BREAK - CAUSE A BREAK IN JUSTIFICATION.
 }

procedure break;
begin {  BREAK  }
if not newoutline
 then begin if not (leftjustify and rightjustify)
   then justify;
  writeline
  end;
underlining := false;
newparagraph := true
end {  BREAK  };




{          INUNDENT - SCHEDULE AN INDENT OR UNDENT.
 *
 *        PARAM  INUN > 0 FOR INDENT,
 *                    < 0 FOR UNDENT.
```

```
  }

procedure inundent( inun : integer );
begin {  INUNDENT  }
break;
nchars := leftmargin + inun;
if nchars < 0 then nchars := 0;
outline[1].nbl := nchars * charwidth
end {  INUNDENT  };




{        INPSAVE - SAVE INPUT SETTINGS
  }

procedure inpsave;
begin {  INPSAVE  }
validate(keepinp,0,maxkeep,1151);
with saveinp[keepinp] do
 begin defined := true;
 b := explicitblank;
 c := casech;
 d := dirch;
 h := hyphen;
 u := underchar;
 w := inwidth
 end
end {  INPSAVE  };



{        INPRESTORE - RESTORE PREVIOUS INPUT SETTINGS.
  }

procedure inprestore;
begin {  INPRESTORE  }
validate(keepinp,0,maxkeep,1151);
with saveinp[keepinp] do
 if defined
  then begin explicitblank := b;
   if casech <> c
    then begin casech := c;
     lowercase := casech <> nul
     end;
   dirch := d;
   hyphen := h;
   underchar := u;
   inwidth := w
   end
  else error(1105)
end {  INPRESTORE  };




{        MARSAVE - SAVE MARGIN SETTINGS.
  }

procedure marsave;
begin {  MARSAVE  }
validate(keepmar,0,maxkeep,151);
with savemar[keepmar] do
 begin defined := true;
 l := leftmargin;
 r := rightmargin
 end
end {  MARSAVE  };




{        MARRESTORE - RESTORE PREVIOUS MARGIN SETTINGS.
  }

procedure marrestore;
begin {  MARRESTORE  }
validate(keepmar,0,maxkeep,151);
with savemar[keepmar] do
 if defined
  then begin leftmargin := l;
   rightmargin := r
   end
  else error(105)
end {  MARRESTORE  };




{        OPTSAVE - SAVE OPTION SETTINGS.
  }

procedure optsave;
begin {  OPTSAVE  }
validate(keepopt,0,maxkeep,251);
with saveopt[keepopt] do
 begin defined := true;
 e := printerrors;
 f := fill;
 j := badjustify;
 l := leftjustify;
 m := multipleblanks;
 p := ensure2;
 r := rightjustify;
 s := space;
 u := shiftup
 end
end {  OPTSAVE  };



{        OPTRESTORE - RESTORE PREVIOUS OPTION SETTINGS.
  }

procedure optrestore;
begin {  OPTRESTORE  }
validate(keepopt,0,maxkeep,251);
with saveopt[keepopt] do
 if defined
  then begin printerrors := e;
   fill := f;
   badjustify := j;
   leftjustify := l;
   multipleblanks := m;
   ensure2 := p;
   rightjustify := r;
   space := s;
   shiftup := u
   end
  else error(205)
end {  OPTRESTORE  };




{        PARSAVE - SAVE PARAGRAPH SETTINGS.
  }

procedure parsave;
begin {  PARSAVE  }
validate(keeppar,0,maxkeep,351);
with savepar[keeppar] do
 begin defined := true;
 c := 0;  {  IT WOULD SEEM THAT THIS IS SUPERFLUOUS  }
 f := parachar;
 i := lockeddent;
 n := numbering;
 p := parapage;
 s := paraskip;
 w := numberwidth
 end
end {  PARSAVE  };




{        PARRESTORE - RESTORE PREVIOUS PARAGRAPH SETTINGS.
  }

procedure parrestore;
begin {  PARRESTORE  }
validate(keeppar,0,maxkeep,351);
with savepar[keeppar] do
 if defined
  then begin paracount := c;
   parachar := f;
   lockeddent := i;
   numbering := n;
   parapage := p;
   paraskip := s;
   numberwidth := w
   end
  else error(305)
end {  PARRESTORE  };




{        DIRECTIVE - PROCESS ONE DIRECTIVE
  }

procedure directive;
var
 dir          : direct;     {  CURRENT DIRECTIVE  }
 fullword     : ch10;       {  CURRENT DIRECTIVE WORD  }
 word         : ch3;        {  3 LETTERS OF CURRENT DIRECTIVE WORD  }
 wordlength   : integer;    {  LENGTH OF CURRENT DIRECTIVE WORD  }
 x1,x2        : integer;    {  GENERAL INDEX VARIABLES  }








{        NEXTCH - ADVANCE TO NEXTCHAR, CONSIDERING CONTINUATIONS.
  }

procedure nextch;
begin {  NEXTCH  }
nextchar;
if eol and (infile↑ = '+')
 then begin nextchar;
  inchar := blank
  end
end {  NEXTCH  };




{        SWITCH - DETERMINE A SWITCH OPTION, CONSIDERING
 *               THE DEFAULT.
 *
 *      PARAM  DEF = DEFAULT.
  }

function switch( def : boolean ) : boolean;
begin {  SWITCH  }
if class[inchar].plusorminus
 then begin switch := inchar = plus;
```

```
   nextch
   end
  else switch := def
 end {  SWITCH  };




{         CHARACTER - DETERMINE A CHARACTER OPTION, CONSIDERING
 *                    THE DEFAULT.
 *
 *       PARAM  DEF = DEFAULT.
  }

 function character( def : ascii ) : ascii;
 begin {  CHARACTER  }
 if inchar <> blank
  then begin character := inchar;
   nextch
   end
  else character := def
 end {  CHARACTER  };




{         NUMBER - DETERMINE A NUMERIC OPTION, CONSIDERING
 *                  THE DEFAULT AND THE PREVIOUS VALUE.
 *
 *       PARAM  DEF  = DEFAULT.
 *              LAST = PREVIOUS VALUE, IF < 0 THEN
 *                     RELATIVE FORM IS NOT RECOGNIZED.
 *              MIN  = MINIMUM ALLOWED VALUE.
 *              MAX  = MAXIMUM ALLOWED VALUE.
 *              ERR  = ERROR NUMBER (IF OUT OF RANGE).
  }

 function number( def,last,min,max,err : integer ) : integer;
 var
  num           : integer;  {  NUMBER BEING BUILT  }
  sign          : ascii;    {  PLUS OR MINUS SIGN  }
 begin {  NUMBER  }
 if class[inchar].plusorminus and (last >= 0)
  then begin sign := inchar; nextch end
  else begin sign := plus; last := 0 end;
 if class[inchar].digit
  then begin num := 0;
   repeat num := num * 10 + inchar - zero;
    if num >= infinity then num := infinity - 1;
    nextch
   until not class[inchar].digit
   end
  else num := def;
 if sign = plus
  then num := last + num
  else num := last - num;
 if num < 0 then num := 0;
 validate(num,min,max,err);
 number := num
 end {  NUMBER  };




{         READWORD - READ THE NEXT DIRECTIVE WORD.
  }

 procedure readword;
 var
  xl            : integer;  {  LOOP INDEX  }
 begin {  READWORD  }
 wordlength := 0;
 while class[inchar].letter do
  begin wordlength := wordlength + 1;
  if wordlength <= 10
   then begin fullword[wordlength] := inchar;
    if wordlength <= 3 then word[wordlength] := upper(inchar)
    end;
  nextch
  end;
 for xl := wordlength + 1 to 10 do fullword[xl] := blank;
 for xl := wordlength + 1 to 3 do word[xl] := blank
 end {  READWORD  };




{         READPSTRING - READ A PSTRING UNTIL A TERMINATOR CHARACTER.
 *
 *       PARAM  STR = PSTRING TO BE READ.
 *              LEN = LENGTH OF PREDEFINED PORTION OF STR, UPDATED
 *                    TO NEW LENGTH.
 *              ENDC = TERMINATOR CHARACTER.
  }

 procedure readpstring( var str : pstring; var len : integer;
                        endc : ascii );
 begin {  READPSTRING  }
 underdir := false;
 while (inchar <> endc) and not eol do
  begin if inchar = underchar
   then underdir := not underdir
   else if len < maxstringlength
    then begin len := len + 1;
    if underdir
     then str[len] := inchar +  128
     else str[len] := inchar
    end;
  nextch
  end
 end {  READPSTRING  };




{         LOOKUP - LOOK UP THE DIRECTIVE WORD.
 *
 *       PARAM  FIRST = FIRST ACCEPTABLE DIRECTIVE WORD.
 *              ILLEGAL = LAST+1 ACCEPTABLE DIRECTIVE WORD.
  }

 function lookup( first,illegal : direct ) : direct;
 var
  d             : direct;   {  LOOKUP LOOP INDEX  }
 begin {  LOOKUP  }
 directs[illegal] := word;
 d := first;
 while (directs[d][1] <> word[1]) or
       (directs[d][2] <> word[2]) or
       (directs[d][3] <> word[3]) do
  d := succ(d);
 lookup := d
 end {  LOOKUP  };




{         INPUT - PROCESS INPUT DIRECTIVE.
  }

 procedure inputd;
 var
  ch            : ascii;    {  KEY CHARACTER  }
 begin {  INPUTD  }
 if inchar = lparen
  then begin nextch;
   keepinp := keepinp + 1;
   while (inchar <> rparen) and not eol do
    begin ch := upper(inchar);
    nextch;
    if class[ch].inputchar
     then case ch of
      b : explicitblank := character(nul);
      c : begin ch := character(nul);
          if ch <> casech
           then begin casech := ch;
            lowercase := casech <> nul
            end
          end;
      d : dirch := character(period);
      h : hyphen := character(nul);
      k : keepinp := number(0,-1,0,maxkeep,1151);
      u : underchar := character(nul);
      w : inwidth := number(150,-1,min,maxiwidth,1154);
      blank :
      end
     else begin errorl := ch; error(1101) end
    end;
   if inchar = rparen
    then nextch
    else error(1102);
   inpsave
   end
  else begin if class[inchar].digit
    then keepinp := number(0,-1,0,maxkeep,1151)
    else keepinp := keepinp - 1;
   inprestore
   end
 end {  INPUTD  };




{         LITERAL - PROCESS LITERAL DIRECTIVE.
  }

 procedure literal;
 var
  ch            : asciix;   {  LITERAL CHARACTER  }
  i             : integer;  {  LOOP INDEX  }
  litlength     : integer;  {  LENGTH OF LITSTRING  }
  litstring     : pstring;  {  ARGUMENT OF LITERAL DIRECTIVE  }
 begin {  LITERAL  }
 litlength := 0;
 readpstring(litstring,litlength,nul);
 for i := 1 to litlength do
  begin ch := litstring[i];
  if ch = explicitblank
   then writel(blank)
   else writel(ch)
  end;
 writeln
 end {  LITERAL  };




{         MARGIN - PROCESS MARGIN DIRECTIVE.
  }

 procedure margin;
 var
  ch            : ascii;    {  KEY CHARACTER  }
 begin {  MARGIN  }
 if inchar = lparen
  then begin nextch;
   keepmar := keepmar + 1;
   while (inchar <> rparen) and not eol do
    begin ch := upper(inchar);
    nextch;
    if class[ch].marginchar
     then case ch of
      k : keepmar := number(0,-1,0,maxkeep,151);
      l : leftmargin := number(0,leftmargin,0,infinity,0);
      r : rightmargin := number(70,rightmargin,0,infinity,0);
      blank :
      end
     else begin errorl := ch; error(101) end
    end;
   if inchar = rparen
    then nextch
    else error(102);
   validate(rightmargin,min,maxmargin,152);
   validate(leftmargin,0,rightmargin,153);
   marsave
   end
  else begin if class[inchar].digit
    then keepmar := number(0,-1,0,maxkeep,151)
    else keepmar := keepmar - 1;
   marrestore
   end;
 nchars := leftmargin;
 outline[1].nbl := nchars * charwidth
 end {  MARGIN  };




{        OPTION - PROCESS OPTION DIRECTIVE.
  }

 procedure option;
 var
  ch              : ascii;    {  KEY CHARACTER  }
 begin {  OPTION  }
 if inchar = lparen
  then begin nextch;
   keepopt := keepopt + 1;
   while (inchar <> rparen) and not eol do
    begin ch := upper(inchar);
    nextch;
    if class[ch].optionchar
     then case ch of
      e : printerrors := switch(true);
      f : fil! := switch(true);
      j : badjustify := number(3,-1,3,infinity,265) - 2;
      k : keepopt := number(0,-1,0,maxkeep,251);
      l : leftjustify := switch(true);
      m : multipleblanks := switch(true);
      p : ensure2 := switch(true);
      r : rightjustify := switch(true);
      s : space := number(1,-1,1,3,266) - 1;
      u : shiftup := switch(false);
      blank :
      end
     else begin errorl := ch; error(201) end
    end;
   if inchar = rparen
    then nextch
    else error(202);
   optsave
   end
  else begin if class[inchar].digit
    then keepopt := number(0,-1,0,maxkeep,251)
    else keepopt := keepopt - 1;
   optrestore
   end
 end {  OPTION  };




{        OUTPUT - PROCESS OUTPUT DIRECTIVE.
  }

 procedure outputd;
 var
  ch              : ascii;    {  KEY CHARACTER  }
 begin {  OUTPUTD  }
 if linecount < 0
  then begin if inchar = lparen
    then begin repeat nextch until (inchar <> blank) or eol;
     readword;
     if wordlength <= 3
      then terminaltype := lookup(ast,ilt)
      else terminaltype := ilt;
     if terminaltype = ilt
      then begin error(1009); terminaltype := ast end;
     case terminaltype of
      ast : ;
      lpt : carriagecontrol := one;
      ajt : begin while inchar = blank do nextch;
            charwidth := number(10,-1,0,infinity,1013);
            if not (charwidth in [10,12])
             then begin error(1013);
              charwidth := 10
              end;
            charwidth := 60 div charwidth;
            outline[1].nbl := leftmargin * charwidth
            end
      end;
     while (inchar <> rparen) and not eol do
      begin ch := upper(inchar);
      nextch;
      if class[ch].outputchar
       then case ch of
        e : eject := switch(false);
        p : pause := switch(false);
        s : shift := number(0,-1,0,maxshift,1064);
        u : underavail := switch(true);
        w : outwidth := number(maxowidth,-1,0,maxowidth,1054);
        blank :
        end
       else begin errorl := ch; error(1001) end
      end;
     if inchar = rparen
      then nextch
      else error(1002);
     shift := shift * charwidth;
     linecount := 0
     end
   end
  else error(1010)
 end {  OUTPUTD  };




{        PARAGRAPH - PROCESS PARAGRAPH DIRECTIVE.
  }

 procedure paragraph;
 var
  ch              : ascii;    {  KEY CHARACTER  }
 begin {  PARAGRAPH  }
 savepar[keeppar].c := paracount;
 if inchar = lparen
  then begin nextch;
   keeppar := keeppar + 1;
   paracount := 0;
   while (inchar <> rparen) and not eol do
    begin ch := upper(inchar);
    nextch;
    if class[ch].paragraphchar
     then case ch of
      c : paracount := number(0,-1,0,infinity,0);
      f : parachar := character(nul);
      i : lockeddent := number(5,-1,0,rightmargin-min,355);
      k : keeppar := number(0,-1,0,maxkeep,351);
      n : begin if not class[inchar].digit
           then numbering := numform(character(blank),307)
           else numbering := numeric;
          numberwidth := number(3,-1,0,maxnumberwidth,356)
          end;
      p : parapage := number(0,-1,0,infinity,0);
      s : paraskip := number(0,paraskip,0,maxskip,357);
      u : lockeddent := -number(0,-1,0,infinity,0);
      blank :
      end
     else begin errorl := ch; error(301) end
    end;
   if inchar = rparen
    then nextch
    else error(302);
   parsave
   end
  else if class[inchar].digit
   then begin keeppar := number(0,-1,0,maxkeep,351);
    parrestore;
    paracount := 0
    end
   else begin keeppar := keeppar - 1;
    parrestore
    end
 end {  PARAGRAPH  };




{        READFORM - READ THE FORM SPECIFICATION TO THE FORM BUFFER.
  }

 procedure readform;
 var
  nobracket       : boolean;  {  IF NO LBRACKET IN THE FORM  }
  quote           : ascii;    {  OUTER QUOTE CHARACTER FOR A STRING  }




 {       ADDCH - ADD A CHARACTER TO THE FORM.
  *
  *       PARAM  CH = CHARACTER TO ADD.
   }

  procedure addch( ch : ascii );
  begin {  ADDCH  }
  formnlength := formnlength + 1;
  formnext[formnlength] := ch
  end {  ADDCH  };




 begin {  READFORM  }
 formnlength := 0;
 nobracket := true;
 if inchar = lparen
  then begin nextch;
   while (inchar <> rparen) and not eol do
    begin addch(inchar);
    nobracket := nobracket and (inchar <> lbracket);
    if class[inchar].quote
     then begin quote := inchar;
      nextch;
      readpstring(formnext,formnlength,quote);
      if inchar = quote
       then nextch
       else error(403);
      addch(quote)
      end
     else nextch
    end;
   if inchar = rparen
    then nextch
    else error(402);
   if nobracket then addch(lbracket)
   end
  else linecount := infinity
 end {  READFORM  };




{        READINX - READ AN INDEX ENTRY.
  }

 procedure readinx;
 var
  index          : pstring;  {  INDEX BUFFER  }
  indexlength    : integer;  {  LENGTH OF INDEX  }
  p              : pinxentry;{  POINTER TO NEW INDEX ENTRY  }
  xl             : integer;  {  GENERAL INDEX VARIABLE  }
 begin {  READINDEX  }
 indexlength := 0;
 readpstring(index,indexlength,nul);
 new(p);
 if indexlength > maxinxlength then indexlength := maxinxlength;
 with p↑ do
  begin xl := indexlength;
  xp := pagenumber;
  for xl := 1 to indexlength do x[xl] := index[xl];
  for xl := indexlength+1 to maxinxlength do x[xl] := nul
  end;
 if inxbase = nil
  then inxbase := p
  else inxlast↑.next := p;
 inxlast := p
 end {  READINX  };




{        RESET - PROCESS RESET DIRECTIVE.
  }

 procedure reset;
 var
  d              : direct;   {  RESET DIRECTIVE NAME  }
  except         : boolean;  {  EXCEPT KEYWORD IS PRESENT  }
  first          : boolean;  {  FIRST DIRECTIVE NAME  }
  which          : dirset;   {  WHICH DIRECTIVES TO RESET  }
 begin {  RESET  }
 if inchar = lparen
  then begin first := true;
   except := false;
   which := [];
   nextch;
   while inchar <> rparen do
    if inchar = blank
     then nextch
     else if class[inchar].letter
      then begin readword;
       d := lookup(bre,ill);
       if d in [cou,frm,inp,inx,mar,opt,out,pag,par,sel,sbt,ttl]
        then which := which + [d]
        else if d = exc
         then if first
          then except := true
          else error(1211)
         else begin error10 := fullword;
          if d = ill
           then error(1206)
           else error(1212)
          end;
       first := false
       end
      else begin error1 := inchar; error(1201); nextch end;
   if except then which := [bre..ill] - which
   end
  else which := [bre..ill];
 while not eol do nextch;
 if [out,pag,frm] * which <> []
  then begin page(infinity);
   if linecount < infinity then advanceform
   end;
 reinitialize(which)
 end {  RESET  };




{        SELECT - PROCESS SELECT DIRECTIVE.
  }

 procedure select;
 var
  xl,x2          : integer;  {  GENERAL INDEX VARIABLES  }
 begin {  SELECT  }
 if inchar = lparen
  then begin nextch;
   for xl := 0 to maxpage do selection[xl] := false;
   while (inchar <> rparen) and not eol do
    if class[inchar].digit
     then begin xl := number(0,-1,0,maxpage,504);
      if inchar = colon
       then begin nextch;
        for xl := xl to number(xl,xl,xl,maxpage,504) do
         selection[xl] := true
        end
       else selection[xl] := true
      end
     else begin if inchar <> blank
       then begin error1 := inchar; error(501) end;
      nextchar
      end;
   if inchar = rparen
    then nextch
    else error(502)
   end
  else for xl := 0 to maxpage do selection[xl] := true
 end {  SELECT  };




{        SORTINX - SORT AND PRINT INDEX ENTRIES.
  }

 procedure sortinx;
 var
  firstinx       : pinxentry;{  FIRST ENTRY FOR SORTING  }
  lastinx        : pinxentry;{  LAST ENTRY FOR SORTING  }
  leftwidth      : integer;  {  L SPECIFICATION  }
  margin         : integer;  {  M SPECIFICATION  }
  pagecol        : integer;  {  P SPECIFICATION  }
  rightwidth     : integer;  {  R SPECIFICATION  }
  sortcol        : integer;  {  S SPECIFICATION  }




 {        PARSE - PARSE THE SORTINDEX DIRECTIVE.
   }

  procedure parse;
  var
   ch             : ascii;    {  KEY CHARACTER  }
  begin {  PARSE  }
  leftwidth := 2;
  margin := 0;
  pagecol := 0;
  rightwidth := 2;
  sortcol := 1;
  if inchar = lparen
   then begin nextch;
    while (inchar <> rparen) and not eol do
     begin ch := upper(inchar);
     nextch;
     if class[ch].sortinxchar
      then case ch of
       l : leftwidth := number(2,-1,0,30,658);
       m : margin := number(0,-1,0,30,659);
       p : pagecol := number(0,-1,0,maxinxlength+min,660);
       r : rightwidth := number(2,-1,0,30,661);
       s : if (inchar = p) or (inchar = smallp)
             then begin sortcol := -1; nextch end
             else sortcol := number(1,-1,1,maxinxlength-min,662);
       blank :
       end
      else begin error1 := ch; error(601) end
     end;
    if inchar = rparen
     then nextch
     else error(602)
    end
  end {  PARSE  };




 {        SORT - SORT THE INDEX ENTRIES.
   }

  procedure sort;
  var
   p              : pinxentry;{  FOR TRAVERSING THE INDEX LIST  }
   s1,s2          : pinxentry;{  TEMPS FOR SORTING  }
   xl             : integer;  {  GENERAL INDEX VARIABLE  }
  begin {  SORT  }
  new(firstinx);
  new(lastinx);
  with firstinx↑ do
   begin xl := 0;
   next := lastinx;
   for xl := 1 to maxinxlength do x[xl] := nul
   end;
  with lastinx↑ do
   begin xl := 0;
   next := nil;
   for xl := 1 to maxinxlength do x[xl] := del
   end;
  if sortcol < 0
   then begin inxlast↑.next := lastinx;
    firstinx↑.next := inxbase;
    inxbase := nil
    end
   else begin p := inxbase;
    inxlast↑.next := nil;
    while p <> nil do
     begin inxbase := p↑.next;
     s2 := firstinx;
     repeat s1 := s2;
      s2 := s1↑.next;
      xl := sortcol;
      while (xl < maxinxlength) and
            (upper(p↑.x[xl]) = upper(s2↑.x[xl])) do
       xl := xl + 1
     until upper(p↑.x[xl]) < upper(s2↑.x[xl]);
     s1↑.next := p;
     p↑.next := s2;
     p := inxbase
     end
    end
  end {  SORT  };




 {       PRINT - PRINT THE INDEX ENTRIES.
   }

  procedure print;
  var
   p             : pinxentry;{  FOR TRAVERSING THE INDEX LIST  }
   x1            : integer;  {  GENERAL INDEX VARIABLE  }




  {       SEND1 - SEND ONE CHARACTER TO THE OUTPUT LINE.
   *
   *      PARAM  CH - CHARACTER TO SEND.
    }

   procedure send1( ch : asciix );
   begin {  SEND1  }
   outlength := outlength + 1;
   with outline[outlength] do
    begin c := ch;
    nbl := charwidth
    end
   end {  SEND1  };




  begin {  PRINT  }
  p := firstinx↑.next;
  while p <> lastinx do
   with p↑ do
    begin for x1 := 1 to margin do send1(blank);
    for x1 := 1 to pagecol do
     if x1 > x1
      then send1(blank)
      else send1(x[x1]);
    convertnumber(outline,outlength,xp,leftwidth,numeric);
    for x1 := 1 to rightwidth do send1(blank);
    for x1 := pagecol+1 to x1 do send1(x[x1]);
    writeline;
    dispose(firstinx);
    firstinx := p;
    p := firstinx↑.next
    end;
   dispose(lastinx)
   end {  PRINT  };




  begin {  SORTINX  }
  parse;
  sort;
  print
  end {  SORTINX  };




 begin {  DIRECTIVE  }
 repeat nextch;
  readword;
  dir := lookup(bre,ill);
  while (inchar = blank) and not eol do nextch;
  if dir in [bre,frm,ind,mar,opt,pag,res,ski,sor,und,weo] then break;
  case dir of
   bre : ;
   com : while not eol do nextch;
   cou : pagenumber := number(1,pagenumber,0,maxpage,759);
   frm : readform;
   ind : inundent(number(5,-1,0,rightmargin,856));
   inp : inputd;
   inx : readinx;
   lit : literal;
   mar : margin;
   opt : option;
   out : outputd;
   pag : page(number(infinity,-1,0,infinity,0));
   par : paragraph;
   res : reset;
   sel : select;
   ski : skip(number(5,-1,0,maxskip,957));
   sor : sortinx;
   sbt : begin titlelength[subtitle] := 0;
         readpstring(title[subtitle],titlelength[subtitle],nul)
         end;
   ttl : begin titlelength[maintitle] := 0;
         readpstring(title[maintitle],titlelength[maintitle],nul)
         end;
   und : inundent(-number(infinity,-1,0,infinity,0));
   weo : putseg(output);
   exc,
   ill : begin error10 := fullword; error(006) end
   end;
  while (inchar <> dirch) and not eol do
   begin if inchar <> blank
    then begin error1 := inchar; error(1) end;
   nextch
   end
 until eol
 end {  DIRECTIVE  };








{ ------------------------------------------------------------------ }
{                                                                    }
{                         TEXT FORMATTING                            }
{                         ---- ----------                            }
{                                                                    }
{ ------------------------------------------------------------------ }






{         NEXTWORD - READ THE NEXT INPUT WORD, PROCESS DIRECTIVES
 *                   WHEN APPROPRIATE.
  }

 procedure nextword;
 var
  x1              : integer;  {  LOOP INDEX  }
 begin {  NEXTWORD  }
 wordlength := 0;
 newinline := false;
 while eol and not endofinput do
  begin nextchar;
  if eol and not endofinput
   then begin break; writenull end
   else if inchar = dirch
    then directive
    else if inchar = parachar
     then begin break;
      if paraskip > 0 then skip(paraskip);
      if parapage > 0 then page(parapage);
      inundent(lockeddent);
      if numbering <> nonumbering
       then begin paracount := paracount + 1;
        convertnumber(word,wordlength,paracount,numberwidth,numbering)
        end;
      nextchar
      end
  end;
 if not endofinput
  then begin nblanks := 0;
   if wordlength = 0
    then while inchar = blank do
     begin nblanks := nblanks + 1;
     nextchar
     end;
   if newinline
    then begin if (nblanks > 0) or not fill then break;
     if underchar <> nul
      then begin understring(inline,inlength,underlining);
       incolumn := incolumn - 1;
       nextchar
       end
     end
    else if not multipleblanks and (nblanks > 1) then nblanks := 1;
   nsplits := 0;
   while inchar <> blank do
    begin if inchar mod  128 = hyphen
     then begin if nsplits < maxsplit
       then begin nsplits := nsplits + 1;
        with splits[nsplits] do
         begin point := wordlength;
         if incolumn > 1
          then hypnt := class[inline[incolumn-1].c mod  128].letter and
                        class[inline[incolumn+1].c mod  128].letter
          else hypnt := false;
         inpnt := incolumn
         end
        end
      end
     else begin wordlength := wordlength + 1;
      with word[wordlength] do
       begin c := inchar; nbl := charwidth end
      end;
    nextchar
    end
   end
 end {  NEXTWORD  };




  {         PACKWORD - PACK A WORD INTO THE OUTPUT LINE.
    }

  procedure packword;
  var
   nb              : integer;  {  NUMBER BLANKS (PRECEDING WORD)  }
   nc              : integer;  {  NCHARS PREDICTED AFTER ADDING WORD  }




   {        ADDWORD - ADD THE WORD TO THE OUTPUT LINE.
     }

   procedure addword;
   var
    x1              : integer;  {  GENERAL INDEX VARIABLE  }
   begin {  ADDWORD  }
   with outline[outlength] do nbl := nbl + nb * charwidth;
```

```
for x1 := 1 to wordlength do
 begin outlength := outlength + 1;
 outline[outlength] := word[x1]
 end;
outlength := outlength + 1;
with outline[outlength] do
 begin c := blank; nbl := 0 end;
nchars := nc;
if nchars >= leftmargin
 then begin ngaps := ngaps + 1;
  gaps[ngaps] := outlength
  end
 else gaps[0] := outlength
end {  ADDWORD  };




{         SETUP - SET UP FOR PACKWORD.
  }

procedure setup;
var
 x1            : integer;  {  LOOP INDEX  }
begin {  SETUP  }
if newparagraph
 then nb := nblanks
 else if newoutline
  then nb := 0
  else begin if newinline
    then nb := nblanks + 1
    else nb := nblanks;
   if ensure2 and
      (outline[outlength-1].c mod  128 = period) and
      (nblanks < 2) and (nchars >= leftmargin)
    then nb := 2
   end;
nc := nchars + nb + wordlength;
if nc > rightmargin
 then if rightmargin - nchars > badjustify * (ngaps - 1)
  then {  GOING TO INSERT TOO MANY BLANKS  }
   begin if nsplits > 0
    then begin x1 := nsplits;
     while x1 > 0 do with splits[x1] do
      begin nc := nchars + nb + point + ord(hypnt);
      if nc <= rightmargin
       then begin x1 := 0; {  EXIT LOOP  }
        incolumn := inpnt;      {  RESET INPUT STREAM  }
        eol := false;
        nextchar;
        wordlength := point + ord(hypnt);
        if hypnt then word[wordlength].c := minus
        end
       else x1 := x1 - 1
      end
     end;
   if nc > rightmargin then error(008)
   end;
 newoutline := false;
 newparagraph := false
 end {  SETUP  };




begin {  PACKWORD  }
setup;
if nc <= rightmargin then addword;
if nc >= rightmargin
 then {  DON-T CALL PACKWORD, TO PREVENT UNENDING RECURSION IN      }
     {  THE CASE OF A WORD THAT DOESN-T FIT BETWEEN THE MARGINS  }
  begin justify;
  writeline;
  if nc > rightmargin
   then begin setup;
    addword;
    if nc >= rightmargin then begin justify; writeline end
    end
  end
end {  PACKWORD  };








{ - - - - - - - - - - - - - - - - - - - - - - - - - - - }
{                                                       }
{                    ERROR PROCESSING                   }
{                    ----- ----------                   }
{                                                       }
{ - - - - - - - - - - - - - - - - - - - - - - - - - - - }




{        ERROR - ISSUE AN ERROR MESSAGE.
 *
 *        PARAM  N = ERROR NUMBER.
 *               N IS NEGATIVE FOR ERRORS DETECTED DURING FORM
 *               PROCESSING TO PREVENT UNENDING RECURSION.
 *               FOR POSITIVE N, THE FOLLOWING CONVENTION IS USED:
 *               N DIV 100 INDICATES WHICH DIRECTIVE THE
 *               REFERS TO.
 *               N MOD 100 SELECTS A PARTICULAR ERROR MESSAGE.
 *               N MOD 100 IS >= 50 FOR NUMERIC ERRORS.
 *               GLOBAL VARIABLES ERROR10, ERROR1, ERRORN1, ERRORN2,
 *               AND ERRORSMALL ARE USED FOR PRINTING SPECIFIC
```

```
 *               VALUES WHICH ARE IN ERROR.
 }

procedure error(  N : INTEGER  };
type
 host5         = packed array[1..5] of char;
 host10        = packed array[1..10] of char;
 host20        = packed array[1..20] of char;
var
len            : integer;  {  LENGTH OF STR  }
 str           : string;   {  FOR PRINTING INLINE  }
 x1,x2         : integer;  {  GENERAL LOOP INDEX  }




{       WR5,WR10,WR20 - WRITE HOST CHARACTERS TO STR.
  }

 procedure wr5( cs : host5; nc : integer);
 var x1 : integer;
 begin {  WR5  }
 for x1 := 1 to nc do
  begin len := len + 1;
  with str[len] do begin c := asciichar(cs[x1]); nbl := charwidth end
  end
 end {  WR5  };


 procedure wr10( cs : host10; nc : integer);
 var x1 : integer;
 begin {  WR10  }
 for x1 := 1 to nc do
  begin len := len + 1;
  with str[len] do begin c := asciichar(cs[x1]); nbl := charwidth end
  end
 end {  WR10  };


 procedure wr20( cs : host20; nc : integer);
 var x1 : integer;
 begin {  WR20  }
 for x1 := 1 to nc do
  begin len := len + 1;
  with str[len] do begin c := asciichar(cs[x1]); nbl := charwidth end
  end
 end {  WR20  };




begin {  ERROR  }
if printerrors
 then begin errors := true;
  str[1].c := blank; str[1].nbl := 0;
  len := 1;
  wr5('---- ',5);
  if n < 0
   then begin wr20('FORM ERROR:         ',12);
    case n of
     -1 : wr20('LINE TOO LONG       ',13);
     -2 : begin len := len + 1;
          with str[len] do begin c := error1; nbl := charwidth end
          end;
     -3 : wr20('PAGENUMBER TOO LARGE',20);
     -4 : wr20('BAD NUMERIC FORM    ',16);
     -5 : wr20('NO "L" FOUND        ',12);
     end;
    writestring(str,len);
    endline
    end
   else begin if firsterror {  FIRST ERROR ON THIS LINE  }
     then begin convertnumber(str,len,linenumber,4,numeric);
      wr5('.   ',2);
      for x1 := 1 to inlength do str[len+x1] := inline[x1];
      len := len + inlength;
      writestring(str,len);
      endline;
      firsterror := false;
      str[1].nbl := 0;
      len := 6
      end;
    case n div 100 of
      0 : ;
      1 : wr10('MARGIN    ',6);
      2 : wr10('OPTION    ',6);
      3 : wr10('PARAGRAPH ',9);
      4 : wr5('FORM ',4);
      5 : wr10('SELECT    ',6);
      6 : wr10('SORTINDEX ',9);
      7 : wr5('COUNT',5);
      8 : wr10('INDENT    ',6);
      9 : wr5('SKIP ',4);
     10 : wr10('OUTPUT    ',6);
     11 : wr5('INPUT',5);
     12 : wr5('RESET',5);
     end;
    wr10(' ERROR:   ',8);
    n := n mod 100;
    if n < 50
     then case n of
       1 : begin len := len + 1;
           with str[len] do begin c := error1; nbl := charwidth end
           end;
       2 : wr10('MISSING ) ',9);
       3 : wr20('UNMATCHED QUOTE     ',15);
       4 : wr20('PAGENUMBER TOO LARGE',20);
       5 : begin wr20('UNDEFINED KEEP BUFFE',20);
           wr5('R    ',1)
           end;
      6 : begin wr20('UNKNOWN DIRECTIVE:  ',19);
          for x1 := 1 to 10 do
           begin len := len + 1;
           with str[len] do
            begin c := error10[x1]; nbl := charwidth end
           end
          end;
      7 : wr20('BAD NUMERIC FORM    ',16);
      8 : begin wr20('HYPHENATION NEEDED: ',20);
          for x1 := 1 to wordlength do
           if len < maxstringlength then
            begin len := len + 1; str[len] := word[x1] end
          end;
      9 : wr20('BAD TERMINAL TYPE   ',17);
     10 : begin wr20('MUST BE IN INITIAL D',20);
          wr20('IRECTIVE GROUP      ',14)
          end;
     11 : begin wr20('"EXCEPT" MUST BE FIR',20);
          wr5('ST   ',2)
          end;
     12 : begin wr20('DIRECTIVE NOT ALLOWE',20);
          wr5('D:   ',3);
          for x1 := 1 to 10 do
           begin len := len + 1;
           with str[len] do
            begin c := error10[x1]; nbl := charwidth end
           end
          end;
     13 : begin wr20('AJ PITCH MUST BE 10 ',20);
          wr5('OR 12',5)
          end;
     end
    else begin case n of
      51 : wr5('KEEP ',4);
      52 : wr20('RIGHT MARGIN        ',12);
      53 : wr20('LEFT MARGIN         ',11);
      54 : wr5('WIDTH',5);
      55 : wr10('INDENT    ',6);
      56 : wr20('NUMBER WIDTH        ',12);
      57 : wr5('SKIP ',4);
      58 : wr10('LEFT WIDTH',10);
      59 : wr10('MARGIN    ',6);
      60 : wr20('PAGE COLUMN         ',11);
      61 : wr20('RIGHT WIDTH         ',11);
      62 : wr20('SORT COLUMN         ',11);
      64 : wr5('SHIFT',5);
      65 : wr20('JUSTIFICATION LIMIT ',19);
      66 : wr10('SPACING   ',7);
       end;
      wr5(' OF  ',4);
      if errorn1 < 0
       then begin wr5('-    ',1); errorn1 := -errorn1 end;
      convertnumber(str,len,errorn1,0,numeric);
      wr10(' IS TOO   ',8);
      if errorsmall then wr5('SMALL',5) else wr5('LARGE',5);
      wr5(',    ',2);
      convertnumber(str,len,errorn2,0,numeric);
      wr5(' USED',5)
      end;
    writestring(str,len);
    endline
    end
  end
end {  ERROR  };




{        VALIDATE NUMERIC OPTION.
 *
 *      PARAM  NUM = NUMBER TO TEST.
 *             MIN = MINIMUM ALLOWED VALUE.
 *             MAX = MAXIMUM ALLOWED VALUE.
 *             ERR = ERROR NUMBER IF NOT IN RANGE.
  }

procedure validate{  VAR NUM : INTEGER; MIN,MAX,ERR : INTEGER  };
begin {  VALIDATE  }
errorn1 := num;
errorsmall := num < min;
if errorsmall
 then begin num := min; errorn2 := num; error(err) end
 else if num > max
  then begin num := max; errorn2 := num; error(err) end
end {  VALIDATE  };









{  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  }
{                                                                          }
{                      SECONDARY INITIALIZATION                            }
{                      ------------------------                            }
{                                                                          }
{  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  }




{         REINITIALIZE - RE-INITIALIZE GLOBAL VARIABLES.
  }


procedure reinitialize;
var
 d                : direct;   {  DIRECTIVE LOOP INDEX  }
 x1               : integer;  {  LOOP INDEX  }




{        INITFORM - INITIALIZE DEFAULT FORM.
  }

 procedure initform;
 var
  default        : packed array[1..40] of char;
                                  {  DEFAULT FORM  }
  x1             : integer;  {  LOOP INDEX  }
 begin {  INITFORM  }
 default := '[//T#62E///L56//#33"- "PN:1" -"///]     ';
 for x1 := 1 to 40 do
  form[x1] := asciichar(default[x1]);
 formlength := 40;
 formnlength := 0;
 formindex := 1;
 textlength := 1;
 textindex := 1;
 text[1].c := blank;
 text[1].nbl := 0;
 end {  INITFORM  };




{        INITINP - INITIALIZE INPUT SETTINGS.
  }

 procedure initinp;
 var
  x1             : integer;  {  LOOP INDEX  }
 begin {  INITINP  }
 lowercase := true;
 lowerdir := true;
 underdir := false;
 underlining := false;
 keepinp := 0;
 explicitblank := nul;
 casech := nul;
 dirch := period;
 hyphen := nul;
 underchar := nul;
 inwidth := 150;
 for x1 := 0 to maxkeep do saveinp[x1].defined := false;
 inpsave
 end {  INITINP  };




{        INITINX - INITIALIZE INX VARIABLES.
  }

 procedure initinx;
 var
  ip             : pinxentry;{  TO DISPOSE INDEX ENTRIES  }
 begin {  INITINX  }
 while inxbase <> nil do
  begin ip := inxbase;
  inxbase := inxbase↑.next;
  dispose(ip)
  end;
 inxlast := nil
 end {  INITINX  };




{        INITMAR - INITIALIZE MARGIN SETTINGS.
  }

 procedure initmar;
 var
  x1             : integer;  {  LOOP INDEX  }
 begin {  INITMAR  }
 keepmar := 0;
 leftmargin := 0;
 rightmargin := 70;
 for x1 := 0 to maxkeep do savemar[x1].defined := false;
 nchars := 0;
 outline[1].nbl := 0;
 marsave
 end {  INITMAR  };




{        INITOPT - INITIALIZE OPTION SETTINGS.
  }

 procedure initopt;
 var
  x1             : integer;  {  LOOP INDEX  }
 begin {  INITOPT  }
 keepopt := 0;
 printerrors := true;
 fill := true;
 badjustify := 1;
 leftjustify := true;
 multipleblanks := true;
 ensure2 := true;
 rightjustify := true;
 space := 0;
 shiftup := false;
 for xl := 0 to maxkeep do saveopt[xl].defined := false;
 optsave
 end {  INITOPT  };




{          INITOUT - INITIALIZE OUTPUT SETTINGS.
  }

 procedure initout;
 begin {  INITOUT  }
 blankcount := 0;
 blankline := false;
 carriagecontrol := blank;
 linecount := -1;
 terminaltype := ast;
 charwidth := 1;
 eject := false;
 pause := false;
 shift := 0;
 underavail := true;
 outwidth := maxowidth
 end {  INITOUT  };




{          INITPAR - INITIALIZE PARAGRAPH SETTINGS.
  }

 procedure initpar;
 var
  xl              : integer;  {  LOOP INDEX  }
 begin {  INITPAR  }
 keeppar := 0;
 paracount := 0;
 parachar := nul;
 lockeddent := 0;
 numbering := nonumbering;
 parapage := 0;
 paraskip := 0;
 numberwidth := 3;
 for xl := 0 to maxkeep do savepar[xl].defined := false;
 parsave
 end {  INITPAR  };





begin {  REINITIALIZE  }
for d := bre to ill do
 if d in which
  then case d of
   bre : ;
   com : ;
   cou : pagenumber := 1;
   frm : initform;
   ind : ;
   inp : initinp;
   inx : initinx;
   lit : ;
   mar : initmar;
   opt : initopt;
   out : initout;
   pag : ;
   par : initpar;
   res : ;
   sel : for xl := 0 to maxpage do selection[xl] := true;
   ski : ;
   sor : ;
   sbt : titlelength[subtitle] := 0;
   ttl : titlelength[maintitle] := 0;
   und : ;
   weo : ;
   exc : ;
   ill :
   end
end {  REINITIALIZE  };









{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
{                                                                     }
{                        PRIMARY INITIALIZATION                       }
{                        ----------------------                       }
{                                                                     }
{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }





{          INITIALIZE - INITIALIZE GLOBAL VARIABLES.
  }

procedure initialize;




 {         INITASC - INITIALIZE HOST TO ASCII CONVERSION TABLES.
  }
```

```

 procedure initasc;                                                 {   }
 var                                                                {   }
  extch          : char;     {  EXTERNAL CHARACTER  }               {   }
  intch          : ascii;    {  INTERNAL CHARACTER  }               {   }
 begin {  INITASC  }                                                {   }
 asc[chr( 0)] := colon;                                             {   }
 intch := a;                                                        {   }
 for extch := 'A' to 'Z' do                                         {   }
  begin asc[extch] := intch; intch := intch + 1 end;                {   }
 intch := zero;                                                     {   }
 for extch := '0' to '9' do                                         {   }
  begin asc[extch] := intch; intch := intch + 1 end;                {   }
 asc['+'] := plus;                                                  {   }
 asc['-'] := minus;                                                 {   }
 asc['/'] := slash;                                                 {   }
 asc['*'] := star;                                                  {   }
 asc['('] := lparen;                                                {   }
 asc[')'] := rparen;                                                {   }
 asc['$'] := dollar;                                                {   }
 asc['='] := equal;                                                 {   }
 asc[' '] := blank;                                                 {   }
 asc[','] := comma;                                                 {   }
 asc['.'] := period;                                                {   }
 asc['#'] := hash;                                                  {   }
 asc['['] := lbracket;                                              {   }
 asc[']'] := rbracket;                                              {   }
 asc[':'] := colon;                                                 {   }
 asc['"'] := dquote;                                                {   }
 asc['_'] := underscore;                                            {   }
 asc['!'] := exclaim;                                               {   }
 asc['&'] := ampersand;                                             {   }
 asc[''''] := squote;                                               {   }
 asc['?'] := question;                                              {   }
 asc['<'] := less;                                                  {   }
 asc['>'] := greater;                                               {   }
 asc[chr( 60)] := nul;                                              {   }
 asc['\'] := backslash;                                             {   }
 asc[chr( 62)] := nul;                                              {   }
 asc[';'] := semicolon;                                             {   }
 for extch := chr(  0) to chr( 63) do asc74[extch] := nul;          {   }
 asc74[chr(  1)] := at;                                             {   }
 asc74[chr(  2)] := caret;                                          {   }
 asc74[chr(  4)] := percent;                                        {   }
 asc74[chr(  7)] := grav;                                           {   }
 asc76[chr(  0)] := nul;                                            {   }
 intch := smalla;                                                   {   }
 for extch := 'A' to 'Z' do                                         {   }
  begin asc76[extch] := intch; intch := intch + 1 end;              {   }
 asc76[chr( 27)] := lbrace;                                         {   }
 asc76[chr( 28)] := verticalbar;                                    {   }
 asc76[chr( 29)] := rbrace;                                         {   }
 asc76[chr( 30)] := tilde;                                          {   }
 asc76[chr( 31)] := del;                                            {   }
 intch := nul;                                                      {   }
 for extch := chr( 32) to chr( 63) do                               {   }
  begin asc76[extch] := intch; intch := intch + 1 end               {   }
 end {  INITASC  };                                                 {   }




{          INITCLASS - INITIALIZE THE CLASSIFICATION TABLE.
  }

 procedure initclass;
 var
  ch              : ascii;    {  INDEX VARIABLE  }
  empty           : charclass;{  ALL FIELDS ARE FALSE  }
 begin {  INITCLASS  }
 with empty do
  begin letter := false;
  digit := false;
  formchar := false;
  optionchar := false;
  marginchar := false;
  paragraphch := false;
  sortinxchar := false;
  plusorminus := false;
  quote := false;
  numform := false;
  end;
 for ch := nul to del do class[ch] := empty;
 for ch := a to z do class[ch].letter := true;
 for ch := smalla to smallz do class[ch].letter := true;
 for ch := zero to nine do class[ch].digit := true;
 class[c].formchar := true;
 class[d].formchar := true;
 class[e].formchar := true;
 class[l].formchar := true;
 class[p].formchar := true;
 class[s].formchar := true;
 class[t].formchar := true;
 class[w].formchar := true;
 class[hash].formchar := true;
 class[lbracket].formchar := true;
 class[rbracket].formchar := true;
 class[slash].formchar := true;
 class[dquote].formchar := true;
 class[squote].formchar := true;
 class[blank].formchar := true;
 class[b].inputchar := true;
 class[c].inputchar := true;
 class[d].inputchar := true;
 class[h].inputchar := true;
 class[k].inputchar := true;
 class[u].inputchar := true;
 class[w].inputchar := true;
 class[blank].inputchar := true;
 class[k].marginchar := true;
```

```
class[l].marginchar := true;
class[r].marginchar := true;
class[blank].marginchar := true;
class[e].optionchar := true;
class[f].optionchar := true;
class[j].optionchar := true;
class[k].optionchar := true;
class[l].optionchar := true;
class[m].optionchar := true;
class[p].optionchar := true;
class[r].optionchar := true;
class[s].optionchar := true;
class[u].optionchar := true;
class[blank].optionchar := true;
class[e].outputchar := true;
class[p].outputchar := true;
class[s].outputchar := true;
class[u].outputchar := true;
class[w].outputchar := true;
class[blank].outputchar := true;
class[c].paragraphch := true;
class[f].paragraphch := true;
class[i].paragraphch := true;
class[k].paragraphch := true;
class[n].paragraphch := true;
class[p].paragraphch := true;
class[s].paragraphch := true;
class[u].paragraphch := true;
class[blank].paragraphch := true;
class[l].sortinxchar := true;
class[m].sortinxchar := true;
class[p].sortinxchar := true;
class[r].sortinxchar := true;
class[s].sortinxchar := true;
class[blank].sortinxchar := true;
class[plus].plusorminus := true;
class[minus].plusorminus := true;
class[dquote].quote := true;
class[squote].quote := true;
class[n].numform := true;
class[smalln].numform := true;
class[l].numform := true;
class[smalll].numform := true;
class[r].numform := true;
class[smallr].numform := true;
class[blank].numform := true;
end {  INITCLASS  };




{         INITCLOCKS - INITIALIZE RAWCLOCK AND WALLCLOCK.
 }

procedure initclocks;                                                  {  }
var                                                                    {  }
 c1             : ascii;    {  TENS DIGIT OF WALLCLOCK  }              {  }
 c2             : ascii;    {  ONES DIGIT OF WALLCLOCK  }              {  }
 c3             : ascii;    {  A OR P FOR AM OR PM  }                  {  }
 systemclock    : alfa;     {  SYSTEM CLOCK AS ' HH.MM.SS.'  }         {  }
 x1             : integer;  {  GENERAL LOOP INDEX  }                   {  }
begin {  INITCLOCKS  }                                                 {  }
{  IF NO SYSTEM CLOCK:              }                                  {  }
{  RAWCLOCK[ 1] := N;               }                                  {  }
{  RAWCLOCK[ 2] := O;               }                                  {  }
{  RAWCLOCK[ 3] := BLANK;           }                                  {  }
{  RAWCLOCK[ 4] := C;               }                                  {  }
{  RAWCLOCK[ 5] := L;               }                                  {  }
{  RAWCLOCK[ 6] := O;               }                                  {  }
{  RAWCLOCK[ 7] := C;               }                                  {  }
{  RAWCLOCK[ 8] := K;               }                                  {  }
{  RAWCLOCK[ 9] := BLANK;           }                                  {  }
{  RAWCLOCK[10] := BLANK;           }                                  {  }
{  WALLCLOCK := RAWCLOCK;           }                                  {  }
time(systemclock);                                                     {  }
for x1 := 1 to 8 do rawclock[x1] := asc[systemclock[x1+1]];            {  }
rawclock[9] := blank;                                                  {  }
rawclock[10] := blank;                                                 {  }
c1 := rawclock[1];                                                     {  }
c2 := rawclock[2];                                                     {  }
c3 := a;                                                               {  }
case c1 of                                                             {  }
 zero : if c2 = zero                                                   {  }
         then begin c1 := one; c2 := two end                           {  }
         else c1 := blank;                                             {  }
 one  : if c2 = two                                                    {  }
         then c3 := p                                                  {  }
         else if c2 > two                                              {  }
          then begin c1 := blank; c2 := c2 - 2; c3 := p end;           {  }
 two  : begin if c2 <= one                                             {  }
         then begin c1 := blank; c2 := c2 - 2 end                      {  }
         else begin c1 := one; c2 := c2 + 2 end;                       {  }
        c3 := p                                                        {  }
        end                                                            {  }
 end;                                                                  {  }
wallclock[ 1] := c1;                                                   {  }
wallclock[ 2] := c2;                                                   {  }
wallclock[ 3] := colon;                                                {  }
wallclock[ 4] := rawclock[4];                                          {  }
wallclock[ 5] := rawclock[5];                                          {  }
wallclock[ 6] := blank;                                                {  }
wallclock[ 7] := c3;                                                   {  }
wallclock[ 8] := m;                                                    {  }
wallclock[ 9] := blank;                                                {  }
wallclock[10] := blank                                                 {  }
end {  INITCLOCKS  };                                                  {  }




{         INITDATES - INITIALIZE RAWDATE AND NICEDATE.
 }

procedure initdates;                                                   {  }
var                                                                    {  }
 month          : ch3;      {  CURRENT MONTH NAME  }                   {  }
 systemdate     : alfa;     {  SYSTEM DATE AS ' YY/MM/DD.'  }          {  }
 x1             : integer;  {  GENERAL LOOP INDEX  }                   {  }
begin {  INITDATES  }                                                  {  }
{  IF NO SYSTEM DATE:               }                                  {  }
{  RAWDATE[ 1] := N;                }                                  {  }
{  RAWDATE[ 2] := O;                }                                  {  }
{  RAWDATE[ 3] := BLANK;            }                                  {  }
{  RAWDATE[ 4] := D;                }                                  {  }
{  RAWDATE[ 5] := A;                }                                  {  }
{  RAWDATE[ 6] := T;                }                                  {  }
{  RAWDATE[ 7] := E;                }                                  {  }
{  RAWDATE[ 8] := BLANK;            }                                  {  }
{  RAWDATE[ 9] := BLANK;            }                                  {  }
{  RAWDATE[10] := BLANK;            }                                  {  }
{  NICEDATE := RAWDATE;             }                                  {  }
date(systemdate);                                                      {  }
for x1 := 1 to 8 do rawdate[x1] := asc[systemdate[x1+1]];              {  }
rawdate[9] := blank;                                                   {  }
rawdate[10] := blank;                                                  {  }
month := months[(rawdate[4] - zero) * 10 + rawdate[5] - zero];{  }
nicedate[ 1] := rawdate[7];                                            {  }
nicedate[ 2] := rawdate[8];                                            {  }
nicedate[ 3] := blank;                                                 {  }
nicedate[ 4] := month[1];                                              {  }
nicedate[ 5] := month[2];                                              {  }
nicedate[ 6] := month[3];                                              {  }
nicedate[ 7] := blank;                                                 {  }
nicedate[ 8] := rawdate[1];                                            {  }
nicedate[ 9] := rawdate[2];                                            {  }
nicedate[10] := blank                                                  {  }
end {  INITDATES  };                                                   {  }




{         INITDIRECTS - INITIALIZE THE DIRECTS TABLE.
 }

procedure initdirects;




{         ONEDIRECT - INITIALIZE ONE DIRECT ENTRY.
 *
 *       PARAM  DIR = DIRECTIVE.
 *              A,B,C = 3 CHARACTERS OF DIRECTIVE NAME.
  }

 procedure onedirect( dir : direct; a,b,c : ascii );
 begin {  ONEDIRECT  }
 directs[dir][1] := a;
 directs[dir][2] := b;
 directs[dir][3] := c
 end {  ONEDIRECT  };




begin {  INITDIRECTS  }
onedirect(bre,b,r,e);
onedirect(com,c,o,m);
onedirect(cou,c,o,u);
onedirect(frm,f,o,r);
onedirect(ind,i,n,d);
onedirect(inp,i,n,p);
onedirect(inx,i,n,x);
onedirect(lit,l,i,t);
onedirect(mar,m,a,r);
onedirect(opt,o,p,t);
onedirect(out,o,u,t);
onedirect(pag,p,a,g);
onedirect(par,p,a,r);
onedirect(res,r,e,s);
onedirect(sel,s,e,l);
onedirect(ski,s,k,i);
onedirect(sor,s,o,r);
onedirect(sbt,s,u,b);
onedirect(ttl,t,i,t);
onedirect(und,u,n,d);
onedirect(weo,w,e,o);
onedirect(exc,e,x,c);
onedirect(ast,a,s,c);
onedirect(lpt,l,p,t);
onedirect(ajt,a,j,blank)
end {  INITDIRECTS  };




{         INITHOST - INITIALIZE ASCII TO HOST CONVERSION TABLE.
 }

 procedure inithost;                                                   {  }
 var                                                                   {  }
  extch          : char;     {  EXTERNAL CHARACTER  }                  {  }
  intch          : ascii;    {  INTERNAL CHARACTER  }                  {  }
 begin {  INITHOST  }                                                  {  }
 with host[nul] do                                                     {  }
  begin chr74 := false;                                                {  }
  chr76 := true;                                                       {  }
  c := chr( 45)                                                        {  }
  end;                                                                 {  }
 for intch := succ(nul) to del do                                      {  }
 with host[intch] do                                                  { }
  begin extch := chr(0);                                              { }
  while (asc[extch] <> intch) and (extch < chr( 63)) do               { }
   extch := succ(extch);                                              { }
  if asc[extch] = intch                                               { }
   then begin chr74 := false;                                         { }
    chr76 := false;                                                   { }
    c := extch                                                        { }
    end                                                               { }
   else begin extch := chr(0);                                        { }
    while (asc74[extch] <> intch) and (extch < chr( 63)) do           { }
     extch := succ(extch);                                            { }
    if asc74[extch] = intch                                           { }
     then begin chr74 := true;                                        { }
      chr76 := false;                                                 { }
      c := extch                                                      { }
      end                                                             { }
     else begin extch := chr(0);                                      { }
      while (asc76[extch] <> intch) and (extch < chr( 63)) do         { }
       extch := succ(extch);                                          { }
      if asc76[extch] = intch                                         { }
       then begin chr74 := false;                                     { }
        chr76 := true;                                                { }
        c := extch                                                    { }
        end                                                           { }
       else writeln(' OOPS: ',intch:3,'B')                            { }
      end                                                             { }
     end                                                              { }
  end;                                                                { }
host[colon].c := ':'                                                  { }
end {  INITHOST  };                                                   { }




{        INITMONTHS - INITIALIZE THE MONTHS TABLE.
  }

procedure initmonths;




{        ONEMONTH - INITIALIZE ONE MONTH NAME.
 *
 *       PARAM  MON : MONTH NUMBER.
 *              A,B,C : THREE LETTERS OF MONTH NAME.
  }

 procedure onemonth( mon : integer; a,b,c : ascii );
 begin {  ONEMONTH  }
 months[mon][1] := a;
 months[mon][2] := b;
 months[mon][3] := c
 end {  ONEMONTH  };




begin {  INITMONTHS  }
onemonth( 1,j,smalla,smalln);
onemonth( 2,f,smalle,smallb);
onemonth( 3,m,smalla,smallr);
onemonth( 4,a,smallp,smallr);
onemonth( 5,m,smalla,smally);
onemonth( 6,j,smallu,smalln);
onemonth( 7,j,smallu,smalll);
onemonth( 8,a,smallu,smallg);
onemonth( 9,s,smalle,smallp);
onemonth(10,o,smallc,smallt);
onemonth(11,n,smallo,smallv);
onemonth(12,d,smalle,smallc)
end {  INITMONTHS  };
```

```





begin {  INITIALIZE  }
reset(infile);
rewrite(output);                                                      { }
linelimit(output,maxint);  {  UNLIMITED OUTPUT  }                     { }
initmonths;   {  BEFORE INITDATES  }
initasc;                                                              { }
initclass;
initclocks;
initdates;
initdirects;
inithost;                                                             { }
directline := false;
endofinput := false;
eol := true;
errors := false;
gaps[0] := 1;
inchar := blank;
incolumn := 150;
inlength := 0;
inxbase := nil;
inxlast := nil;
linenumber := 0;
linenums := infile↑ in ['0'..'9'];
moreonleft := false;
nblanks := 0;
nchars := 0;
newinline := true;
newoutline := true;
newparagraph := true;
ngaps := 0;
nwords := 0;
outlength := 1;
outline[1].c := blank;
outline[1].nbl := 0;
reinitialize([bre..ill])
end {  INITIALIZE  };








{ ---------------------------------------------------------------- }
{                                                                  }
{                              PROSE                               }
{                              -----                               }
{                                                                  }
{ ---------------------------------------------------------------- }




begin {  PROSE  }
initialize;
nextword;
while not endofinput do
 begin packword; nextword end;
break;
if linecount < infinity
 then begin page(infinity);
  selection[pagenumber] := true;
  advanceform
  end;
if errors then halt(' PROSE ERRORS DETECTED.')                        { }
end {  PROSE  }.
```

Programs

We have received a short version of the Printme program (P-1) from Japan. The program is printed here as a mental exercise for the interested readers who want to clean the rust off their reasoning mechanisms. The only clue we feel we ought to give you is that CHR(48) is meant to be the apostrophe character. The fun things are around the edges...

INFORMATION ENGINEERING COURSE
DIVISION OF ENGINEERING
UNIVERSITY OF TOKYO GRADUATE SCHOOL

Bunkyoku, Tokyo 113 Japan.
Telephone: (03) 812 - 2111

Dear Mr. Mickel:

November 15, 1978

Program Printme (Pascal News #12, P.32) made me write my own version. My Printme is as follows.

Sincerely yours,

Eiiti Wada

Eiiti wada

```
PROGRAM PRINTME(OUTPUT);VAR I:INTEGER;
PROCEDURE P(I:INTEGER);BEGIN CASE I OF
0:WRITE(':WRITE(');
1:WRITE('PROGRAM PRINTME(OUTPUT);VAR I:INTEGER;');
2:WRITE('PROCEDURE P(I:INTEGER);BEGIN CASE I OF');
3:WRITE('END END;BEGIN P(1);WRITELN;P(2);WRITELN;FOR I:=0');
4:WRITE('TO 7 DO BEGIN WRITE(I:1);P(0);WRITE(CHR(48));');
5:WRITE('P(I);WRITE(CHR(48));P(7);WRITELN END;FOR I:=3 TO');
6:WRITE('6 DO BEGIN P(I);WRITELN END END.');
7:WRITE(');');
END END;BEGIN P(1);WRITELN;P(2);WRITELN;FOR I:=0
TO 7 DO BEGIN WRITE(I:1);P(0);WRITE(CHR(48));
P(I);WRITE(CHR(48));P(7);WRITELN END;FOR I:=3 TO
6 DO BEGIN P(I);WRITELN END END.
```

Algorithms

A Perfect Hashing Function A-3

Title: A Class of Easily Computed, Machine Independent, Minimal Perfect Hash Functions for Static Sets

Author: Richard J. Cichelli

Address: Software Consulting Services, 901 Whittier Drive, Allentown, Pa. 18103

Abstract:
A method is presented for computing machine independent minimal perfect hash functions of the form : hash value ← key length + the associated value of the key's first character + the associated value of the key's last character. Such functions allow single probe retrieval from minimally sized tables of identifier lists. Application areas include table look-up for reserved words in compilers and filtering high frequency words in natural language processing. Functions for Pascal's reserved words, Pascal's predefined identifiers, frequently occurring English words, and month abbreviations are presented as examples.

Key Words and Phrases:
Hashing, hashing methods, hash coding, direct addressing, dictionary lookup, information retrieval, lexical analysis, identifier-to-address transformations, perfect hashing functions, perfect hash coding, scatter storage, searching, Pascal, Pascal reserved words, backtracking

CR Categories:
3.7, 3.74, 4.34, 5.25, 5.39

In several recent articles [1], [2] it has been asserted that in general computing minimal perfect hash functions for identifier lists (keys) is difficult. Here, several examples of such functions are shown and an efficient method for computing them is described.

The form of my hash function is:

Hash value ←- key length +
associated value of the key's first character +
associated value of the key's last character.

Example #1: Pascal's Reserved Words

For Pascal's 36 reserved words, the following list defines the associated value for each letter.

A=11, B=15, C=1, D=0, E=0, F=15, G=3, H=15, I=13, J=0, K=0, L=15, M=15, N=13, O=0, P=15, Q=0, R=14, S=6, T=6, U=14, V=10, W=6, X=0, Y=13, Z=0.

(For lookup routines these values are stored in an integer array indexed by the letters. Note: associated values need not be unique.)

The corresponding hash table with hash values running from 2 through 37 is as follows:

DO, END, ELSE, CASE, DOWNTO, GOTO, TO, OTHERWISE, TYPE, WHILE, CONST, DIV, AND, SET, OR, OF, MOD, FILE, RECORD, PACKED, NOT, THEN, PROCEDURE, WITH, REPEAT, VAR, IN, ARRAY, IF, NIL, FOR, BEGIN, UNTIL, LABEL, FUNCTION, PROGRAM.

As an example, consider the computation for "CASE":

(1←-"C") + (0←-"E") + (4←-length("CASE")) = 5

The advantage of hash functions of the above form is that they are simple, efficient, and machine (i.e. character representation) independent. It is also likely that any lexical scanning process will have, as a by-product of its identifier scanning logic, the identifier length and the values of the first and last characters. Two disadvantages of functions of this form are 1) that it requires that no two keys share length and first and last characters and 2) for lists with more than about 45 items segmentation into sublists may be necessary. (This is a result of the limited range of hash values that the functions produce.)

The associated values for each of the letters are computed by the following procedure: 1) Order the identifier list, and 2) Search, by backtracking, for a solution.

The ordering process is twofold. First, order the keys by the sum of the frequencies of the occurrences of each key's first and last letter in the list. For example: "E" occurs 9 times as a first or last letter in the Pascal reserved word list. It is the most frequent letter and thus, "ELSE" is the first word in the search list. "D" is the next most frequent letter, and thus "END" is second. After the words have been put in order by character occurrence frequencies, modify the order of the list such that any word whose hash value is determined by assigning the associated character values already determined by previous words is placed next. Thus, after "OTHERWISE"[1] has been placed as the third element of the frequency ordered list, the hash value of the word "DO" is determined and so it is placed fourth. (I.e. during search, after the placement of the word "END" a value will be associated with "D", and after the placement of the word "OTHERWISE" a value will be associated with "O".) The ordering process causes hash value conflicts during search to occur as early as possible thus pruning the search tree and speeding the computation.

The completely ordered search list for Pascal's reserved words is:

ELSE, END, OTHERWISE, DO, DOWNTO, TYPE, TO, FILE, OF, THEN, NOT, FUNCTION, RECORD, REPEAT, OR, FOR, PROCEDURE, PACKED, WHILE, CASE, CONST, DIV, VAR, AND, MOD, PROGRAM, NIL, LABEL, SET, IN, IF, GOTO, BEGIN, UNTIL, ARRAY, WITH.

The backtracking search procedure then attempts to find a set of associated values which will permit the unique referencing of all the members of the key word list. It does this by trying the words one at a time in order. The backtracking procedure is as follows: If both the first and last letter of the identifier already have associated values, try the word. If either the first or last letter has an associated value, vary the associated value of the unassigned character from zero to the maximum allowed associated value, trying each occurrence. If both letters are as yet unassociated, vary the first and then the second, trying each possible combination. (An exception test is required to catch situations where the first and last letters are the same.) Each "try" tests whether the given hash value is already assigned and, if not, reserves the value and assigns the letters. If all identifiers have been selected, print the solution and halt. Otherwise, invoke the search procedure recursively to place the next word. If the "try" fails, the word is removed in backtracking.

The search time for computing such functions is related to the number of identifiers to be placed, the maximum value which is allowed to be associated with a character, and the density of the resultant hash table. If the table density is one (i.e. a minimal perfect hash) and the maximum associated value is allowed to be the count of distinct first and last letter occurrences (21 for Pascal's reserved words), then the above procedure finds a solution for Pascal's reserved words in about seven seconds on a DEC PDP-11/45 using a straightforward implementation of the algorithm in Pascal. (Without the second ordering, the search required 5½ hours.) If the maximum associated value is limited to 15, as in the above list, the search requires about 40 minutes. (There is no solution with 14 as a maximum value.)

Incorporation of the above hash function into a Pascal cross reference program yielded a 10% reduction in total run time for processing large programs. The method replaced a well coded binary search which was used to exclude reserved words from cross referencing.

[1] Inclusion of the word "OTHERWISE" in Pascal's reserved word list anticipates the acceptance by the Pascal Users Group of the recommendation for a revised CASE construct submitted by its International Working Group for Extensions.

Example #2

The second example is for the list of Pascal's predefined identifiers.
A=15, B=9, C=11, D=19, E=5, F=3, G=0, H=0, I=3, J=0, K=16, L=13, M=1, N=19, O=0, P=18,
Q=0, R=0, S=15, T=0, U=17, V=0, W=10, X=0, Y=0, Z=0.

GET, TEXT, RESET, OUTPUT, MAXINT, INPUT, TRUE, INTEGER, EOF, REWRITE, FALSE, CHR, CHAR,
TRUNC, REAL, SQR; SQRT, WRITE, PUT, ORD, READ, ROUND, READLN, EXP, PAGE, EOLN, COS,
SUCC, DISPOSE, NEW, ABS, LN, BOOLEAN, WRITELN, SIN, PACK, UNPACK, ARCTAN, PRED.

Computation of this function required about seven minutes. Note: since the predefined identifier "ODD" conflicts with "ORD", it was not included in the list.

Example #3: Frequently Occurring English Words

This example uses the word list of [1,3] Search time was less than one second.
A=3, B=15, C=0, D=7, E=0, F=15, G=0, H=10, I=0, J=0, K=0, L=0, M=12, N=13, O=7, P=0,
Q=0, R=12, S=6, T=0, U=15, V=0, W=14, X=0, Y=0, Z=0.
6

I, it, the that, at, are, a, is, to, this, as, he, and, have, in, not, be, but, his,
had, or, on, was, of, her, by, you, with, which, for, from.

Example #4: Month Abbreviations

This example is from [2] . The function's form was modified slightly to:
Hash value <-- associated value of the key's second character +
associated value of the key's third character.
A=4, B=5, C=2, D=0, E=0, F=0, G=3, H=0, I=0, J=0, K=0, L=6, M=0, N=0, O=5, P=1, Q=0,
R=6, S=0, T=6, U=0, V=6, W=0, X=0, Y=5, Z=0.

JUN, SEP, DEC, AUG, JAN, FEB, JUL, APR, OCT, MAY, MAR, NOV.

This form avoids the conflict between "JAN" and "JUN" and takes into account the constant key length. Search time was again well less than one second. Note: the method presented here is applicable to sets up to four times as large as those said to be feasible by the methods described in [2]

Moral:

This article does not have a conclusion, but it does have a moral. In the words of the renowned chess programmer, Jim Gillogly, author of the Technology chess program which was the prototype of the current generation of highly successful chess programs, "When all else fails, try brute force."

References:

[1] Sheil, B. A. Median Split Trees: A Fast Lookup Technique for Frequently Occurring Keys. Comm. ACM 21, 11 (Nov. 1978), 947-958.

[2] Sprugnoli, Renzo. Perfect Hashing Functions: A Single Probe Retrieving Method for Static Sets. Comm. ACM 20, 11 (Nov. 1977), 841-850.

[3] Knuth, D.E. Sorting and Searching, Vol 3, The Art of Computer Programming, 506.

```
program perfect(tty) {  R.J.CICHELLI 2-FEB-79  };
{  COMPUTE A PERFECT HASH TABLE FOR PASCAL RESERVED WORDS  }
 const
   debug = false;
   startsolmax = 1;
   startwordmax = 36;
   maxwordsize = 10;
   maxhashvalue = 50;
   maxreservedwords = 50 {  0 .. N-1  };

 type
   letter = 'A' .. 'Z';
   possiblehashvalues = 0 .. maxhashvalue;
   wordsize = 1 .. maxwordsize;
   aword =  array [wordsize] of char;

   reword = record
               fstlet, lstlet : char;
               length, sortval : integer;
               word :  aword
             end;

   descletter = record usecount, representedby : integer end;

   alfa = packed array [1..10] of char;

 var
   i: integer;
   keys : array [0 .. maxreservedwords] of resword;
   letterdata : array [letter] of descletter;
   taken : array [possiblehashvalues] of boolean;
   wordstodo, solutioncnt, maxsolutns : integer;
   wordcount, numberofreservedwords, maxcharval: integer;
   ptime, pdate : alfa;

 procedure sort(l, r : integer) {  QUICKSORT  };

 var
   i, j, x : integer;
   w: resword;

 begin
   i := l;   j := r;   x := keys[(i+j) div 2].sortval;
   repeat
     while keys[i].sortval < x do i := i + 1;
     while x < keys[j].sortval do j := j - 1;
     if i <= j then
       begin
         w := keys[i];   keys[i] := keys[j];
         keys[j] := w;   i := i + 1;;   j := j - 1;
       end;
   until i > j;
   if l < j then sort(l,j);
   if i < r then sort(i,r);
 end {  SORT  };

  procedure printsolution(numwords: integer);

 var
   i, j: integer;
   ch: char;

 begin
   date(pdate); time(ptime);
   solutioncnt := solutioncnt + 1;
   writeln(tty,' SOLUTION ', solutioncnt);
   writeln(tty,' LETTER  --- REPRESENTED BY ');
   for ch := 'A' to 'Z' do
     writeln(tty,'   ',ch,'    ',letterdata[ch].representedby);
   writeln(tty);
   writeln(tty,' RESERVED WORD LIST');
   write(tty,'     WORD         HASH VALUE');
   if debug then writeln(tty,' FST LST  LENGTH ') else writeln(tty);
   writeln(tty,'     ----         ---- -----');
   if solutioncnt >= maxsolutns then sort(0, numberofreservedwords);
   for i := 0 to numwords do
   with keys[i] do
   begin
     write(tty,' ',i+1:3,' ',word,'  ',sortval);
     if debug then writeln(tty,' ',fstlet,' ',lstlet,' ',length:3)
     else writeln(tty);
   end;
   writeln(tty);
   writeln(tty,' PRINTING AT ',ptime,'  ',pdate);
   if solutioncnt >= maxsolutns then halt;
  end;

 procedure initkeys;
```

```

begin
  keys[0].word :='OTHERWISE ';
 keys[1].word := 'AND       ';
 keys[2].word := 'ARRAY     ';   keys[3].word := 'BEGIN     ';
 keys[4].word := 'PACKED    ';   keys[5].word := 'CASE      ';
 keys[6].word := 'GOTO      ';   keys[7].word := 'CONST     ';
 keys[8].word := 'DIV       ';   keys[9].word := 'DO        ';
 keys[10].word := 'DOWNTO    ';   keys[11].word := 'ELSE      ';
 keys[12].word := 'END       ';   keys[13].word := 'FILE      ';
 keys[14].word := 'FOR       ';   keys[15].word := 'FUNCTION  ';
 keys[16].word := 'IF        ';   keys[17].word := 'IN        ';
 keys[18].word := 'LABEL     ';   keys[19].word := 'MOD       ';
 keys[20].word := 'NIL       ';   keys[21].word := 'NOT       ';
 keys[22].word := 'OF        ';   keys[23].word := 'OR        ';
 keys[24].word := 'PROCEDURE ';   keys[25].word := 'PROGRAM   ';
    keys[27].word := 'RECORD    ';
 keys[28].word := 'REPEAT    ';   keys[29].word := 'SET       ';
 keys[30].word := 'THEN      ';   keys[31].word := 'TO        ';
 keys[32].word := 'TYPE      ';   keys[33].word := 'UNTIL     ';
 keys[34].word := 'VAR       ';   keys[35].word := 'WHILE     ';
 keys[26].word := 'WITH      ';
 numberofreservedwords := 35;
end;

procedure clearletters;

var
  ch : char;

begin
 for ch := 'A' to 'Z' do
   with letterdata[ch] do
     begin usecount := 0;   representedby := 0 end;
end;

procedure setkeys;

var
 i, j: integer;

begin
 for i := 0 to numberofreservedwords do
   with keys[i] do
     begin
       fstlet := word[1];
       j := maxwordsize;
       while word[j] = ' ' do j := j - 1;
       lstlet := word[j];
       length := j;
       sortval := 0;
     end;
end;

procedure conflicts;

var
  nogood: boolean;
  i, j: integer;
  ch1, ch2: char;

begin
  nogood := false;
  clearletters;
  setkeys;
  for i := 0 to numberofreservedwords do
    begin
      with keys[i] do
        begin
          ch1 := fstlet;
          ch2 := lstlet;
        end;
      for j := i+1 to numberofreservedwords do
        begin
          if keys[i].length = keys[j].length
          then
            begin
              with keys[j] do
                begin
                  if ((ch1 = fstlet) and (ch2 = lstlet)) or
                      ((ch2 = fstlet) and (ch1 = lstlet))
                  then
                    begin
                      writeln(tty,' ',keys[i].word,' CONFLICTS WITH ',
                        keys[j].word);
                      nogood := true;
                    end;
                end;
            end
        end;
    end;
   if nogood then halt else writeln(tty,' NO CONFLICTS ');
end;

procedure order;

var
  i: integer;

begin
 clearletters;
 setkeys;
 for i := 0 to numberofreservedwords do
   with keys[i] do
     begin
       letterdata[fstlet].usecount := letterdata[fstlet].usecount + 1;
       letterdata[lstlet].usecount := letterdata[lstlet].usecount + 1;
     end;
 for i := 0 to numberofreservedwords do
   with keys[i] do
     sortval := -(letterdata[fstlet].usecount + letterdata[lstlet].usecount);
 sort(0, numberofreservedwords);
end;

procedure reorder;

var
 i, j, mark : integer;

begin
  clearletters;
  setkeys;
  mark := 1;
  for i := 0 to numberofreservedwords do
   if keys[i].sortval = 0 then
     begin
       with keys[i] do
         begin
           sortval := mark;
           mark := mark + 1;
           letterdata[fstlet].representedby := 1;
           letterdata[lstlet].representedby := 1;
         end;
       for j := i+1 to numberofreservedwords do
         if keys[j].sortval = 0 then
         begin
           with keys[j] do
             begin
               if (letterdata[fstlet].representedby = 1) and
                  (letterdata[lstlet].representedby = 1) then
                 begin
                   sortval := mark;
                   mark := mark + 1
                 end;
             end;
         end;
     end;
   sort(0, numberofreservedwords);
end;

procedure init;

var
  i, j: integer;
  ch :char;
  w : resword;

begin  { INIT  }

  wordcount := 0;
  maxsolutns := startsolmax;   wordstodo := startwordmax - 1;
  solutioncnt := -1;
  initkeys;
  conflicts;
  order;
  reorder;
  maxcharval := 0;
  for ch := 'A' to 'Z' do maxcharval := maxcharval
    + letterdata[ch].representedby;
 setkeys;
 printsolution(numberofreservedwords);
 clearletters;
end;

 procedure addword;

var
 ch1, ch2: char;
 len, repfirstlet, replastlet : integer;

procedure try;

var
  hsh: integer;

begin
  hsh := len + letterdata[ch1].representedby +
         letterdata[ch2].representedby;
  if not taken[hsh]
  then
    begin
      taken[hsh] := true;
      letterdata[ch1].usecount := letterdata[ch1].usecount + 1;
      letterdata[ch2].usecount := letterdata[ch2].usecount + 1;
      keys[wordcount].sortval := hsh;
      wordcount := wordcount + 1;
      if wordcount > wordstodo
      then printsolution(wordstodo)
      else addword;
      wordcount := wordcount - 1;
      letterdata[ch2].usecount := letterdata[ch2].usecount - 1;
      letterdata[ch1].usecount := letterdata[ch1].usecount - 1;
      taken[hsh] := false;
    end
  end {  TRY  };

begin {  ADDWORD  }
  with keys[wordcount] do
    begin
      ch1 := fstlet;
      ch2 := lstlet;
      len := length;
    end;
  if letterdata[ch1].usecount > 0
  then
    if letterdata[ch2].usecount > 0
    then
      try {  BOTH CHARACTERS SPECIFIED  }
    else
      for replastlet := 0 to maxcharval do
        begin {  FIRST CHARACTER ONLY SPECIFIED  }
          letterdata[ch2].representedby := replastlet;
          try;
        end
   else
     if letterdata[ch2].usecount > 0
     then
       for repfirstlet := 0 to maxcharval do
         begin {  LAST LETTER ONLY SPECIFIED  }
           letterdata[ch1].representedby := repfirstlet;
           try;
         end
     else
       for repfirstlet := 0 to maxcharval do
         begin {  BOTH LETTERS UNSPECIFIED  }
           letterdata[ch1].representedby := repfirstlet;
           if ch1 = ch2 then try else
           for replastlet := 0 to maxcharval do
             begin
               letterdata[ch2].representedby := replastlet;
               try;
             end;
         end;
 end;
begin
  writeln(tty,' FIND PERFECT HASH FUNCTIONS FOR RESERVED WORDS.');
 date(pdate);    time(ptime);
 writeln(tty,' STARTING AT ',ptime,' ON ',pdate);
 writeln(tty,' SOLVING FOR ',startsolmax,' SOLUTIONS');
 writeln(tty,' PLACING ',startwordmax,' WORDS');
 for i := 0 to maxhashvalue do taken[i] := false;
 {   ASSURE THAT THE TABLE HAS NO OPEN LOCATIONS    };
 for i:= 39 to maxhashvalue do taken[i] := true;
  init;
 time(ptime);
 writeln(tty,' STARTING SEARCH AT ',ptime);
 {  SPECIAL CODE TO DO MAXCHARVAL == 15  }
 maxcharval := 15;  {  14 DOESN'T WORK  }
 addword;
 time(ptime);
 writeln(tty,' NO SOLUTIN AT ',ptime);
end.
```

Articles

A CONTRIBUTION TO MINIMAL SUBRANGES

Laurence V. Atkinson
University of Sheffield
England

Introduction

Two topics which have received recent attention in Pascal News are the evaluation of boolean expressions [3, 8, 10, 11, 14] and extended subranges [4, 5, 7]. Two articles [1, 2], prompted largely by the programs presented during the aforementioned discussion, show how a state transition approach to multi-exit loops avoids issues of boolean expression evaluation and, as an added bonus, facilitates minimal subranges. Wherever feasible in a Pascal program the range of values that a variable is permitted to take should be as small as possible. This aids program transparency (the declaration is more informative), improves efficiency (see [13]) and increases security (the assignment of illogical values is more readily detectable, both at compile-time and at run-time).

A recent letter from Judy Bishop [6] suggests that the relevance of state transition loops to minimal subranging is not fully appreciated. This article emphasises this particular aspect.

Bishop's example

The example which started all this discussion was a linear search algorithm presented by Barron and Mullins [3]. A state transition implementation is given in [1]. Judy Bishop gives a similar solution in [6] but implies that a state transition approach necessitates an extended subrange. This is not so!

She identifies three mutually exclusive states:

$(i \leqslant n) \wedge (a_i \neq item)$ => searching
$(i \leqslant n) \wedge (a_i = item)$ => item found
$i > n$ => item absent

and produces a solution of the form shown in figure 1.

```
var  a : array [1 .. n] of ... ;
     i : 1 .. nplus1;
     state : (searching, absent, found);
  . . .
i := 1;    state := searching;
repeat
   if i > n then state := absent else
      if a[i] = item then state := found else
         i := i + 1
until state <> searching
```

Figure 1.

The extended subrange for i is necessitated only by the states chosen. In this example it is impossible for n to be less than 1 (for then the array declaration would not compile) so testing i>n immediately upon entry to the loop is pointless. Instead we should make a[i]=item the first test and then test i=n before incrementing i. Thus the states which should be chosen are

$(i < n) \wedge (a_i \neq item)$ => searching
$(i \leqslant n) \wedge (a_i = item)$ => item found
$(i = n) \wedge (a_i \neq item)$ => item absent

and the corresponding solution is in figure 2. Notice that i now takes its minimal subrange: the index range of the array.

In this example the index type of the array is a subrange type which can be extended and the table is assumed to be full. We now examine the state transition approach in circumstances where the array index type is not a subrange and where the table may be empty.

```
var  a : array [1 .. n] of ... ;
     i : 1 .. n;
     state : (searching, absent, found);
  . . .
i := 1;    state := searching;
repeat
   if a [i] = item then state := found else
      if i = n then state := absent else
         i := i + 1
until state <> searching
```

Figure 2.

Full range index type

When the index type of an array is a subrange type we are able to extend this subrange for a subscript variable (but note that minimal subranging is particularly important for array subscripts). If the index type of an array is not a subrange type but a full type, such as char, then we have no choice; we cannot extend the range. This point was raised by John Strait [12]. As shown in [1], the fact that a state transition approach does not incur an extension of the index type makes the technique directly applicable. This is illustrated in figure 3.

Table possibly empty

A common technique is to use a variable to record the number of entries a table currently contains. For a table with index range 1..n the number of entries (say,m) may be anywhere in the range 0 to n. Hence, 0..n is the appropriate subrange for m. This does not affect consideration of the subscript work-variable: this should sensibly refer only to actual entries and so should never take a value outside the range 1 to m. Its full range is therefore 1 to max(m) and so its minimal subrange is 1..n.

The states are

$(m > 0) \wedge (i < m) \wedge (a_i \neq \text{item}) \Rightarrow$ searching
$(m > 0) \wedge (i \leqslant m) \wedge (a_i = \text{item}) \Rightarrow$ item found
$(m = 0) \vee (i = m) \wedge (a_i \neq \text{item}) \Rightarrow$ item absent

and the program is in figure 4.

Alternatively, some other information may record whether or not the table is occupied, as in figure 5. This will probably be so, whatever the search algorithm, if the index type of the array is a full range type.

```
const  firstch = ... ;   lastch = ... ;
  . . .
var  a : array [char] of ... ;
     ch : char;
     state : (looking, exhausted, located);
  . . .
ch := firstch;   state := looking;
repeat
   if a[ch] = item then state := located else
      if ch = lastch then state := exhausted else
         ch := succ (ch)
until state <> looking
```

Figure 3.

```
var  a : array [1 .. n] of ... ;
     i : 1 .. n;
     noofentries : 0 .. n;
     state : (searching, absent, found);
  . . .
if noofentries > 0 then
begin
   i := 1;   state := searching;
   repeat
      if a[i] = item then state := found else
         if i = noofentries then state := absent else
            i := i + 1
   until state <> searching
end else
   state := absent
```

Figure 4.

```
   . . .
   occupancy : (empty, occupied);
   . . .
case occupancy of
   occupied :
      begin
         i := 1;   state := searching;
           . . .
      end;
   empty :
      state := absent
end { case }
```

Figure 5.

Efficiency

It would be inappropriate to end this discussion without reference to the efficiency considerations raised by Wilsker [14]. He stresses the reduction in execution time achieved by the data sentinel approach to linear search as advocated by Knuth [9]. I have some sympathy with this view but my concern, both here and in [1], is not with the algorithm itself, but the statement of the algorithm in Pascal.

Conclusions

Enumerated and subrange types are two of the most important features of Pascal. Their contribution to transparency, security and efficiency is often not fully appreciated. Their under-utilisation is one of the (many!) features I repeatedly criticise when reviewing Pascal books.

Minimal subranging in Pascal is desirable. One benefit of a state transition approach to dynamic processes, as described here and in [1] and [2], is that minimal subranging can be achieved.

References

[1] L.V. Atkinson, "Know the state you are in", Pascal News, 13, 66-69, 1978.

[2] L.V. Atkinson, "Pascal scalars as state indicators", Software-Practice and Experience (to appear), 1979.

[3] D.W. Barron and J.M. Mullins, "What to do after a while", Pascal News, 11, 48-50, 1978.

[4] J.M. Bishop, "Subranges and conditional loops", Pascal News, 12, 37-38, 1978.

[5] J.M. Bishop, Letter to John Strait, Pascal News, 12, p51, 1978.

[6] J.M. Bishop, Letter to Michael Irish, Pascal News, 13, p82, 1978.

[7] K. Fryxell, Letter to the editor, Pascal News, 13, p80, 1978.

[8] T.M.N. Irish, "What to do after a while ... longer", Pascal News, 13, p65, 1978.

[9] D.E. Knuth, "Structured programming with goto statements", Computing Surveys, 6, 261-301, 1974.

[10] M.W. Roberts and R.N. Macdonald, "A resolution of the boolean expression evaluation question", Pascal News, 13, 63-65, 1978.

[11] A.H.J. Sale, "Compiling boolean expressions", Pascal News, 11, 76-78, 1978.

[12] J. Strait, Letter to Judy Bishop, Pascal News, 12, p51, 1978.

[13] J. Welsh, "Economic range checks in Pascal", Software-Practice and Experience, 8, 85-97, 1978.

[14] R.A. Wilsker, "On the article : what to do after a while", Pascal News, 13, 61-62, 1978.

(* Received 79/04/04 *)

****

Reprinted with the permission of the author,
from
Australian Computer Science Communications,
Volume 1 Number 1, March 1979.

(* Received 79/05/15 *)

A Note on Scope, One-Pass Compilers, and Pascal

A.H.J. Sale
Department of Information Science, University of Tasmania

1. Introduction

Very few Pascal compilers correctly implement the scope rules of Pascal. Partly this may be due to their obscurity as some of the key statements are buried in the introduction to the *Pascal Users Manual*, and partly it may be due to the frequent use of one-pass recursive descent compilation techniques. However, with the publication of the draft *Pascal Standard* in issue 14 of *Pascal News*, the scope rules have been clarified and it is therefore appropriate to see how the compilers may be made to conform. The following program fragment illustrates the sort of error that should be detected.

```
program NonStandard(output);
    type
        state = record
                    status : (defined,undefined);
                    value  : integer
                end;
    ...
    procedure InnerScope;
        var
            ageofperson  : state;        {meant to be the type above}
            state : (scanning,found,notpresent);
        begin
            ...     {including references to variable state}
        end;
    begin
        ...
    end.
```

Most Pascal compilers will compile this program, attaching the first use of *state* in *InnerScope* to its outer definition. In fact, this use is inside the scope of the second definition and is in error on two counts: (1) it is an instance of use preceding definition, and (2) *state* is not a type-identifier in this scope.

2. The relevant rules

The relevant rules laid down by the *Pascal Standard* may be paraphrased as follows:

2.1 The scope of an identifier extends over the whole of the program, procedure, function, or record definition in which it is declared with the exception noted in 2.2.
2.2 If an identifier is defined in a procedure, function, or record definition, then that scope and all enclosed scopes are excluded from the scope of any identifier of the same name in an enclosing scope. {*the redefinition rule* }
2.3 No two identifiers may have the same name in a scope. {*uniqueness of association* }
2.4 The definition of an identifier must precede its use, with the exception of pointer-type definitions and forward-declared procedures and functions (see *Standard* for the exceptions).

Note that I use *identifier* as meaning a handle attached to a Pascal object, and *name* as the character-string itself. Thus *Arthur* is the identifier to which I respond in appropriate contexts, but other people have the same name.

3. Outline of the algorithm

Consider a particular scope S. If we denote the point of definition by D, and uses of an identifier by U, then the allowable pattern is illustrated by

```
scope S:  (...
            D
            ...
            U
            ...
            ...
            U
            ...)
```

Consequently, I can formulate the pre-condition R which must hold immediately before the definition of the identifier at D:
R = "No occurrences of the name of the identifier may have occurred in accessible scope between the start of S and the point of definition at D."
This follows from rules 2.1,2.3 and 2.4. Rule 2.2 is brought in by the reference to "accessible scope".
Consequently, we may incorporate the precondition in a one-pass compiler by checking at this point. We search the symbol-table for any accessible identifier of the same

name before entering the new use. There are three distinct possibilities:

3.1 There is no identifier of this name. This means that no previous definitions have occurred in accessible scope, and any attempted uses have already been detected as errors (references to unknown identifiers).
3.2 There is an identifier of the same name declared at this scope level. This is an error as it violates rule 2.3 (name already defined for this scope).
3.3 There is an identifier of the same name at an enclosing scope level. This is therefore a redefinition of the name. The problem that arises is that uses of this name preceding D will have been bound to the outer definition of the name, and some may have occurred in the forbidden region.

The problem of 3.3 may be handled by associating a unique symbol with each new scope as it is encountered, such that the symbols are ordered. Each identifier in the symbol-table then carries the symbol indicating its last occurrence. When the pre-condition search is made, if the table-symbol is earlier in the ordering than the current-scope-symbol, then no use has been made of the name in the forbidden region. If the table-symbol is equal to or follows the current-scope-symbol, then references to the identifier have occurred in the forbidden region and an error has occurred.

The simplest implementation is to make the scope-symbol a natural number stating at 0 for the program block and incremented for each new scope. It would be rare for programs to exceed even the limits of integers in 16-bit machines!

4. The exceptions

The type-identifier of pointer-type definition may occur anywhere in the type part; this relaxes rule 2.4. In all implementations of which I am aware, there are no properties of pointers (such as bit-size) which depend on their bound types, though this is possible. Therefore, the type-definitions may be compiled normally with the exception that all references to type-identifiers are deferred, and examined only at the close of the type-part. This defers all occurrences of the type-identifiers to *virtual occurrences* at the close of the type-part, and satisfies rule 2.4 and the algorithm requirements.

A full definition of a forward-declared procedure may follow a use of the procedure. However, the forward-declaration is a defining occurrence of the procedure identifier, and incorporates a pseudo-scope for the parameter list. Within the parameter list only references to types and definitions of variables can occur. Application of the algorithm is still necessary to detect uses before definition and duplicate uses of names. However, any names so introduced are not accessible in the intervening scopes between the forward-declaration and its associated body, and the algorithm will still work when the parameter list is again accessible in the newly created scope of the body. (It is not neccessary to alter the parameter list scope-symbols to the newly created one, but it can be done.)

Functions may be treated identically. The *Pascal Standard* does not prohibit re-defining the function-designator name as an identifier local to the function, but the resulting function-definition must then be non-standard as it cannot assign a value to the function.

5. Conclusions

The scope rules set out in section 2 and now incorporated into the draft *Pascal Standard* are sufficient to permit even one-pass compilers to reject incorrect programs. The suggested algorithm adds an overhead at every defining occurrence, but since uses exceed definitions in general it may not be too expensive in time to implement. In any case, what price can be put on correctness?

6. References

Addyman, A (1979): "The BSI/ISO Working Draft of Standard Pascal by the BSI DPS/13/4 Working Group", Pascal News, no 14, January 1979, pp 4-60.

Jensen, K. & Wirth, N. (1974): "Pascal User Manual and Report", Springer-Verlag, pp 8, 69-71, 136, 150, 155-156 (Second corrected Edition).

Pascal-I - Interactive, Conversational Pascal-S

Richard J. Cichelli
901 Whittier Drive
Allentown, Pa. 18103

PASCAL-I is a version of the Wirth PASCAL-S (PASCAL subset) system designed to interact with the terminal user. The system contains a compiler, interpreter, text editor, formatter and a run-time debugging system. The compiler compiles the source into a stack code which is interpreted. After program changes, the compiler recompiles only the minimal set of affected procedures. The compiler also automatically formats the program upon compilation and recompilation. Extensive on-line documentation is available. The HELP command will give either a list of all the commands with short descriptions or will give a detailed description of any command (s) specified. Compiler error messages are detailed and sometimes include recommendations for possible fixes. The program source text is stored to allow interaction with the run time system on the source level.

All editting commands (except the GET file and SAVE file commands) follow the PASCAL scope rules. (i.e. the LIST command defaults to listing only the block being edited.) Strings can be searched for and changed. The REPEAT command reapplies the last edit command. There are no line numbers; the editting scope is always very local, and none seem needed nor desired. The edit pointer can be moved from procedure to procedure, to the top or bottom of any of the three sections of a PASCAL block (HEADER, DECLARATIONS, and BODY), and up and down within the block. Text lines or entire procedures can be inserted, deleted or moved. A tree structured listing of procedure relationships is produced by the STRUCTURE command.

The run time system allows the user to execute his program and to suspend execution at any time during execution. Breakpoints can be set, cleared or ignored. Execution limits can be set (statements executed, instructions executed and output lines). A user abort entered from the terminal will also suspend execution of the users program (but not terminate PASCAL-I). Execution errors and I/O errors will also suspend the program (not terminate it).

Once execution is suspended, the user has several options. He may use the PMD command to examine any of the simple variables in the stack and the contents of the I/O buffers and may display the recent execution history of his program. He may also enter code for immediate execution! Immediate code may be anything from a PASCAL-S statement to an entire block (without the header or any blocks declared inside it). One block of immediate code may be stored for each procedure and can be executed anytime the program is suspended within that procedure.

Part of the research involved in creating PASCAL-I was to test whether procedure oriented languages like PASCAL could be easily used interactively. Some language designers have suggested that only line oriented languages such as APL and BASIC could be used. The argument was that highly structured languages would inhibit programmer interaction. We argue that disciplined design structure is esential for reliable software development. PASCAL-I makes such discipline implicit in its commands and their scope. When you edit a PASCAL-S program with PASCAL-I, you modify text within a procedure. Error correction and most other program interaction is oriented towards the current statement in the current procedure.

We believe that PASCAL-I's automatic formatting and procedure orientation overcome any limitations that PASCAL might have as a conversational language, and that the discipline imposed by languages such as PASCAL is essential for reliable software design and implementation.

```
     B[ottom] - Set pointer to bottom of environment.
      BR[eak] - Set breakpoints.
        BY[e] - Exit PASCAL-I.
     C[hange] - Change strings.
   COM[pile] - Compile program.
 CO[ntinue] - Continue execution of program.
    DE[lete] - Delete a block.
      D[own] - Move edit pointer down.
      DU[mp] - Dump internal tables (debug command).
      E[dit] - Begin editting a specified block.
       EN[d] - Exit PASCAL-I.
     ERA[se] - Erase a line of text.
    ER[rors] - List compilation errors.
  EX[ecute] - Execute program.
      F[ind] - Find strings.
       G[et] - Get a file.
      H[elp] - Print this list.
   HI[story] - Display recent trace history.
    IG[nore] - Ignore breakpoints.
    I[nsert] - Insert a line.
      LIM[it] - Set execution limits.
      L[ist] - List program.
  M[essage] - List selected error messages.
  MON[itor] - Display variable changes.
      MO[ve] - Move lines of text.
    N[oveto] - Stop requesting veto responses.
O[verwrite] - Overwrite line of text.
       PM[d] - Post mortem dump.
     P[rint] - Print current line (and subsequent lines).
    R[epeat] - Repeat previous command.
RES[truct] - Move a block.
       SA[ve] - Save program to a file.
    S[tatus] - Display current status.
STR[ucture] - List program structure.
        T[op] - Set pointer to top of environment.
      TR[ace] - Set trace flag.
        U[p] - Move edit pointer up.
      V[eto] - Request veto responses on changes.
            $ - Execute PASCAL statements.
            ? - Gives explanation of command errors.
```

SAMPLE SESSION

```
COMMAND-copy,queens
 PROGRAM QUEENS(OUTPUT);
(* EIGHT QUEENS PROBLEM - PLACE EIGHT HOSTILE QUEENS
ON A CHESS BOARD SUCH THAT NONE ATTACKS ANOTHER.
THIS PROGRAM IS FOR DEMONSTRATION PURPOSES.
IT CONTAINS BOTH SYNTAX AND LOGIC ERRORS. *)
VAR BOARD
:ARRAY[0..7]OF INTEGER;COL:ARRAY[0..7]OF
BOOLEAN;UP:ARRAY[0..14]OF BOOLEAN;DOWN:ARRAY
[-7..+7]OF BOOLEAN;PROCEDURE PRINTBOARD;VAR R
:INTEGER;BEGIN FOR R:=0 TO 7 DO WRITE(# #,
BOARD[R]:2);WRITELN;END(* PRINTBOARD *);
PROCEDURE GENERATE(R:INTEGER);VAR C:INTEGER;
PROCEDURE SETSQUARE(R,C:INTEGER;VAL:BOOLEAN);
BEGIN COL[C]:=VAL;UP[R+C]:=VAL;DOWN[R-C]
:=VAL;END(* SETSQUARE *);BEGIN(* GENERATE *)
FOR C:=0 TO 7 DO IF COL[C]AND UP[R+C]AND
DOWN[R-C]THEN BEGIN(* SQUARE FREE *)SETSQUARE
(R,C,FALSE);IF R=7 THEN(* BOARD FULL *)
PRINTBOARD ELSE GENERATE(R+1);SETSQUARE(R,C,
TRUE);END END(* GENERATE *);PROCEDURE
INITIALIZE;BEGIN FOR I:=0 TO 7 DO COL[I]:=
TRUE;FOR I:=0 TO 14 DO UP[I]:=TRUE;FOR I:=
-7 TO+7 DO DOWN[I]:=TRUE;END(* INITIALIZE *)
;BEGIN(* QUEENS *)INITIALIZE;GENERATE(0);
END(* QUEENS *).
```

List the input - messy rendition of the notorious queen's problem

```
COMMAND-pascali,queens
- PASCALI (1.1.79)
```

Invoke Pascal-I

```
PROGRAM QUEENS CONTAINS 5 BLOCKS
THE FOLLOWING BLOCKS CONTAIN ERRORS:
  QUEENS.INITIALIZE
```

The edit pointer is automatically set to the first procedure with errors

```
:list m

 PROCEDURE INITIALIZE;


*  BEGIN
     FOR I := 0 TO 7 DO
         '0
       COL[I] := TRUE;
           '0'26
     FOR I := 0 TO 14 DO
         '0
       UP[I] := TRUE;
          '0'26
     FOR I := - 7 TO + 7 DO
         '0
       DOWN[I] := TRUE;
            '0'26
   END (* INITIALIZE *);
```

List the procedure giving full error messages

```
 EXPLANATIONS OF ERROR CODES:
   0: THE DESIGNATED IDENTIFIER HAS NOT BEEN
      DECLARED.
  26: THE TYPE OF AN INDEX EXPRESSION MUST BE
      IDENTICAL TO THE INDEX TYPE SPECIFIED IN
      THE ARRAY DECLARATION.

 :edit * d

 :i var i: integer;
```

Forgot to declare i. - edit the declarations ... and insert the declaration.

```
 :comp
 2 BLOCKS RECOMPILED
```

Recompile - system compiles minimum that assures consistency

```
 :edi queens
```

Let's look at the whole thing.

```
 :list a

  PROGRAM QUEENS(OUTPUT);
  (* EIGHT QUEENS PROBLEM - PLACE EIGHT HOSTILE QUEENS
  ON A CHESS BOARD SUCH THAT NONE ATTACKS ANOTHER.
  THIS PROGRAM IS FOR DEMONSTRATION PURPOSES.
  IT CONTAINS BOTH SYNTAX AND LOGIC ERRORS. *)
```

... and list all of it - nicely formatted too.

```
VAR
  BOARD: ARRAY [0..7] OF INTEGER;
  COL: ARRAY [0..7] OF BOOLEAN;
  UP: ARRAY [0..14] OF BOOLEAN;
  DOWN: ARRAY [- 7..+ 7] OF BOOLEAN;


PROCEDURE PRINTBOARD;

  VAR
    R: INTEGER;


  BEGIN
    FOR R := 0 TO 7 DO
      WRITE(# #, BOARD[R]: 2);
    WRITELN;
  END (* PRINTBOARD *);

PROCEDURE GENERATE(R: INTEGER);

  VAR
    C: INTEGER;


  PROCEDURE SETSQUARE(R, C: INTEGER; VAL: BOOLEAN);


    BEGIN
      COL[C] := VAL;
      UP[R + C] := VAL;
      DOWN[R - C] := VAL;
    END (* SETSQUARE *);

  BEGIN (* GENERATE *)
    FOR C := 0 TO 7 DO
      IF COL[C] AND UP[R + C] AND DOWN[R - C] THEN
        BEGIN (* SQUARE FREE *)
          SETSQUARE(R, C, FALSE);
          IF R = 7 THEN (* BOARD FULL *)
            PRINTBOARD
          ELSE
            GENERATE(R + 1);
          SETSQUARE(R, C, TRUE);
        END
  END (* GENERATE *);
```

Check out the fancy style for formatting comments.

```
 PROCEDURE INITIALIZE;

  VAR
    I: INTEGER;


  BEGIN
    FOR I := 0 TO 7 DO
      COL[I] := TRUE;
    FOR I := 0 TO 14 DO
      UP[I] := TRUE;
    FOR I := - 7 TO + 7 DO
      DOWN[I] := TRUE;
  END (* INITIALIZE *);

*BEGIN (* QUEENS *)
   INITIALIZE;
   GENERATE(0);
 END (* QUEENS *).
```

The asterisk denotes the edit pointer.

```
:structure

* 1     QUEENS
  2       PRINTBOARD
  2       GENERATE
  3         SETSQUARE
  2       INITIALIZE
```

Give an overview of the program. Here the asterisk shows the current block.

```
:restruct printboard w generate
```

Only Generate calls Printboard so let's make it local.

```
:str

* 1     QUEENS
  2       GENERATE
  3         PRINTBOARD
  3         SETSQUARE
  2       INITIALIZE
```

```
:exec
5 BLOCKS RECOMPILED
INTERPRETING QUEENS


EXECUTED 24895 STEPS IN 3120 STATEMENTS.
```

Ok. Let's run it. Restructuring makes everything recompile. It's all in memory so you get fast response.

```
HALT AT:  *         WRITE(# #, BOARD[R]: 2);
IN: QUEENS.GENERATE.PRINTBOARD
BECAUSE OF UNDEFINED VALUE IN EXPRESION.

USER INPUT FILE BUFFER - EOLN: TRUE; - EOF: FALSE
USER OUTPUT BUFFER:
```

Who's undefined?
Let the symbolic post mortum dump print.

```
QUEENS.GENERATE.PRINTBOARD
CALLED AT THE 7TH LINE OF THE BODY OF QUEENS.GENERATE
I.E. >>                PRINTBOARD
  R           =            0
```

Here's the traceback.

```
QUEENS.GENERATE
CALLED AT THE 9TH LINE OF THE BODY OF QUEENS.GENERATE
I.E. >>                GENERATE(R + 1);
  C           =            3
  R           =            7

QUEENS.GENERATE
CALLED AT THE 9TH LINE OF THE BODY OF QUEENS.GENERATE
I.E. >>                GENERATE(R + 1);
  C           =            1
  R           =            6

QUEENS.GENERATE
CALLED AT THE 9TH LINE OF THE BODY OF QUEENS.GENERATE
I.E. >>                GENERATE(R + 1);
  C           =            6
  R           =            5

QUEENS.GENERATE
CALLED AT THE 9TH LINE OF THE BODY OF QUEENS.GENERATE
I.E. >>                GENERATE(R + 1);
  C           =            2
  R           =            4

QUEENS.GENERATE
CALLED AT THE 9TH LINE OF THE BODY OF QUEENS.GENERATE
I.E. >>                GENERATE(R + 1);
  C           =            5
  R           =            3


QUEENS.GENERATE
CALLED AT THE 9TH LINE OF THE BODY OF QUEENS.GENERATE
I.E. >>                GENERATE(R + 1);
  C           =            7
  R           =            2

QUEENS.GENERATE
CALLED AT THE 9TH LINE OF THE BODY OF QUEENS.GENERATE
I.E. >>                GENERATE(R + 1);
  C           =            4
  R           =            1

QUEENS.GENERATE
CALLED AT THE 3RD LINE OF THE BODY OF QUEENS
I.E. >>    GENERATE(0);
  C           =            0
  R           =            0


:$writeln(r, c);$

        0          3
```

Is "R" or "C" undefined? ... nope.

```
:$writeln(board[3]);$


EXECUTION ERROR IN IMMEDIATE CODE.
```

What about the BOARD[3]? If this is a solution, it should have a value. ... ah! Let's doublecheck what was happening.

```
:history
H>>       END (* SETSQUARE *);
 LEAVING: QUEENS.GENERATE.SETSQUARE
H>>             IF R = 7 THEN (* BOARD FULL *)
H>>               PRINTBOARD
```

```
ENTERING: QUEENS.GENERATE.PRINTBOARD                Yes - it thinks it has an
H>>       BEGIN                                     answer but it doesn't.
H>>         FOR R := 0 TO 7 DO
H>>           WRITE(# #, BOARD[R]: 2);

:edit generate

:find /begin/                                       We've got to put the queens on
*  BEGIN (* GENERATE *)                             the board if we want them to
OK?n                                                print.
          BEGIN (* SQUARE FREE *)
OK?y                                                Here.  This should do it.

:i board[c] := r;

:edit printbord
WHAT?                                               What did I do, misspell it?

:?
NO SUCH BLOCK.

:edit printboard                                    Yes.

:bot                                                Let's stop it after it prints
                                                    some answers - Set a breakpoint.
:break 1 s

:break
BRKPNT 1 AT:  *    END (* PRINTBOARD *);            - right at the end of printboard

:ignore 3                                           This should get us three answers

:execute                                            Do it.
1 BLOCK RECOMPILED
INTERPRETING QUEENS

 0  6  4  7  1  3  5  2                             Answers!
 0  6  3  5  7  1  4  2
 0  5  7  2  6  3  1  4
 BREAKPOINT 1 AT:                                   Here's the breakpoint.
B>> *    END (* PRINTBOARD *);
IN: QUEENS.GENERATE.PRINTBOARD                      Hey!  Shouldn't these solutions
                                                    look like they are increasing
EXECUTED 39147 STEPS IN 4982 STATEMENTS.            in value?


:monitor board                                      Let's check out what's happen-
                                                    ing to the board.
:limit 100 s

:help status                                        Now where were we?
THE STATUS COMMAND IS USED TO REPORT THE            Should I use the status command.
CURRENT STATE OF THE USER SESSION. THE EDITING,
BREAKPOINT, TRACING, AND LIMIT CONDITIONS ARE
REPORTED.  THERE ARE NO PARAMETERS ASSOCIATED
WITH THIS COMMAND.


:sta
EDITING BODY OF QUEENS.GENERATE.PRINTBOARD

BRKPNT 1 AT:  *    END (* PRINTBOARD *);


STATEMENT LINE LIMIT: 100                           Fair enough.
   INSTRUCTION LIMIT: 100000
   OUTPUT LINE LIMIT: 1000
   VETO =       TRUE
MONITORING:
           BOARD
:continue                                           Ok.  Let's start this up again.
RESUMING QUEENS.GENERATE.PRINTBOARD

M>>            BOARD[C] := R;
M>>            ! := 5                               What?
M>>            BOARD[C] := R;
M>>            ! := 3
 STATEMENT LIMIT AT:                                Statement limit stopped us.
S>>          BEGIN (* SQUARE FREE *)
IN: QUEENS.GENERATE

EXECUTED 791 STEPS IN 101 STATEMENTS.


                                                    Clear the monitors.
:monitor

:continue s                                         Let's look at it one statement
RESUMING QUEENS.GENERATE                            at a time.
```

```
R>>            BOARD[C] := R;
R>>            SETSQUARE(R, C, FALSE);              <CR> gets the next statement
R>>       BEGIN                                     to execute.
R>>         COL[C] := VAL;
R>>         UP[R + C] := VAL;
R>>         DOWN[R - C] := VAL;
R>>       END (* SETSQUARE *);                      Enough of this.
R>>            IF R = 7 THEN (* BOARD FULL *)
R>>            ELSE
lim 10000 s

:cont                                               Go on.
RESUMING QUEENS.GENERATE

 0  4  7  5  2  6  1  3                             Isn't this the first solution?
 BREAKPOINT 1 AT:
B>>       END (* PRINTBOARD *);
IN: QUEENS.GENERATE.PRINTBOARD

EXECUTED 3225 STEPS IN 412 STATEMENTS.

:$                                                  Let's look at that board again.
 var i: integer;
 begin
  writln(r, c);
  for i := 0 to 7 do write(board[i]:4);
  writeln;
 end;
$
ERROR(S) IN IMMEDIATE CODE:                         Shucks.



       VAR
         I: INTEGER;



*
          BEGIN
            WRITLN(R, C);
                 '0'14
            FOR I := 0 TO 7 DO
              WRITE(BOARD[I]: 4);
            WRITELN;
          END;

:mes 0 14                                           What was that?

EXPLANATIONS OF ERROR CODES:
  0: THE DESIGNATED IDENTIFIER HAS NOT BEEN
     DECLARED.
 14: A SEMICOLON IS EXPECTED.

:.c/writln/writeln/                                 Fix the typo.
           WRITELN(R, C);
1 CHANGE MADE                                       Ok. Try again.

:$$

       7           2
  0    4   7   5   2   6   1   3                    I get it!


:edi generate

:noveto                                             Stop verifying.

:f/board[
           BOARD[C] := R;

:o board[r] := c;                                   Put the queen on the row not
*          BOARD[C] := R;                           the row on the queen!




:up 2                                               Show the change in context.

:p4
*      IF COL[C] AND UP[R + C] AND DOWN[R - C] THEN
         BEGIN (* SQUARE FREE *)
           BOARD[R] := C;
           SETSQUARE(R, C, FALSE);

:ign 5
                                                    Let's get five solutions.
```

```
:execute
1 BLOCK RECOMPILED
INTERPRETING QUEENS

 0  4  7  5  2  6  1  3
 0  5  7  2  6  3  1  4
 0  6  3  5  7  1  4  2                        Right on!
 0  6  4  7  1  3  5  2
 1  3  5  7  2  0  6  4
 BREAKPOINT 1 AT:
B>>      END (* PRINTBOARD *);
IN: QUEENS.GENERATE.PRINTBOARD

EXECUTED 57827 STEPS IN 7360 STATEMENTS.


:bye
WARNING - PROGRAM NOT SAVED.                 Thanks for reminding me.
OK?n

:save queens1

:bye
 - END PASCALI
COMMAND-
```

(* Received 79/04/02 *)

TRACING THE HEAP

*Steve Schach
Applied Mathematics Department
Weizmann Institute of Science
Rehovot, Israel

A programmer using a high-level language rightly expects to be shielded from machine implementation details. If there is a bug in a Pascal program, one does not wish to be presented with an assembler listing, or a core dump, but rather with information in a format as close as possible to the original source code. Watt and Findlay [3]have constructed a trace for the stack (i.e., the static Pascal data structures) whichgives the user diagnostic information in the terminology of his program. However, the dynamic data structures created by the procedure new, and stored on the heap, are not traced at all.

The package HEAPTRACE outlined in this paper aids the user to debug his programs by providing information as to the contents of the records on the heap. Each field is named, and its value is given in what might be termed "high-level format". For example, the values of types defined by enumeration (including Boolean) are explicitly printed out as identifiers. The contents of sets are similarly handled. The first and last elements of arrays are given, or the first and last strings of packed arrays of char.

The user may specify which record types are to be traced, and whether variants are to be ignored (if a tag field is not assigned). At any point he may request the entire heap to be dumped, or just the contents of the last n records. He may even specify a variable name, and if that variable is a pointer to a record being traced, then the values of the fields of that record are given.

For portability's sake HEAPTRACE is written in Pascal. It takes the form of a one-pass precompiler which produces as output the original Pascal program suitably modified for tracing the heap according to the user's instructions. The basis of the program is the Pascal-P3 compiler [1] with the code generation routines removed, and an additional 1500 lines of code inserted. Reasons for choosing this form of implementation include

(a) a precompiler needs lexical and syntax analysers, as well as data structures for symbol tables, etc. In order to speed up development time it seemed sensible to start with a thoroughly tested working program which had these features.

(b) At a later stage, it will be relatively simple to implement HEAPTRACE as a compiler by re-inserting the code generation routines and producing the output in the form of P-code rather than Pascal.

(c) A Pascal user may wish to implement this form of trace for the heap as an option to his or her own Pascal compiler. As HEAPTRACE consists of additions and modifications to a well-known and widely circulated compiler, the chances are good that such a person could rapidly understand the principles of HEAPTRACE merely by examining the clearly marked changes to the P3 compiler.

HEAPTRACE works as follows: the command new is modified so that when the user wishes a record to be created on the heap, a second record, a so-called "hyperrecord", is also created. The hyperrecords form a doubly-linked list (the "hyperlist") and each hyperrecord is two-way linked to its associated user-created record. In this way one can ensure that the records to be traced are vertices of a connected graph, even if the user has somehow erred in his handling of pointers. Tracing the heap is then effected by moving along the hyperheap and dumping the contents of the records as selected by the user.

An example of a variant record is given on pages 44-46 of the Pascal User Manual [2]. A program for that example was submitted to HEAPTRACE; the output of the resulting program appears below.

```
***** HEAPTRACE CALLED AT LINE          34

 NODE #       1 TYPE -  PERSON

  NAME             :     RECORD
      FIRST             :     ARRAY
                 STRING : EDGARD
      LAST              :     ARRAY
                 STRING : WOODYARD
  SS               : 845680539
  SEX              : MALE
  BIRTH            :     RECORD
      MO                : AUG
      DAY               : 30
      YEAR              : 1941
  DEPTDS           : 1
    MS                : SINGLE
      INDEPDNT          :  TRUE

 NODE #       2 TYPE -  PERSON

  NAME             :     RECORD
      FIRST             :     ARRAY
                 STRING : NICOLAS
      LAST              :     ARRAY
                 STRING : ROBERTSMAN
  SS               : 627259003
  SEX              : MALE
  BIRTH            :     RECORD
      MO                : MAY
      DAY               : 15
      YEAR              : 1932
  DEPTDS           : 4
    MS                : DIVORCED
      DDATE             :     RECORD
           MO                : FEB
           DAY               : 23
           YEAR              : 1977
      FIRSTD            :  FALSE
```

*On leave from: Department of Computer Science, University of Cape Town, Rondebosch, 7700 South Africa.

This work is supported in part by the South African Council for Scientific and Industrial Research.

format", the underlying structure of each record is reflected in the indentation.

HEAPTRACE is currently in the testing stage. It is hoped to make it available to any interested user as soon as its machine independence has been adequately demonstrated.

REFERENCES

[1] U. Ammann, "The Zurich Implementation", Proc. Symp. on Pascal - the language and its implementation, Southampton, 1977.

[2] K. Jensen and N. Wirth, "Pascal User Manual and Report", Springer-Verlag, Berlin, 1974.

[3] D.A. Watt and W. Findlay, "A Pascal Diagnostics System", Proc. Symp. on Pascal - the language and its implementation, Southampton, 1977.

(* Received 78/11/21 *)

****

WHY USE STRUCTURED FORMATTING?

J. E. Crider
Shell Oil Company
P.O. Box 20329
Houston, Texas 77025

(This paper should be construed as a personal rather than an organizational statement.)

What is Structured Formatting?

"Structured formatting" is a technique for formatting ("prettyprinting") Pascal programs. It is described in a paper in SIGPLAN Notices 13, No. 11 (1978), pp. 15-22. It is designed to display clearly the Pascal statements and their structural relationships.

Structured formatting is based upon a single indented display pattern, which is:

```
introductory phrase
   dependent clause
   dependent clause
   .
   .
   .
   dependent clause
```

This pattern is used to display almost all of the structured statements of a Pascal program. Each dependent clause is typically a statement; if such a statement is itself structured, then it, too, is displayed in the above form. The resulting display clearly shows the nesting that is the

Each dependent clause is typically a statement. If the introductory phrase of a structured statement ends in begin or of, then the last line of the pattern ends with end (possibly followed by a semicolon). For a repeat statement, the last dependent clause is the until clause.

hallmark of structured programs.

Each type of structured statement has its own form of introductory phrase. The complete list of introductory phrases for Pascal statements is:

```
while expression do begin
for control variable := for list do begin
with record variable list do begin
case expression of
repeat
if expression then begin
else if expression then begin
else begin
begin
```

In order for structured statements to begin with these introductory phrases, certain Pascal statements in a program must first be modified. The display preparation modification involves the insertion of redundant begin-end pairs, as follows: every controlled statement in a while, for, with, or if statement is converted into a compound statement, with two optional exceptions. The first exception is that, if the controlled statement is a simple statement such that the complete structured statement can fit on one line, then it need not be converted. An example is:

```
while a[i] <> x do i := i + 1;
```

The other optional exception is that, if the controlled statement in the else clause of an if statement is itself an if statement, then it need not be converted. This exception leads to if statements displayed in a very useful form:

```
if k = n then begin
   count := count + 1;
   r := r + d[k];
   k := k - d[k]  end
else if k > 0 then begin
   r := r + d[k];
   k := k - d[k]  end
else begin
   r := r + 1  end;
```

Thus it is seen that the if statement may appear as a sequence of display patterns: one pattern for the "if" part, one for each "else-if" part, and one for the final "else" part. (Note also that the last two lines in the example above could be replaced by the single line "else r := r + 1;", according to the first exception.)

The one structured statement that is not usually displayed through the display pattern is the compound statement. Instead, it is typically used with another structured statement to indicate the range of control of the latter. Generally, the only compound statements that are displayed through the display pattern are those that represent selection statements in a case statement and those that represent the statement part of a program, procedure, or function. Thus, begin is an introductory phrase only when it cannot be part of another introductory phrase.

From a slightly different point of view, it is seen that the compound statement is always displayed in the same form. This form is:

```
[introductory phrase prefix] begin
    statement;
    statement;
    .
    .
    .
    statement  end
```

Note that begin and end symbols always appear on the ends of lines (followed only by semicolons and comments).

It is worthwhile to force a single exception to this compound statement form. For the compound statement that is the statement part of a program, procedure, or function, the end symbol should appear by itself as the last dependent clause. This last end is treated specially to emphasize the end of the statement part; typically this end is followed on its line by the name of the program, procedure, or function in a comment.

Another important element of the structured format is the indentation increment; it must be the same for every application of the display pattern throughout the program. This facilitates counting the level of nesting, which can be very useful, as seen below.

What about Other Formatting Techniques?

Structured formatting differs from other formatting techniques in several ways. These are:

1. Other techniques generally combine at least two display patterns in various ways. The other display pattern commonly used has all lines indented except the first and the last.

2. Other techniques generally allow for the vertical alignment of matching begin and end symbols. Structured formatting places begin and end symbols at the ends of lines, and provides other ways of confirming valid structures.

3. Structured formatting may require program modification, as described above. Most other techniques can be applied directly to any Pascal program.

4. Other techniques treat the compound statement as a structured statement. In contrast, structured formatting uses begin and end symbols as markers to confirm the range of control of other structured statements; this range of control is expressed primarily through indentation.

What are the Advantages of Structured Formatting?

1. The structured format clearly displays the structure of a Pascal program. The indentation shows the range of control and indicates the dependency of the controlled statements. The overhanging introductory phrase begins with a keyword that indicates the nature of control and also usually includes the controlling condition.

2. The structured format is simple. It uses a single display pattern that has three distinct and well defined parts: an introductory phrase, a sequence of dependent clauses, and the indentation increment.

3. Each line starts with the beginning of a new statement (or else or until clause). Each statement begins on a new line (exceptions: most compound statements, if statements in "else-if" structures, and simple controlled statements). These two properties add to the clarity of the display by emphasizing the statement content, while the indentation pattern emphasizes the control relationships.

4. The structured format is conservative of lines. There are few lines that contain only single symbols; in particular, begin and end symbols rarely appear alone on lines. Thus, the structured format brings the statements of a program structure close, so that their interrelationships may be easily comprehended by the reader.

5. The structured format is conservative of indentation. Each indentation increment corresponds to a change in the level of control of statements; the begin and end symbols of a compound statement are auxiliary to this correspondence, and do not of themselves cause additional indentation increments. These last two advantages mean that space is conserved both horizontally and vertically, an important factor in the publication of programs.

6. If a line contains end or until symbols, then the number of indentation increments that it has, relative to the following line, is equal to the total number of end and until symbols that it contains. This is the indented end relationship; it is extremely useful in desk-checking the structure of Pascal programs. It is a localized relationship, applying to two adjacent lines at a time. (Note that treating the last end symbol of the statement part of a program, procedure, or function as the last dependent clause allows any preceding end symbols to participate in this relationship).

7. The begin and end symbols are always the last symbols of the lines on which they appear (excluding semicolons). Although matching pairs of these symbols are not vertically aligned, arcs connecting them can be drawn easily, if needed.

8. The display preparation modification leads to the very small set of introductory phrases, and also to the valuable indented end relationship. Further, it inhibits the use of some of the more confusing structured statement sequences, such as "if . . . then if . . . then . . . else . . .".

9. The "else-if" exception to the display preparation modification provides for a valuable and commonly used control structure, and avoids the "stair-step" pattern that would otherwise appear.

10. With the display preparation modification, the fundamental algorithm for managing indentation and display is quite simple: for each begin, of or repeat symbol, increment indentation and follow with a new line; put out a new line after each semicolon and before each else or until symbol, and also before the last end symbol of the statement part of a program, procedure, or function; and for each end or until symbol, decrement indentation for the lines following.

11. The structured format allows every line to end with a semicolon; the sole exception is the line preceding a line that begins with the else symbol. Further, semicolons need appear nowhere else but at the end of a line.

12. Structured formatting can be applied to complete Pascal programs, as well as to Pascal statements. At the top level, the display pattern gives:

```
program heading
   label declaration part
   constant declaration part
   type declaration part
   variable declaration part
   procedure or function declaration
   procedure or function declaration
   .
   .
   .
   procedure or function declaration
   statement part .
```

The display pattern is then applied to each of the declaration parts. Thus, the introductory phrases for Pascal include the program heading, the procedure heading, the function heading, and the keywords label, const, type, and var, as well as the introductory phrases for statements (note that the introductory phrase for the statement part is begin).

13. Structured formatting can be applied to each procedure or function declaration as well, for each one has a structure quite similar to that of a program. Because procedure and function declarations can be nested, the number of indentation increments at a procedure heading or a function heading is equal to the static level of that procedure or function.

14. Structured formatting can be used to advantage with structured programs in many other languages as well. In other languages, however, the indented end relationship may not obtain.

What about an Example?

This example is Program 3.7 from Niklaus Wirth's book, Algorithms + Data Structures = Programs (Prentice-Hall, 1976). The comments have been changed and semicolons have been inserted before the last end symbols. Further, the display preparation modification has been made to the first for statement in the program (the controlled statement was not simple or compound) and to the for statement within the repeat statement (the controlled statement was too long).

```
program selection (input, output);
(* find optimal selection of objects under constraint *)
   const
      n = 10;
   type
      index = 1..n;
      object = record
         v, w: integer  end;
   var
      i: index;
      a: array [index] of object;
      limw, totv, maxv: integer;
      w1, w2, w3: integer;
      s, opts: set of index;
      z: array [boolean] of char;

   procedure try (i: index; tw, av: integer);
      var
         avl: integer;
      begin                       (* try *)
         if tw + a[i].w <= limw then begin
            s := s + [i];         (* try inclusion of
                                     object i *)
            if i < n then try (i + 1, tw + a[i].w, av)
            else if av > maxv then begin
               maxv := av;
               opts := s  end;
            s := s - [i]  end;
         avl := av - a[i].v;      (* try exclusion of
                                     object i *)
         if avl > maxv then begin
            if i < n then try (i + 1, tw, avl)
            else begin
               maxv := avl;
               opts := s  end  end;
         end;                     (* try *)
```

```
begin                                (* selection *)
   totv := 0;
   for i := 1 to n do begin
      with a[i] do begin
         read (w, v);
         totv := totv + v  end  end;
   read (w1, w2, w3);
   z[true] := '*';
   z[false] := ' ';
   write (' weight   ');
   for i := 1 to n do write (a[i].w: 4);
   writeln;
   write (' value    ');
   for i := 1 to n do write (a[i].v: 4);
   writeln;
   repeat
      limw := w1;
      maxv := 0;
      s := [];
      opts := [];
      try (1, 0, totv);
      write (limw);
      for i := 1 to n do begin
         write ('   ', z[i in opts])  end;
      writeln;
      w1 := w1 + w2
      until w1 > w3;
   end                               (* selection *)
```

(* Received 79/03/22 *)

Future of Pascal News - Save the PUG

The University of Southampton

Computer Studies

Professor D W Barron

30th January 1979.

Dear Andy,

Here are some thoughts on the future of PUG, prompted by your Open Letter in PN13. Perhaps I should start by stating my own position, which is this. PUG has succeeded beyond all reasonable expectation because it has been informal and unconventional. To institutionalise it is to administer the kiss of death. I have been happy to support PUG in its present form with my volunteer effort, but I want no part in an institutionalised PUG. The day the proposed constitution is adopted, someone else can take over the European printing and membership services.

Reading various contributions to PN13, it is clear that there are two very different views of PUG. There are those who want PUG to be "pre-eminent with regard to Pascal", and to have some sort of authority over the language. Obviously, institutionalising PUG is attractive to this group. But there already exist organisations to deal with standards - ISO, ANSI and BSI. It is folly to believe that a self-appointed, institutionalised PUG can keep Pascal to itself. And has anyone thought about the logistics of obtaining a consensus from 3000 members in 41 countries and 49 states?

The alternative school of thought, to which I adhere, recognises that the enormous success of Pascal has been achieved not through the existence of PUG per se, but from the publication of Pascal Newsletter and Pascal News. It is the dissemination of the "vast quantities of information" that has done the trick. The value of Pascal News is incalculable, but institutionalising PUG won't make any difference to it, except by probably putting the price up and adding layers of unnecessary formality and bureaucracy to the production process.

Pascal News is the most valuable thing we do - not so much the articles, which could perfectly well go into SIGPLAN Notices (or Software Practice and Experience), but the Implementation Notes and the miscellaneous information. We don't need a Constitution to keep on producing Pascal News, just an Editor and a sympathetic print-shop. If we can't maintain our informal but effective publication without a lot of (*expletive deleted*) formality, let's shut down the enterprise. We've nothing to

continued.........

Department of Mathematics, The University, Southampton, SO9 5NH. Tel: 0703 559122 Ext: 700 Telex: 47661

Open Forum for Members

be ashamed of: we've done what many people thought was impossible. Your description of such an act was a quotation - "for one brief shining moment there was Camelot". Let me close with another quotation (from that excellent European, James Joyce); ".. better pass boldly into that other world, in the full glory of some passion, than fade and wither dismally with age..."

Yours sincerely,

David.

D.W. Barron.

P.S. You should worry about passing 30. I just passed 44, but a few people still trust me.

****

March 12, 1979

Mr. Andy Mickel
Pascal User's Group
University Computer Center: 227 EX
208 S.E. Union Street
University of Minnesota
Minneapolis, MN 55455

Dear Andy:

I have sent my ballot on to Rick Shaw, but I wanted to say that I can understand your position. With each issue of Pascal News I have been amazed that you could have produced such a product. I know the time it takes to bring it all together. In a real way Pascal News is PUG. I would urge you to pass the editor's job on to someone else very carefully. And while I agree you should try to keep the cost of PUG membership down, you are perhaps being unrealistic about the help needed to produce a quarterly publication for 3,000 members.

Sincerely,

Paul Brainerd
1630 S. 6th Street, D-1605
Minneapolis, MN 55454

Open Forum for Members

2918 Kevin Lane
Houston, Texas 77043
March 19, 1979

Andy Mickel
University Computer Center: 227 EX
208 SE Union Street
University of Minnesota
Minneapolis, Minnesota 55455

Dear Andy:

I am writing this letter for several reasons. First, I have now received my copies of Pascal News #13 and #14. I wrote you earlier, wondering what had happened to the Pascal News, because I had read the minutes of the first ANSI X3J9 meeting in which #14 was mentioned, at a time when I had not even received #13!

Second, enclosed is a paper that I am herewith submitting to you for publication as an article in the Pascal News. Its purpose is to promote "structured formatting", a technique that I have found very useful in visualizing statement structures. The technique also has features that are important for the publication of programs (it saves space, at one line per statement yet!). While it takes some getting used to, I hope that you and other Pascalers will give it a try.

Third, enclosed is a copy of a letter that I am writing to Tony Addyman regarding his standardization efforts. The letter describes two additional changes to Pascal that I have found in the working draft published in Pascal News #14. I hope that both changes can be removed.

Fourth, I have a correction to Wirth's EBNF of Pascal in Pascal News #12: additional Predeclared identifiers are FALSE and TRUE.

Fifth, as a PUG member and a Pascal user, I want to tell you that I appreciate very much the incredible effort that you have put into PUG and the Pascal News. The Pascal News has become an impressive journal that is my major link with Pascal developments, and I am sure that it serves most other PUG members the same way. If Pascal helps the computing community to move on to better languages that supplant primitive languages like FORTRAN, it is largely through your work in promoting Pascal in these last few years.

Sincerely yours,

John Earl Crider

19 March 1978

Dear Andy,

Here is that quote that I read to you on the phone; I've translated it from the book "10 años con mafalda", drawn by Quino:

"This air of happiness, of tranquility that you have now, Quino; is it due to the fact that you've killed off Mafalda?
--I stopped doing her a few months ago, and yes, I am more comfortable. More free. It's been ten years of cartooning, and I was beginning to repeat myself. It seemed to me more honest, more healthy to stop doing her.
--Have you ever regretted at any moment creating her?
--No, not that. I did her with much enthusiasm. What happened is that she came to be an oppressive personality, an obligation, and then it wasn't fun any longer; I was fed up with it.
--Nonetheless, you owe your popularity to Mafalda.
--Yes, (he admits), and that used to irritate me.
--I must confess that it's hard to imagine you irritated.
--Well, I had spent the previous twelve years doing humorous cartooning when Mafalda came out; it's not that I was a complete unknown (not like they stop me on the streets now either), but only when the comic started did I get the "boom". And actually, one could say that the whole world, more or less, knows who Mafalda is.
A little bit earlier, on the street, we saw a Mafalda made of coloured wood displayed on the balcony of a store selling infants' goods, and Quino stopped for a moment and said, "Hey, look at her!"
--Does the inveitable commercialisation of your characters bother you?
--It disgusts me more than it bothers me. As you said, it's inevitable. The time comes when, if one doesn't have a license to make shirts or whatnot, someone will do it, and you'll have to prosecute and all that...thus, there's no sense in denying it. What irks me is the need that some people have to buy a shirt or blouse with the character. It's a bit sad, because you notice that it's a matter of pure consumerism; that this year Mafalda can be in style and sell a mountain of blouses with her effigy, while the next year the style could change...
--Has Mafalda made you rich?
Quino smiles broadly, and, with an almost energetic negative:
--No, no. Rich, for me, no. Perhaps, for the editors. For them surely. It's like every process: he who gains the least is he who creates."

I have enclosed a couple of cartoons from the book; you don't have to know Spanish to enjoy them. The man really is a genius. In case you're wondering, he's currently back doing editorial cartooning and, from a recent cartoon I saw, he has not lost his touch.

As for the other topic we discussed (the constitution), I proudly give you the following (with apologies to Eugene Ionesco, whose play The Bald Soprano I highly recommend; if for nothing other than the fable about the fox and the snake).

The Bald Organization
(An Anti-Constitution)

ARTICLES I,II, and III
A, an, and the (respectively)

ARTICLE IV - Name of the organization

The name of this organization shall be "The Organization With No Name". This will enable us to, en masse, star in Spaghetti Westerns and acquire great masses of money.

ARTICLE V - Purposes of the Organization

To promote Pascal by keeping it in as tight a strait-jacket as possible.

To promote Pascal by adding extensions to it willy-nilly.

(Choose one of the above depending on which side of the fence you're on.)

To fight for Truth, Justice, and the American Way (you'll believe a program can fly!)

ARTICLE VI - Membership

You pays your money, you takes your choice. Voting rights: one person, one vote. (In deference to historical tradition, Chicago members need not be alive at the time their votes are cast.)

ARTICLE VII - Officers

The Organization With No Name will have the following officers:
-The Chair
-The Vice-Chair (a.k.a. the Social Director - in charge of vice)
-The Secretary/Treasurer
-The Editor of the "No News is Good News" no-name newsletter
-The Sergeant-at-Arms

Officers have terms as follows, and are elected by the means stated below:

The Chair: elected by voice vote or Applause-O-Meter, in office until another election is held, or Chair is deposed or impeached. (Impeachable offense: actually doing something). The Chair's major duty is to be a figurehead.

The Vice-Chair: elected by reputation. This person, being social director, must have impeccable taste in pizza and beer. Holds office until tired of throwing parties, deposed, or impeached. (Impeachable offense: ordering anchovies on the pizza)

The Secretary/Treasurer: must be able to type at least 50 words a minute, and be able to add and subtract simple quantities without the aid of a hand calculator. Must have great legs and a decent figure (yes, this DOES go for male candidates as well; we don't want to be sexist and surely there are women out there who can judge men's figures). Holds office until tired, elected out, deposed, or impeached. (Impeachable offense: absconding with the funds -- and getting caught at it.)

The Editor of the "No News is Good News" no-name newsletter: also must be able to type at least 50 words a minute, but nobody cares how good he/she/it looks. Must have a nodding acquaintance with the grammar of the English language; helpful if candidate does not cringe in terror when confronted by the wrong use of "its" vs. "it's" in a document. Holds office until elected out, deposed, impeached, or taken off to the Laughing Academy. (Impeachable offense: printing an issue without at least one article that can start a stream of nasty debates.)

The Sergeant-at-Arms: elected in trial by combat among candidates. Must be able to bench press 100 kilograms; at least a brown belt in judo or karate is helpful. Major duties include keeping decormm at meetings (see below). Holds office until thrashed severly by up-and-coming candidates, deposed, or impeached. (Impeachable offense: are you kidding? YOU want to tell the Sergeant-at-Arms that he/she/it is out?)

ARTICLE VIII - Meetings

Meetings are called by the Vice-Chair (social director) and are held, if possible, in low-class dives late at night or early in **the** morning. The Annual meeting is an exception, being held during the annual ACM conference; these usually take place in high-class dives. Elections are held during the Annual meeting; the secretary/treasurer should be prepared to pay for damages to the premises (see Sergeant-at-Arms, above). All copies of Robert's Rules of Order will be confiscated at the door for use when the meeting place runs out of toilet paper.

ARTICLE IX - Dress Code

Of course it's ridiculous to have a dress code, but with all the other mickey-mouse crap you usually find in a constitution don't you think one belongs here? Men: Black tie and sneakers (Adidas and Puma preferred, but deck shoes are permitted). Women: Plumed hat and high heels. Other clothing is optional (for both sexes).

ARTICLE X - Amendments

If you want to change the costitution, go ahead, but that puts you first in line for the Chair position.

Bylaws

ARTICLE I - Buy low, sell high.

-0-

No hard news in this letter; I'll send another in a few days with some of the stuff I heard at San Diego (if I find the time to write it before heading off to the gymnastics tournament this weekend.) By the way, congratulations to the University of Minnesota gymnastics team, who won Big 10 a couple of weeks ago here in Michigan. (An addition error in scoring almost gave the title to Ohio State, but it was found and corrected. Ohio State was mightily unamused.)

I leave you with the following poem by the wondrous Dorothy Parker:

Observation

If I don't drive around the park,
I'm pretty sure to make my mark.
If I'm in bed each night by ten,
I may get back my looks again.
If I abstain from fun and such,
I'll probably amount to much;
But I shall stay the way I am,
Because I do not give a damn.

John

JDEisenberg
1510 Plymouth Rd. #59
A^2, MI 48105

UNIVERSITY OF MINNESOTA
TWIN CITIES

Social Science Research
Facilities Center
25 Blegen Hall
269 19th Avenue South
Minneapolis, Minnesota 55455
612-373-5599

79/05/01

To: "Friends of PUG"
Tony Addyman
David Barron
Judy Bishop
Rich Cichelli
Scott Jameson
Bob Johnson
Andy Mickel
Bill Price
Arthur Sale
Rick Shaw
Barry Smith
Rich Stevens

From: Jim Miner

Enclosed is a draft contribution to Pascal News #15.

Because of the fundamental importance of the issue to the future of PUG, I am requesting that you return comments (of any kind) to me as soon as possible.

The following address is simplest:

Jim Miner
SSRFC: 25 Blegen Hall
University of Minnesota
Minneapolis, MN 55455
U.S.A.

Thanks in advance!

Save the PUG!

Abstract

There may still be a chance to save the PUG from extinction.

What Is PUG?

To anyone who cares to look, it is obvious that PUG is a mailing list used to distribute Pascal News to individuals around the world. PUG was really started by George Richmond at the University of Colorado when he decided to publish the Pascal Newsletter. Later, Andy Mickel at the University of Minnesota extended George's efforts and added the name PUG.

Pascal News is a "bulletin board" where nearly anyone can post or read messages. It is accessible to large numbers of people. It is inexpensive. It is simple. And many members of the Pascal community have told me that it is very important that Pascal News not die.

PUG is the fastest-growing, and possibly the largest group of its kind in the world. Its membership (i.e., Pascal News subscribers) includes a very broad base of experience and interests.

It is important that PUG has never taken an "official" stand on any important issue. But PUG has provided the means for coordinating the actions of individuals who have had lasting effects on the language and its implementations. For example, Tony Addyman is undoubtedly the major force behind the current international standardization effort for Pascal. But PUG itself has never done any work on the standard. Tony, along with other individuals, has taken the burden, and has reported on progress to the rest of the community in Pascal News.

Many individual members of PUG played an important role in the UCSD Workshop last summer. Rich Cichelli endangered his own pride and reputation to act as a conscience for the entire group. In spite of the unkind things that have been said about his viewpoints, his individual actions strongly influenced the results of the Workshop. Ken Bowles insisted that there should be an "official" PUG stand, but those of us attending knew all too well that we could not represent a group of 2000 people other than by reporting the results in Pascal News. We could, and did, act as individuals.

All of this leads me to the most basic observation. PUG is NOT a policy-making body. For it to adopt "official" positions on anything requires either a consensus from its 3000+ members, or else a formal means for deciding that one viewpoint is "better" than another one. Any such formal decision mechanism is inherently political, and as such is subject to power struggles, costly overhead, and bureaucracy. In my view, there is no better way to destroy what we have.

The Proposed Constitution

Before going any farther I want to say that I respect Rich Cichelli as a person and as a member of the Pascal community. But I do not agree with his view of what PUG "should be".

The Constitution and Bylaws proposed in Pascal News #13 would effectively allow PUG to try to legislate policy, in addition to its current status as a publisher. I think there would be several very specific harmful effects of this change.

First, we can expect that the cost of Pascal News would probably increase substantially. The overhead involved in holding meetings, supporting the necessary bureaucracy, etc., must be paid somehow. As individual members, we can expect to do the paying. And we can expect that some subscribers will not continue at the higher rates. Also the true cost of participating would be prohibitively high for most members, especially those outside the United States. This is a simple case of economic discrimination. PUG policy would be determined by those who could afford to attend the yearly business meetings.

Second, a political PUG may lose many of its members for non-economic reasons. David Barron has already stated that he will not continue to support European distribution under such a regime. Andy Mickel has told me personally that he would not even be a member. Another individual, a highly respected software engineer in the industry, has told me that he might not have the time necessary to participate in a political PUG, and further that his participation might constitute a conflict of interest with his job. Another person from industry offered his company's support for PUG, but only if it remains "informal" (read "apolitical"). I personally have no desire to spend the time and money to attend yearly meetings where I can expect the inevitable power plays designed to capitalize on the influence of PUG in the industry and consumer market.

Third, the creation of PUG policy will very likely cause factions of the community to break off in order to form their own biased organizations and publications to counter what they perceive as the biases in PUG. Certainly if PUG tries to claim that it "represents" its members with a position on an issue, either some members will be left out or else only those who agree with the position will stay in PUG. Either way, somebody loses.

One other thought occurs: if the proposed constitution did not actually destroy PUG, it might have the opposite effect -- to make PUG outlive its usefulness, and to promote Pascal long after better languages have overtaken it. How ironic this would be, and how sad!

Where Now, PUG?

Well, the votes are in, and as detailed elsewhere, the results are fairly certain:

For	2 %
Against	1 %
Abstain	97 %

The meaning of this is not obvious, but we can make some guesses. As one person said to Andy Mickel, "I didn't vote because I didn't think you were serious." He probably spoke for a large number of members.

But rather than try to second-guess 2900+ people, let's consider constructive alternatives to the Constitution. What is it that we really need?

First, as Bill Price explained to me, any publication has two functional components: a publisher, and an editor (and staff). Currently Andy Mickel (with help from friends and the University of Minnesota) is providing both services. With the growth of PUG and the explosion of Pascal it is no longer feasible for these volunteers to do both tasks.

What we need to create (or find) is a publisher whose only purpose is to provide the support functions necessary to providing Pascal News. It should assure editorial autonomy and the availability of Pascal News as an open forum for members of the Pascal community. It must obtain funds from memberships, subscriptions, grants, etc.

Based on discussions with a number of other PUG members, I think our best chance lies in creating a non-profit institution whose one and only goal is the publication of an autonomous and open Pascal News.

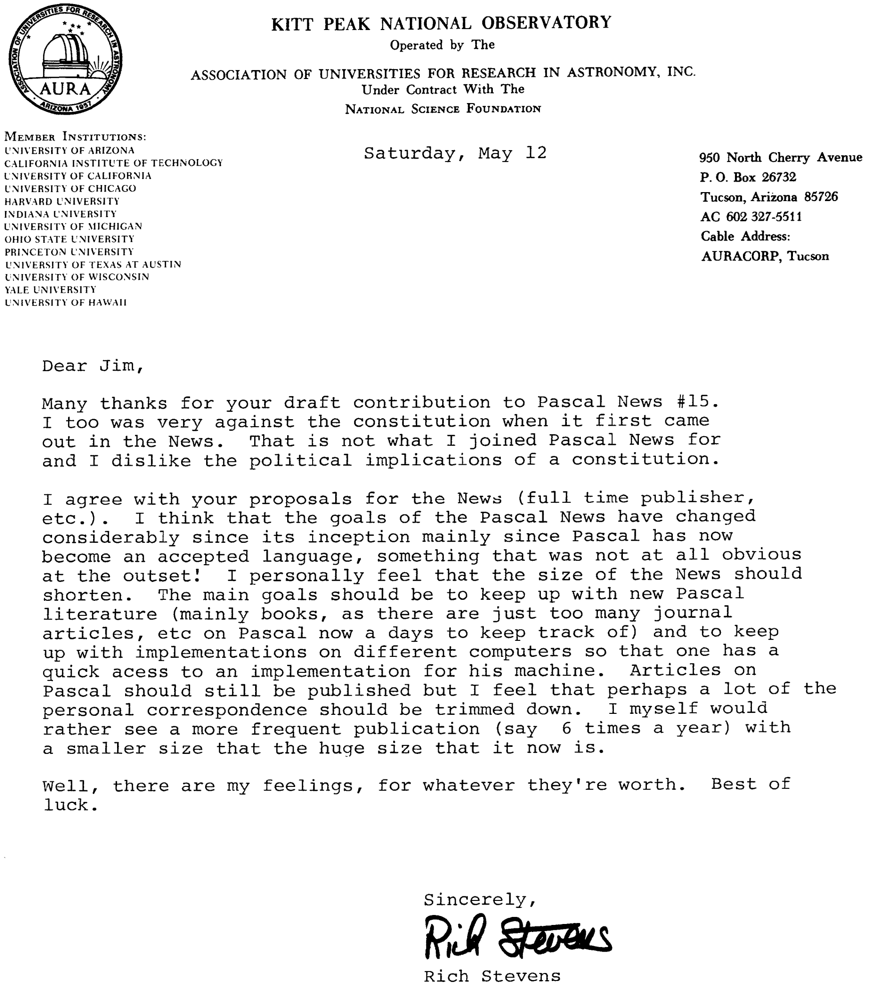
We also need an editor.

The success of this scheme will depend on support from individuals and (at least in the short term) from corporations. It is notable that a number of companies have already offered monetary or other support.

Save the PUG

Pascal is growing like never before. This growth will continue. Pascal News is needed to unite the Pascal community, to aid its communication, and to prevent a vacuum which special interests will inevitably fill.

Arthur Sale remarked in these pages in 1977 that "Pascal has much more to fear from its friends than its enemies." These words might just as well have been spoken about PUG.



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

Saturday, May 12

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

Dear Jim,

Many thanks for your draft contribution to Pascal News #15. I too was very against the constitution when it first came out in the News. That is not what I joined Pascal News for and I dislike the political implications of a constitution.

I agree with your proposals for the News (full time publisher, etc.). I think that the goals of the Pascal News have changed considerably since its inception mainly since Pascal has now become an accepted language, something that was not at all obvious at the outset! I personally feel that the size of the News should shorten. The main goals should be to keep up with new Pascal literature (mainly books, as there are just too many journal articles, etc on Pascal now a days to keep track of) and to keep up with implementations on different computers so that one has a quick acess to an implementation for his machine. Articles on Pascal should still be published but I feel that perhaps a lot of the personal correspondence should be trimmed down. I myself would rather see a more frequent publication (say 6 times a year) with a smaller size that the huge size that it now is.

Well, there are my feelings, for whatever they're worth. Best of luck.

Sincerely,

Rich Stevens

Rich Stevens

The University of Tasmania

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

18th May, 1979

Dear Jim,

This letter is in reply to yours of 1st May to "Friends of PUG".

I agree with your sentiments, expressed in your draft. I have only two points to make:

(a) Policization of PUG on a US-basis as proposed would effectively eliminate international co-operation by ignoring it. I think the non-US PUG members deserve a few moments thought.

(b) A non-profit corporation seems a good idea, so long as it is possible to wind it up when we want to. I completely agree with the bad effects of PUG surviving beyond its legitimate life-span, and I said so to Andy while he was here.

More power to your pen; go ahead.

Yours sincerely,

Arthur

Arthur Sale,
Information Science Department.

****

A Note on the future of PUG

I wholeheartedly support Jim Miner's proposal to create a non-profit institution to publish Pascal News. When Andy changed the name from "PUG Newsletter" to "Pascal News" he recognised implicitly that the only real function of PUG is to publish "Pascal News". If such a body is to be set up I shall be happy to help in any way I can.

(Incidentally, I had already had a similar idea as a contingency against the vote going in favour of a "Political PUG". My scheme was to pre-empt the issue by separating Pascal News from PUG, creating a new company to publish the former, leaving the latter to indulge in pointless politics).

David Barron

varian

May 11, 1979

Mr. Andy Mickel
Pascal User's Group
University Computer Center: 227 EX
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455

Dear Andy:

Attached is an all-purpose coupon with my new mailing address and phone number.

It was nice talking to you last week. I called Rick Shaw and volunteered my services. He said he would call as soon as he has finished his move. Between Rick's and a couple of local PUG members' comments, I think the vote results were a combination of confusion and simply not noticing the ballot. In any event, I am left with the impression that PUG will continue as currently organized with Rick et al. taking over most of your tasks. In light of the current situation I believe a distributed work approach will provide a workable, though not optimal, solution to PUG's immediate needs.

I still feel Pascal News provides a useful source of information and will vehemently oppose any movements which advocate dissolution, or radical change from the current editorial policies. I hope my conviction to PUG is substantiated by my volunteering to help with the production of Pascal News.

The group PASCAL (see attached) is a local interest group and wants to stay strictly local. The article in Intelligent Machines Journal is a bit misleading. ***

I look forward to working with Rick and you in the near future.

Sincerely,

Gregg E Marshall

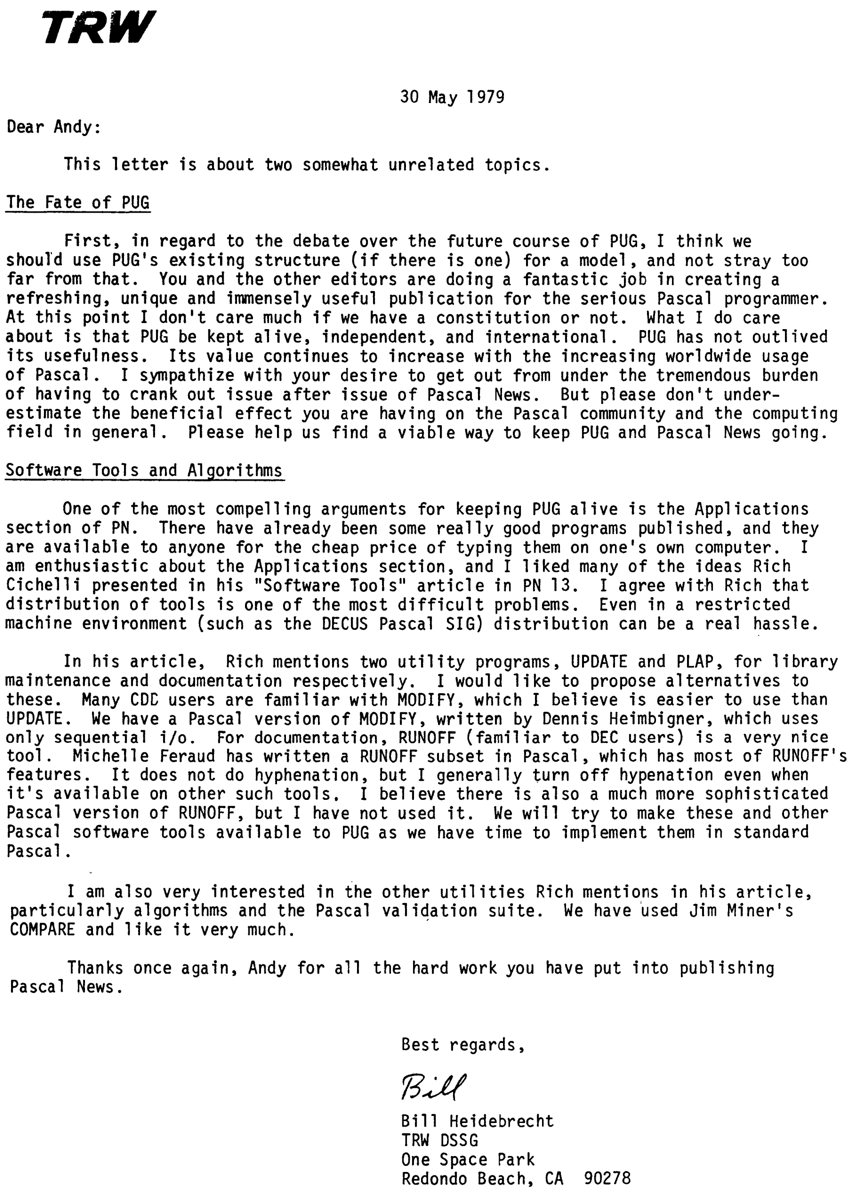
Gregg E. Marshall
Scientific Programmer
Software Development

GEM:bb

cc: Rick Shaw
Enclosures

*** (* See Pascal in the News in the Here and There section. The Pascal Advancement Society of CALifornia (PASCAL) was also publicized in the May, 1978 Byte. - Andy *)

instrument division / aerograph operations
2700 mitchell dr. / walnut creek / calif. 94598 / 415 939-2400



TRW

30 May 1979

Dear Andy:

This letter is about two somewhat unrelated topics.

The Fate of PUG

First, in regard to the debate over the future course of PUG, I think we should use PUG's existing structure (if there is one) for a model, and not stray too far from that. You and the other editors are doing a fantastic job in creating a refreshing, unique and immensely useful publication for the serious Pascal programmer. At this point I don't care much if we have a constitution or not. What I do care about is that PUG be kept alive, independent, and international. PUG has not outlived its usefulness. Its value continues to increase with the increasing worldwide usage of Pascal. I sympathize with your desire to get out from under the tremendous burden of having to crank out issue after issue of Pascal News. But please don't underestimate the beneficial effect you are having on the Pascal community and the computing field in general. Please help us find a viable way to keep PUG and Pascal News going.

Software Tools and Algorithms

One of the most compelling arguments for keeping PUG alive is the Applications section of PN. There have already been some really good programs published, and they are available to anyone for the cheap price of typing them on one's own computer. I am enthusiastic about the Applications section, and I liked many of the ideas Rich Cichelli presented in his "Software Tools" article in PN 13. I agree with Rich that distribution of tools is one of the most difficult problems. Even in a restricted machine environment (such as the DECUS Pascal SIG) distribution can be a real hassle.

In his article, Rich mentions two utility programs, UPDATE and PLAP, for library maintenance and documentation respectively. I would like to propose alternatives to these. Many CDC users are familiar with MODIFY, which I believe is easier to use than UPDATE. We have a Pascal version of MODIFY, written by Dennis Heimbigner, which uses only sequential i/o. For documentation, RUNOFF (familiar to DEC users) is a very nice tool. Michelle Feraud has written a RUNOFF subset in Pascal, which has most of RUNOFF's features. It does not do hyphenation, but I generally turn off hypenation even when it's available on other such tools. I believe there is also a much more sophisticated Pascal version of RUNOFF, but I have not used it. We will try to make these and other Pascal software tools available to PUG as we have time to implement them in standard Pascal.

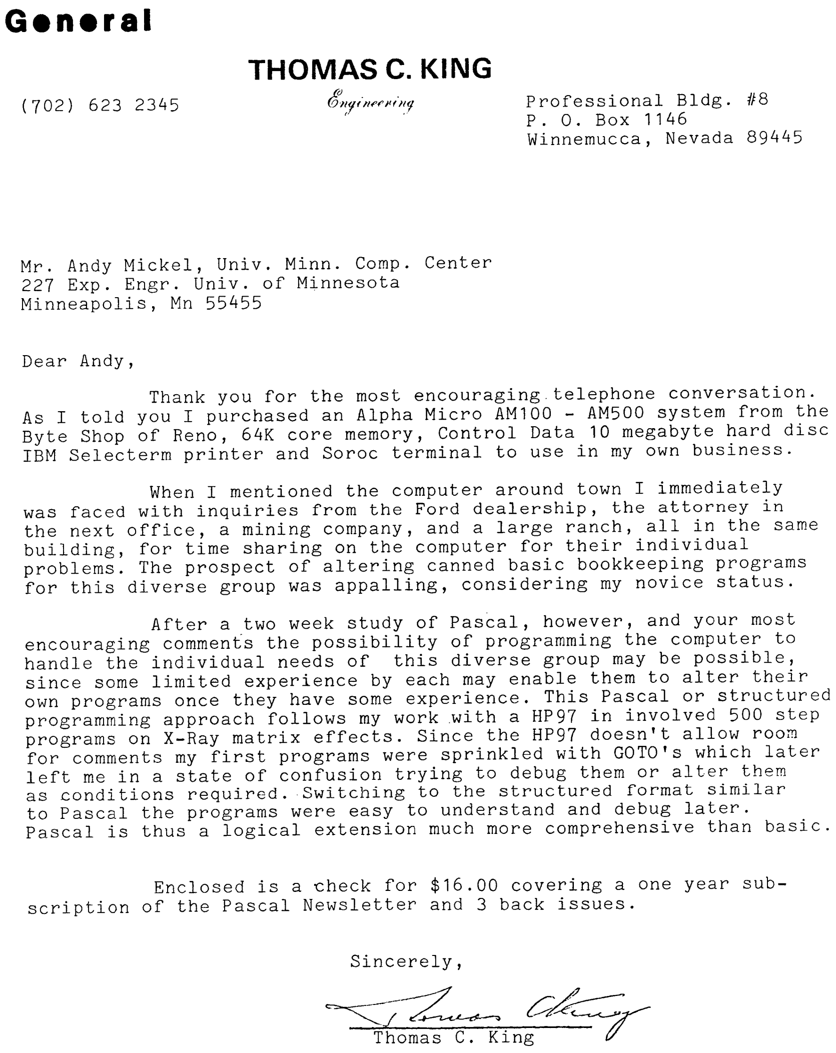
I am also very interested in the other utilities Rich mentions in his article, particularly algorithms and the Pascal validation suite. We have used Jim Miner's COMPARE and like it very much.

Thanks once again, Andy for all the hard work you have put into publishing Pascal News.

Best regards,

Bill

Bill Heidebrecht
TRW DSSG
One Space Park
Redondo Beach, CA 90278



General

THOMAS C. KING

Engineering

(702) 623 2345

Professional Bldg. #8
P. O. Box 1146
Winnemucca, Nevada 89445

Mr. Andy Mickel, Univ. Minn. Comp. Center
227 Exp. Engr. Univ. of Minnesota
Minneapolis, Mn 55455

Dear Andy,

Thank you for the most encouraging telephone conversation. As I told you I purchased an Alpha Micro AM100 - AM500 system from the Byte Shop of Reno, 64K core memory, Control Data 10 megabyte hard disc IBM Selecterm printer and Soroc terminal to use in my own business.

When I mentioned the computer around town I immediately was faced with inquiries from the Ford dealership, the attorney in the next office, a mining company, and a large ranch, all in the same building, for time sharing on the computer for their individual problems. The prospect of altering canned basic bookkeeping programs for this diverse group was appalling, considering my novice status.

After a two week study of Pascal, however, and your most encouraging comments the possibility of programming the computer to handle the individual needs of this diverse group may be possible, since some limited experience by each may enable them to alter their own programs once they have some experience. This Pascal or structured programming approach follows my work with a HP97 in involved 500 step programs on X-Ray matrix effects. Since the HP97 doesn't allow room for comments my first programs were sprinkled with GOTO's which later left me in a state of confusion trying to debug them or alter them as conditions required. Switching to the structured format similar to Pascal the programs were easy to understand and debug later. Pascal is thus a logical extension much more comprehensive than basic.

Enclosed is a check for $16.00 covering a one year subscription of the Pascal Newsletter and 3 back issues.

Sincerely,

Thomas C. King

Thomas C. King

1510 Plymouth Rd. #59
Ann Arbor, MI 48105
2 November 1978

Dear Andy,

Thanks very much; I now have all the back issues. (I accidentally got two copies of #11 and #12, and am sending one of each back to you.)

As anyone who has been a member of PUG for over a year knows, a lot of verbiage about extending Pascal in one form or another has appeared in the PUGN pages. New members, though, may be wondering "What is all this bickering about?". Well, I've been doing some thinking about this, and would like to present a (perhaps overly simplistic) view of all this confusion. (If the reasons are really obvious to everyone, then I guess I'm just slow catching on.)

There appears to be one group of people who wish to repair the minor inconsistencies in the definition of Pascal (User Manual and Report; Axiomatic Definition). The best example of this group's views is in the article by Welsh, Sneeringer, and Hoare [1] . I don't think anyone really has any argument about the things they point out; if they are fixed or not, the essential "character" of Pascal remains the same.

The three major groups (as I see it) who are arguing about Pascal extensions are:

> Group A: Educators using Pascal to teach computer science students about programming and computing
>
> Group B: "Working stiffs" (usually non-educational environment) who wish to use Pascal in their day-to-day endeavours.
>
> Group C: Educators using Pascal to teach people in a non-computer science discipline about programming and computing as a tool for that discipline.

Arguments about extensions usually go like this:

B: I think Pascal should have feature X. I can demonstrate its immense utility for the work I am doing in discipline Q.

A: Feature X is not needed. It is merely a combination of Y,Z, and W, which are already part of Pascal. Computer science students need to know about Y,Z, and W anyway; therefore they should use them instead of X.

C: I am teaching my students to use Pascal for solving problems in discipline Q. I would prefer to have X available so that my students need not worry about Y, Z, and W -- after all, I'm teaching Q, not computer science. But Pascal still has to be easy enough so my students can appreciate the value of computing (and Pascal) in relation to Q.

And the damn shame is that they are all making absolutely correct statements. The computer scientist SHOULD learn how to combine elementary features of Pascal to make complex functions. The educator (outside computer science) doesn't want his students to worry about those details; that's not their province. The "applications" (non-educators) either have been through Computer Science and know about the elementary features, or have had the "canned" features available -- in any case, their goal is not to learn about computing but to get some task done.

All of this seems to come down to the question of the design goals of Pascal. Vavra [2] also realizes, and points out the existence of these different groups and their differing goals. I agree wholeheartedly that some heavy thinking has to occur in this area. At any rate, for those of you who might have been confused about all this argument about "Whither Pascal?", you now have another viewpoint to (hopefully) make things clearer. End of Sermon.

Just a random thought -- and this idea is one I've heard before; certainly not original with me. Credit to whomever came up with it. Those who wish to implement some new control structure in Pascal which is a combination of existing elementary functions should provide a standard Pascal program that translate programs using the extension into the standard version. For features which can be implemented equally well as calls to user-defined procedures, some body of people should start collecting those procedures so that everyone can use the same ones and portability won't go down the tubes. (This includes things like the IMSL library, data base manipulation, formatted I/O, et al.) I am sure this has all been said before; someone out theee please jog my memory and tell me where I've seen it. Take this entire paragraph for what it's worth, and call me in the morning.

It's getting late again, and I'm beginning to flake out. I'd best quit while I'm ahead.

John Eisenberg

REFERENCES (they always make ideas seem so official...)

1. Welsh, Sneeringer, and Hoare, "Ambiguities and Insecurities in Pascal", Software--Practice and Experience, Vol. 7 (1977), 685-696
2. Vavra, R, "What are Pascal's Design Goals", Pascal News, No. 12 (June 1978), pp. 34-35

ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLEAIRE

EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

SIÈGE: GENÈVE, SUISSE

Adresse postale/Postal address

R. Cailliau PS Division
CERN
CH 1211 GENÈVE 23
SUISSE/SWITZERLAND

TÉLEX: 23698 CH
TÉLÉGRAMMES: CERNLAB-GENÈVE

TÉLÉPHONE: GENÈVE (022)
Direct: 83 5041 /83 2535 /83
Central/Exchange: 83 61 11

PASCAL News
c/o Andy Mickel
University Computing Center/227EX
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455
U.S.A.

Votre/Your ref. Notre/Our ref.
PS/CCI/RC/ww

Geneva, 16th October, 1978

Dear Andy,

Here are a few comments on things I read in the latest Pascal News:

1. Mr. Terje Noodt's letter on the user interface and environment interface of Pascal is indeed to the point. The manipulation of sequential files is elegantly supported by the procedures READ, WRITE, RESET, REWRITE, GET, PUT and the functions EOF and EOLN. There is, however, no way of setting up a relationship between a file variable FV and an externally existing file EEF. The only way of indicating that such a relationship is supposed to exist is to put the name FV in the list of program parameters. This means a) a Pascal program is not a stand-alone unit but nothing more than a "procedure", called by the external world (see P4-implementation for example), b) the externally existing files are passed as VAR-parameters to the program (although the reserved word VAR is not used in the program header), and the program is not able to change the relationships.

This approach may work well for the classical student program that is submitted in a batch environment, reads from one file (INPUT!) and writes output to one other file (OUTPUT!) both of which exist only as long as the job lasts. Problems arise immediately when one wants to write a useful, interactive program. These programs have the following characteristics:

- they obtain information from the user, and must try to recover from his typing errors,
- the relationships between internal file variables and externally existing files cannot be set up at load time, since they are obtained from the user at run time.

As Pascal programs always execute under supervision of an operating system, externally existing files will have to be supported (in most cases) by that operating system or by its associated file system. This implies that setting up the above mentioned relationships must be done according to the ideosyncrasies of the underlying system.

In principle, just two procedures suffice to do the job:

```
CONNECT       relates an FV with an EEF,
DETACH(FV)    ends the connection.
```

The problem is in the parameters of CONNECT: one of them clearly is the FV. The rest must specify an EEF in a system dependent manner, and to be useful probably some extra information and system return codes.

I have received a preliminary copy of the manual for Mr. Noodt's implementation on the Sintran-III system for the NORD-10 computer, and he did a very good job on the system interface. He was able to provide a CONNECT procedure with only 3 parameters: the FV, a string specifying the name of the EEF, and an integer returning system provided file status. It must be added that Sintran-III is a very user-friendly system, in which files (including peripheral devices) are specified by a string with an internal syntax. (Buffering, blocking, file control blocks, etc. are provided by the system and transparent to the user by default.)

2. Several problems remain with Pascal I/O. Again, in interactive use (and as Mr. Noodt pointed out) any call of the kind

```
READ(F,I)     (*integer I*)
```

will crash the program if I is not given a string convertible to an integer. And again, fortunately the Sintran-III system lets a program find out whether or not it was called interactively, so that the following loop can be built into the run-time support system:

```
OK:=FALSE;
REPEAT
      READ(F,I);
      IF interactive AND error THEN BEGIN
                    WRITELN;WRITE('NOT AN INTEGER VALUE')
                    END;
      ELSE IF error THEN abort
          ELSE OK:=TRUE;
UNTIL   OK;
```

Further, Pascal adopts the philosophy that all variables must be initialized before their contents can be used. Although this is not a requirement, some systems go to great lengths to abort programs that access undefined values. This philosophy is in fact very good. But why are file buffers initialized automatically ? This exception of the rule of explicit initialisation leads to problems with character files connected to terminal inputs, as everyone knows. Why not insist on an explicit first GET ?

Finally, (and again for interfactive input mainly) why do READ and WRITE work in the way they do ? For batch jobs, the equivalence

```
READ(F,CH)    <==>    CH:=F↑; GET(F)
```

is acceptable, because you never notice anyway. Try to explain this to someone writing an interactive program ! I have now resigned to the simple recommendation: use GET, and do everything character by character yourself. It suffices to look at how the P4 compiler reads characters to be convinced that READ(F,CH) should be equivalent to GET(F); CH:=F↑ (just notice how the EOLN is delayed !)

3. The problem of the controlled variable in the FOR statement: Mr. John Nagle (Pascal News No. 12) writes that it should be truly undefined outside the FOR and proposes as a solution that it be considered as a variable declared local to the FOR. To this I can only remark :

a) many programmers, including myself, would in fact be happy with a truly defined value. There are many arguments for either case.

b) a language called ALGOL68 does exactly what Mr. Nagle proposes 10 years after its definition. In fact, many Pascalers, especially thosewho write in Pascal News, Sigplan Notices and other respectable periodicals as if they have discovered the Only True Religion, would in fact do well to look up the Algol68 report[1]). Nearly all the "problems" with Pascal that are so frequently discussed in these columns have a decent solution in Algol68. Yet somehow that language seems a taboo subject.

4. Mr. Nagle further addresses the problem of the GOTO. I have written a 3000 line program in Pascal without a single GOTO. However, the abolishment of the GOTO would mean programming with flags. It becomes then nearly impossible to program an efficient and understandable sequential machine (another taboo subject ?). How do we get out of inner loops that must be fast and therefore should not test flags ? Or is efficiency completely gone from our list of desirable program properties ?

Consider Knuth's article on programming with GOTOS[2]. Consider also the following program:

```
    type  T=record    ...     ...     next:  ↑T  end;
    var   head,p,newt:↑T;  found:Boolean;
    begin
          ...  ...
          p:=head; found:=FALSE;
          while  (not found) and (p<>nil)  do
                 if  p↑=newt↑ then  found:=TRUE
                 else   p:=p↑.next;
          if found then  this else theother;
```

The search can be written :

```
    1:  if  p<>nil then
               if  p↑=newt↑ then begin this;goto2 end
               else begin p:=p↑.next;goto1 end;
        theother;
    2:  ...  ...
```

The last version is even easier to explain. I am not advocating writing this particular example in the way I did. What I would much prefer to write is:

```
    loop
         if p=nil then theother;  exit endif;
         if p↑=newt↑  then this; exit
         else  p:=p↑.next
         endif
    endloop
    ... ...
```

But alas that is another programming language[3]. The removal of the GOTO is only practical when some new structures are added at the same time.

Since Von Neumann computer architecture is probably here for several more decades, we will continue to have machines on which it is much faster and more economical to program jumps than to program any other operation. IF-THEN-ELSE and the other control structures are nothing but elegant ways to safely write common combinations of jumps. Every practical program contains also combinations that can only be built efficiently by explicit jumps, i.e. GOTO's.

At CERN we have a continuous flow of students from the member states that spend some time here as apprentices. Those educated in Pascal come here with mental blocks against GOTO's, and overload their programs with flags of all colours. The flags create a software maintenance problem no less formidable than locally used GOTO's.

A flag has to be declared (like a label), it must be set initially (the label planted) and it must be correctly used (the GOTO's written). Where is the improvement ? Witness the many different uses of the global flag TEST in the <P> compiler.

As an aside, a lot of "flag-waving" or "GOTO-ing" is caused by the absence from Pascal of the conditional AND and OR operators. Since the Report does not solve the question of how

A and B

is evaluated, another heated discussion ensues: when A is FALSE, do we still want to evaluate B??Dijkstra's answer is: yes, because if we do not want to evaluate B, we write

A cand B

indicating clearly that B is only evaluated on the condition that A is TRUE. The example program reduces to :

```
      while  (p<>nil)  cand  (p↑<>new↑) do  p:=p↑.next;
      if  p=nil  then theother else this;
```

This still tests (p=nil) more than necessary, but at least the loop is fast. (Incidentally, can anybody provide me with a sound explanation of why the parentheses in the while expression are necessary ?)

Finally, if the GOTO must go, then why not also pointers ? They are far more dangerous !

5. Bugs in the portable P4 compiler:

a) the bug of the non-closed comment at the end of a program which produces an infinite loop printing the message

   **** EOF ENCOUNTERED

   can also be fixed in a more economical way by testing at the printing of the message that this printing occurs only once. That requires the inclusion of a STOP procedure or the setting of a flag (to be tested after the comment loop). Remembering that the compiler spends 80% of its time in the lexical scanner, that seems to pay.

b) the sentence at the bottom of page 8 in the Implementation Notes:

   "Also, storage allocation of data is according to the simple rule that consecutively declared entities are allocated the requisite number of consecutive storage units"

   is quite ambiguous. It is certainly not true that the declaration

```
          var   I,J,K:integer;
```

   leads to allocation of I,J,K in that order: the allocated order is K,J,I ! This is the case in several places, e.g. fields in records.

Thus

```
      type  T1=record   I:integer;  J:integer   end;
            T2=record   K,L:integer  end;
```

should declare two compatible types, but after

```
      var  X:T1;  Y:T2;
      ...
      begin
           Y:=X;
```

Y.L has the value of X.I ! Inspection of the compiler reveals where the lists I,J,K... are built, and it is sufficient to put in a line or two that turns them around.

References

1) Revised Report on the Algorithmic Language Algol68
A.Van Wyngaarden et al,
Sigplan Notices, vol. 12, No. 5, May 1977

2) Structured Programming with GOTO statements
D.E. Knuth,
Computer Surveys, vol. 6, No. 4, December 1974, pp. 261-301

3) Modula, a language for modular multiprogramming
N. Wirth,
Software-Practice and Experience, vol. 7, No. 1, Jan/Febr. 1977

Bibliography

- Ignorance of Algol69 considered harmful
R. Hamlet,
Sigplan Notices Vol. 12, No. 4, April 1977

- Can programming be liberated from the Von Neumann Style ?
AC Turing Award Lecture 1977,
J. Backus
Communications of the ACM, Vol. 21, No. 8, August 1978

Yours sincerely,

Robert Cailliau
PS Division

*****

People's Computer Company

P. O. Box E, 1263 El Camino Real, Menlo Park, California 94025, Telephone (415) 323-3111

October 22, 1978

Dear Mr. Mickel,

PASCAL NEWS readers may be interested to know of two special events related to the use of PASCAL in music applications.

There will be a lecture / demonstration on "PASCAL and Music" at the 1978 Fall DECUS Symposium (a meeting of users of Digital Equipment Corporation's computers) in San Francisco, in late November.

In addition, COMPUTER MUSIC JOURNAL will be running an article on the PASCAL language, with music applications, and a survey of the available PASCAL compilers. This article should appear in early January.

I'm looking forward to the next issue of PASCAL NEWS.

Best regards,

CR

C. Roads
Editor
COMPUTER MUSIC JOURNAL

People's Computers • Dr. Dobb's Journal of Computer Calisthenics & Orthodontia • Computer Music Journal

ICL

International
Computers
Limited

ICL Belgium
Avenue Lloyd George, 7
1050 - Brussels

MEMO

Date 7/11/1978 Your ref.: Ref n LOG/sm Tel ext.:

To
PASCAL User's Group
c/o Andy Mickel
University of Minnesota
Computer Center
208 S.E. Union Street
MINNEAPOLIS MN 55455
U.S.A.

From
Laurent O. Gelinier
ICL Belgium S.A.
Avenue Lloyd George 7
B-1050 BRUSSELS
Belgium

Andy:

The European Division of ICL is responsible for the first field trial of some new equipment designed for large distributed systems. This new equipment includes mainly:

- File processor: - 16-bit mini computer
  - large capacity disks
  - up to 1 Mega-byte of memory.
- Intelligent terminal: - 2 or more 8085 microprocessors
  - up to 64K of memory.

The field trial consists of 800 file processors and 4.000 terminals in a bank application.

We are currently looking for a high level language for "system" programming which would be implemented on both file processor and terminal. Specific application environments or programming tools would be built using this system tool, achieving hopefully ease of implementation, ease of maintenance and portability.

We are considering: - PL/M
- CORAL (UK standard)
- PASCAL.

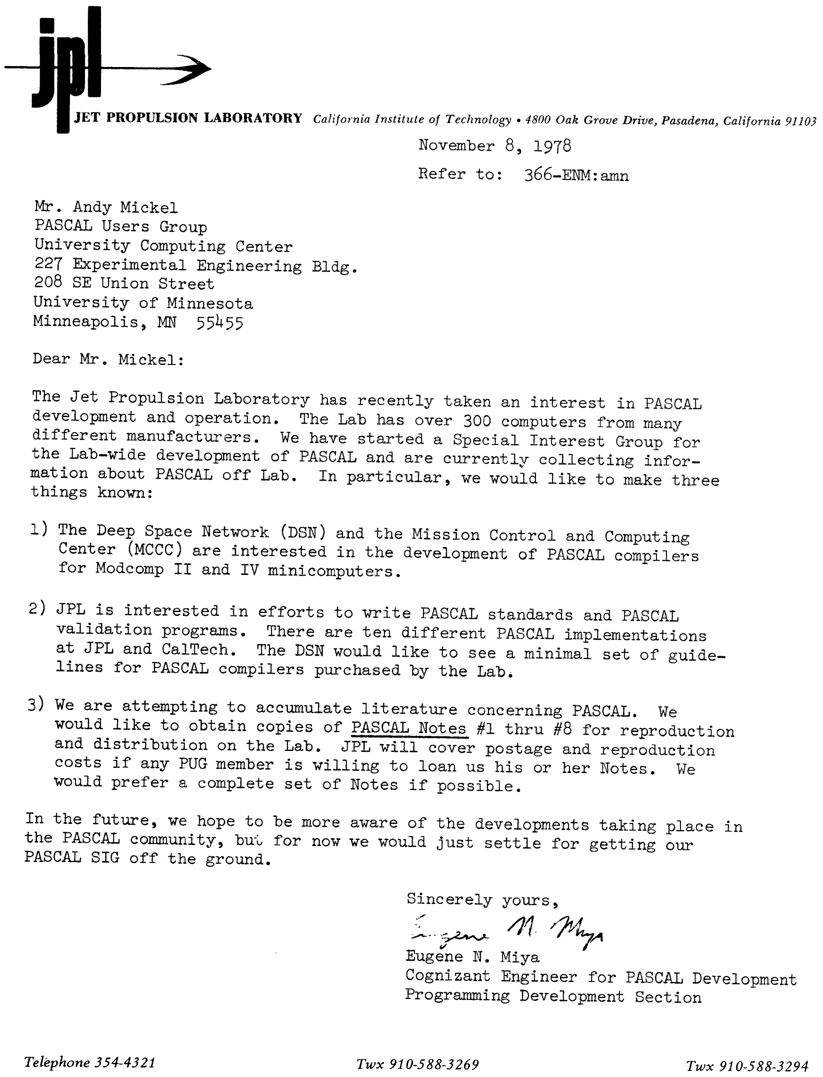
At this stage we have the basic documentation on PASCAL, mainly the language definition. But, in order to speed up the implementation of PASCAL on our machines, we would like to investigate the possibility of acquiring and using some existing PASCAL compilers.
More specifically, could you provide me with some documentation/information/references about:

- PASCAL compiler implmentations for the INTEL 8080/8085 (except the adaptation of the Hartmann's compiler to the INTEL MDS system)
- potentially "portable" PASCAL compilers.
- a possible PASCAL User's Group contact in Europe.

Regards,

laurent O. GELINIER

Laurent O. Gelinier



jpl

JET PROPULSION LABORATORY *California Institute of Technology • 4800 Oak Grove Drive, Pasadena, California 91103*

November 8, 1978

Refer to: 366-ENM:amn

Mr. Andy Mickel
PASCAL Users Group
University Computing Center
227 Experimental Engineering Bldg.
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455

Dear Mr. Mickel:

The Jet Propulsion Laboratory has recently taken an interest in PASCAL development and operation. The Lab has over 300 computers from many different manufacturers. We have started a Special Interest Group for the Lab-wide development of PASCAL and are currently collecting information about PASCAL off Lab. In particular, we would like to make three things known:

1) The Deep Space Network (DSN) and the Mission Control and Computing Center (MCCC) are interested in the development of PASCAL compilers for Modcomp II and IV minicomputers.

2) JPL is interested in efforts to write PASCAL standards and PASCAL validation programs. There are ten different PASCAL implementations at JPL and CalTech. The DSN would like to see a minimal set of guidelines for PASCAL compilers purchased by the Lab.

3) We are attempting to accumulate literature concerning PASCAL. We would like to obtain copies of PASCAL Notes #1 thru #8 for reproduction and distribution on the Lab. JPL will cover postage and reproduction costs if any PUG member is willing to loan us his or her Notes. We would prefer a complete set of Notes if possible.

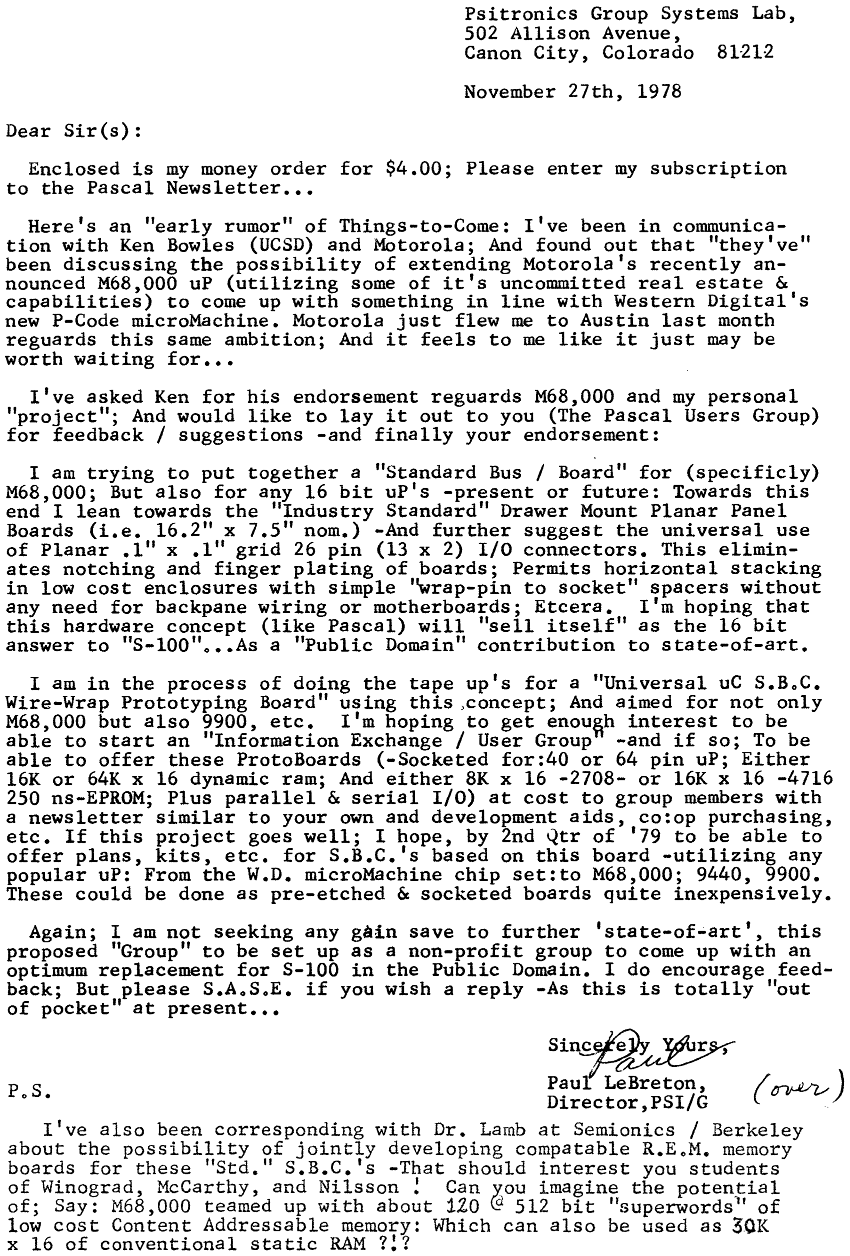
In the future, we hope to be more aware of the developments taking place in the PASCAL community, but for now we would just settle for getting our PASCAL SIG off the ground.

Sincerely yours,

Eugene N. Miya

Eugene N. Miya
Cognizant Engineer for PASCAL Development
Programming Development Section

Telephone 354-4321 *Twx 910-588-3269* *Twx 910-588-3294*



Psitronics Group Systems Lab,
502 Allison Avenue,
Canon City, Colorado 81212

November 27th, 1978

Dear Sir(s):

Enclosed is my money order for $4.00; Please enter my subscription to the Pascal Newsletter...

Here's an "early rumor" of Things-to-Come: I've been in communication with Ken Bowles (UCSD) and Motorola; And found out that "they've" been discussing the possibility of extending Motorola's recently announced M68,000 uP (utilizing some of it's uncommitted real estate & capabilities) to come up with something in line with Western Digital's new P-Code microMachine. Motorola just flew me to Austin last month reguards this same ambition; And it feels to me like it just may be worth waiting for...

I've asked Ken for his endorsement reguards M68,000 and my personal "project"; And would like to lay it out to you (The Pascal Users Group) for feedback / suggestions -and finally your endorsement:

I am trying to put together a "Standard Bus / Board" for (specificly) M68,000; But also for any 16 bit uP's -present or future: Towards this end I lean towards the "Industry Standard" Drawer Mount Planar Panel Boards (i.e. 16.2" x 7.5" nom.) -And further suggest the universal use of Planar .1" x .1" grid 26 pin (13 x 2) I/O connectors. This eliminates notching and finger plating of boards; Permits horizontal stacking in low cost enclosures with simple "wrap-pin to socket" spacers without any need for backpane wiring or motherboards; Etcera. I'm hoping that this hardware concept (like Pascal) will "sell itself" as the 16 bit answer to "S-100"...As a "Public Domain" contribution to state-of-art.

I am in the process of doing the tape up's for a "Universal uC S.B.C. Wire-Wrap Prototyping Board" using this concept; And aimed for not only M68,000 but also 9900, etc. I'm hoping to get enough interest to be able to start an "Information Exchange / User Group" -and if so; To be able to offer these ProtoBoards (-Socketed for:40 or 64 pin uP; Either 16K or 64K x 16 dynamic ram; And either 8K x 16 -2708- or 16K x 16 -4716 250 ns-EPROM; Plus parallel & serial I/O) at cost to group members with a newsletter similar to your own and development aids, co:op purchasing, etc. If this project goes well; I hope, by 2nd Qtr of '79 to be able to offer plans, kits, etc. for S.B.C.'s based on this board -utilizing any popular uP: From the W.D. microMachine chip set:to M68,000; 9440, 9900. These could be done as pre-etched & socketed boards quite inexpensively.

Again; I am not seeking any gain save to further 'state-of-art', this proposed "Group" to be set up as a non-profit group to come up with an optimum replacement for S-100 in the Public Domain. I do encourage feedback; But please S.A.S.E. if you wish a reply -As this is totally "out of pocket" at present...

Sincerely Yours,
Paul
Paul LeBreton,
Director,PSI/G

(over)

P.S.

I've also been corresponding with Dr. Lamb at Semionics / Berkeley about the possibility of jointly developing compatable R.E.M. memory boards for these "Std." S.B.C.'s -That should interest you students of Winograd, McCarthy, and Nilsson ! Can you imagine the potential of; Say: M68,000 teamed up with about 120 @ 512 bit "superwords" of low cost Content Addressable memory: Which can also be used as 30K x 16 of conventional static RAM ?!?

Dear Mr. Mickel,

Recently I've carried out an experiment in using Pascal for documentation. The problem was to specify the syntax of a graph produced by some phases of an optimizing compiler; previously it was fixed in a BLISS-like machine-oriented language, without any thought of such a documentation in Pascal, although with a certain idea of regularity in mind.

It was a pleasant surprise for me to discover how easily Pascal suited this purpose, and how informative it was of the intended use of the node attributes. In fact, there was only one minor problem, and this is what this letter is about.

I had to render in Pascal a double-variant node, i.e. a node which had two groups of variants, each group conditioned by an independent tag of its own. A less particular example might be

```
type person =
   record first name, name : alfa;
          age : 0..255;
   case sex : (male, female) of
      male : (enlisted : boolean);
      female : (maidenname : alfa);
   case position : (student, lecturer, assistant) of

      lecturer, assistant : (subject : (algebra, geometry);
                             degree : (none, phd, master));
      student : (year : 1..5; scolarship : integer)
   end;
```

This example presents the extention I've used in my document; namely, several variant parts are allowed at the same level, which are gathered at the end of the record definition.

Of course I could make the first variant part into a record field, and thus remain within the standard Pascal; but the very simplicity of this transformation calls for its inclusion into a compiler: this would eliminate the necessity to invent irrelevant field identifiers and repeat them in field selectors. Furthermore, alignment of all the variants at their logical level enables an intelligent compiler to produce a better packing.

I think that such multi-variant notions emerge quite naturally at a certain level of complexity. I could mention the file concept in which there are three logically independent variant groups conditioned by transmission mode (record, stream), buffering and function (input, output, update) - and e.g. attribute "keyed" is meaningful only within record mode; the concept of a variable in, say FORTRAN, which could have storage class and structure attribute groups etc.

Sincerely yours,

С. Покровский
21 Nov 1978

Sergei Pokrovsky
Computing Center
Novosibirsk 630090
USSR

****

SA SANDERS ASSOCIATES, INC.
OCEAN SYSTEMS DIVISION

FEDERAL SYSTEMS GROUP
95 Canal Street, Nashua, N. H. 03060
Telephone (603) 885-4321
Telex 094-3430

26 March 1979

Dear Andy:

I've been meaning to write for some time to express my gratitude for the way you've been steering PUG through the last few years, but your farewell letter in #13 really pushed me to action. Somehow you've been able to administer PUG through a period of rapid growth, organize the News and recruit good section editors, and mediate some thorny disputes over changes to the language. And all this was done on a volunteer basis! I think its obvious that we wouldn't have gotten as far as we have without your enormous energy and good humor. Thanks for everything.

By the way, the four PASCAL implementations we have here at Sanders show a remarkable diversity of ways to deal with TRUNC and ROUND for negative arguments. Here's a summary:

Implementation	TRUNC (-4.3)	ROUND (-4.3)
PDP-10 (Hamburg) Dec. '76 version	-5	-4
PDP-11 (Stockholm) Apr. '77 version	-4	-3
PDP-11 (OMSI) RSX V1.1F	-4	-4
NOVA (Manchester) Rev 2 Update 0	-5	-5
Correct Result (User Manual & Rept: p. 107)	-4	-4

(Newer versions of the first two have been issued and they may have corrected these errors.)

Best wishes,

Bill

Bill Marshall

THE UNIVERSITY OF NEBRASKA
COMPUTER NETWORK

LINCOLN PRODUCTION SERVICES
225 NEBRASKA HALL
LINCOLN, NEBRASKA 68588

TELEPHONE (402) 472-3701

February 9, 1979

Dear Andy,

This is a remedial letter to let you know of my change of address and to try to update the general knowledge of the status of Pascal at Nebraska. First the technicalities.

My old home address was: Curt Hill
7535 Sherman Drive
Omaha, NE 68134

My new home address is: 2314 Orchard St.
Lincoln, NE 68503

The business address remains the same. Now on to the good stuff.

Pascal is alive and well at the University of Nebraska, as we all might have suspected. We are now on our second semester of teaching computer science majors Pascal as their first and principal language. Progress in other majors who use programming is slower but coming along. The sure sign that it has caught on here is that thesis projects are being done in Pascal rather than the competition. Furthermore, I was asked to talk to the state chapter of IEEE on Pascal which shows that interest is spreading. As a part of the Computer Network, I also teach a three day (two hours a day) mini course to University users at large. Pascal is available on all three of the available large systems, and there are several copies of UCSD Pascal and other micro or mini versions.

I would also like to comment, for the record, on our compiler for IBM 360/370. We are using the Stanford implementation by Sassan Hazeghi and it is by far the best one we have looked at for our machines. It is very compatible with the standard, and Pascal-6000 programs usually run, only after massaging the character set (no ↑). The code generated is pretty good, and reliability excellent. I have managed to find two obscure bugs and both were quickly fixed. Anyone who has an older copy of the compiler should get July of 78 or newer version, if only for the nice symbolic dump for runtime problems. We implemented three compilers and looked at about three more and Stanfords was the clear winner.

Well that is the current status. I am sorry I did not get this out sooner for your use.

Sincerely,

Curt Hill

Curt Hill
Computer Programmer/Analyst II

CH/mw

DENISON UNIVERSITY

AREA CODE 614
587-0810

GRANVILLE, OHIO 43023

Founded 1831

DEPARTMENT OF MATHEMATICAL SCIENCES

March 8, 1979

Dear Andy,

I've been meaning to write this letter for some time, but the latest PASCAL News finally moved me to action.

First, I'm sorry you feel the need to get out from under. I'm sure that none of us realize fully how much work you have expended on this project, but I know that I for one appreciate it.

Second, I have some mixed emotions about the trend towards non-Standard (new Standard, Revised Standard, etc) PASCAL. I was particularly interested in Richard Cichilli's report on the UCSD workshop since it made me reconsider many of my views. Using his discussion on the desirability of an exponentiation operator, I freely concede that a function can be written, but by the same logic we could eliminate the multiplication and division since these could be handled by addition and subtraction. Similarly, three Boolean operators could be reduced to one (NAND, NOR) or two (AND - NOT, OR - NOT). On the other hand, implementing all the nice-to-have operations would create a PL/1 mess, something none of us want. Thus, it seems to me that the problem is to decide where to draw the line. My suggestion is to meet the problem by a compromise. Leave STANDARD PASCAL where it is, but define one or two supersets. My method would work as follows. Any PASCAL program which may be transported from one system to another must be written in the STANDARD version. Thus, we would have a language which is appropriate for teaching , for exchanging algorithms, etc. However, for some production programming in which a multiplicity of procedures may be required, have a PASCAL II. PASCAL II would have certain features added to it. External procedures, better I/O instructions, a few text handling instructions are obvious candidates. These would have to be as well defined as in STANDARD, but would not have to be implemented. Further, require that any PASCAL II compiler have all and only the specified options. Thus, a PASCAL II program would be transportable to any other PASCAL II system. By requiring that STANDARD PASCAL programs could also be compiled by a PASCAL II system, upward compatibility could be attained. Admittedly this implies some sort of certification, but I don't believe that this is unreasonable. Admittedly this is a compromise, but I believe that it may satisfy a majority of the users.

Finally, on a more philosophical note, I wonder if it is really possible to define a language without also defining implementation methods. The articles in PN#13 on evaluating Boolean expressions, and several articles over the last few years in IEEE Transactions on Software Engineering, have pointed out that two or more different implementations of language specifications can produce different results while remaining faithful to the definitions of the language.

Sorry this is so long, thus adding to your workload, but I wanted to throw in my two cents worth.

Sincerely yours,

James Cameron, Professor
Dept. of Mathematical Sciences

University of Illinois at Urbana-Champaign

Nuclear Physics Research Laboratory • 23 Stadium Drive • Champaign, Illinois 61820 • (217) 333-3190

March 13, 1979

Dear Andy and all PUG members,

I would like to reply to a few articles that I have seen in Pascal News. In particular, I would like to reply to Richard J. Cichelli. He has said that complex numbers "are easily created within the standard mechanisms of the language". As far as this statement goes, I agree. However, this only mentions creation, not use! No one argues that it is not possible to create a "complex" record type. But the standard does not allow simple usage of these records. In particular a function is only allowed results of "scalar, subrange, or pointer type". Given this restriction I would like the ivory tower types (i.e. people whose major source of income does not come from their ability to program computers (talking about does not constitute programming)) to use STANDARD Pascal to produce a simple, usable, and UNDERSTANDABLE optical potential calculation(this calculation relies heavily on complex arithmetic). I think this only goes to show a major weakness of Pascal. One of the reasons that I find Pascal so useful is the ease of creating complicated data types. But it is not always easy to use, and initialize these structures. In order to overcome these problems, I would like to suggest some additions to Pascal. I don't claim that these ideas are in a polished form, but I hope that they will stimulate discussion.

The first point, which is not new by any means, is that Pascal needs a method to initialize data, and in particular structured data. Whatever form this takes it should have the capability of allowing the data to determine the structure. The particular case that comes to mind is an array whose maximum subscript is determined by the number of data elements (table generation). The only way (that I know) of doing this is to use assembly language!

The second addition is structured type binary operators. *** A simple example should indicate what I mean by this.

```
     TYPE COMPLEX = RECORD  R,I  :  REAL  END;
     VAR  C1, C2, C3  :  COMPLEX;

     OPERATOR MPY ( Z1,Z2 : COMPLEX) : COMPLEX;
     BEGIN    MPY.R := Z1.R*Z2.R - Z1.I*Z2.I ;
              MPY.I := Z1.R*Z1.I + Z2.I*Z1.R
     END;

     BEGIN   ...  C3 := C1 MPY C2 ;   ...   END;
```

*** (* David A. Mundie suggested this idea in a letter dated 78/07/17. - Andy *)

While I don't think that it is realistic to use the standard operators (+,-,*, etc.) as structured operator names, it would certainly lead to simple expressions (C1*C2) such as are possible with FORTRAN. While I agree that this does not look all that different from "all type" functions, there are several points that should be made. Notably is the absence of the parenthesis forest that can exist from complicated expressions. This form should also make vector and array calculations easily implementable on vector computers. Also, for efficiency, it should be possible to have these operators expanded as a macro. And, it should be possible to "create" several like named operators which are distinguished by type (the standard operators are).

Another addition, which does not concern the language but rather the implementation, is the need for code optimizers. While it may be true that on most machines Pascal is as efficient as FORTRAN, this is certainly not true of the large mainframes like the CDC Cyber 74, the CDC 7600, and the Cray 1. As some members of PUG may know this class of computer does a substantial part of the scientific community's number crunching. Considering the present efficiency of Pascal compilers for these machines it is simply not economical to convert from FORTRAN. And this is one of those cases where one cannot say that this is caused by a dinosaur architecture. After all, the world's fastest computer can hardly be called a dinosaur. (I will note that it is unfortunate that a simple stack architecture cannot make sufficient use of parallel computation.) Maybe the dinosaurs in this case are the people who are unwilling to go beyond simple one pass compilation (for production programs).

I hate to have this sound like I have joined the ranks of those who want to add everything to Pascal, including the kitchen sink. I realize that it was just this way of thinking that created PL/1. I just find it difficult to promote a language that cannot in a simple, efficent, and understandable way handle calculations that are part of my everday life. And, I would like these comments to be taken in a positive light. I happen to like Pascal very much. It, among other things, makes it difficult to write sloppy programs. I wish I could understand why some (FORTRAN) people abuse the GO TO the way they do. I don't think even a sewer rat could decipher the logical (??) flow of some programs that I have been coerced to work on. Maybe when Pascal supercedes it predecessors this type of program will vanish!

Sincerely,

Roger L. Gulbranson

INTERACTIVE TECHNOLOGY INCORPORATED
14350 N.W. SCIENCE PARK DR. • PORTLAND, OR 97229
TELEPHONE (503) 644-0111
COMPUTER SYSTEMS CONSULTANTS

April 30, 1979

Dear Mr. Mickel,

I recently read your latest publication "Pascal News" with great interest. Our firm is simply ecstatic over recent articles and the general overall enthusiasm that is growing for Pascal. Our firm has spent many man months developing a Data Base Management System in Pascal plus developing business applications from our DBMS. I would like to expose to "Pascal News" just exactly what ITI has been up to these past few years and primarily of late.

First of all, two gentlemen on our staff began approximately two years ago (Bruce Johnson and Peter Mackie, formerly of Electro Scientific Industries and Tektronix, respectively) developing a Data Base Management System (DBMS) called "Realtime Database Manager" (RDM). Just a few quick "bullets" on RDM:

- Transportable from the LSI-11 through the VAX (Compatibility Mode). Same set of tools runs on all DEC PDP-11's.
- Runs under OMSI Pascal 1.
- Will run under DEC's RT-11, RSX-11, and RSTS/E.
- Operates with TSX (RT-11) allowing up to 8 users.
- Has complete routine of Forms Input or "ITI Prompt" which displays in most cases the format of the originating document.
- Interactive Report Generator or "ITI Inquirer". Accesses data bases with free form inquiry language that merely by typing English-like commands on a terminal, an operator can read, enter, delete, or modify data. Inquirer even gives special formatting capabilities, such as report titles, page and column headings, page numbering, data sorting by categories--even subtotals, totals, and averages. We have developed a product brochure for those interested in additional information. RDM is for sale in the market place at this time.

Secondly, to date ITI has proven that RDM and Pascal are very powerful tools for developing commercial oriented applications. One of many comments coming out of the DECUS meeting in New Orleans was that indeed Pascal is a viable higher level language but it is oriented to the education field and not in business applications field. We have disproved that "grossly"!! We have to date many successful applications going beautifully, and our programmer productivity is probably in the area of 10 to 1--seriously!! To date we have applications in General Ledger, Accounts Receivable, Accounts Payable, Order Entry - Inventory Control, Parts and Inventory for automotive dealerships and parts houses, Order Processing, and Payroll. By the time this reaches you and Pascal News, we will have generated many more applications.

Thirdly, we now are teaching formal classes in Introduction to Pascal (programming experience required), Advanced Pascal, and RDM and Pascal in data base management systems and how to use them. The Introduction class and Advanced class will run one week each. The RDM class (requires Intro) will run three days.

I look forward to your upcoming "Pascal News", and if I can be of additional assistance, don't hesitate to contact me.

Best regards,

B. J. Smith
Vice President, Marketing

*****

COMPUTER LABORATORY
THE UNIVERSITY
LEICESTER LE1 7RH

Telephone 0533-50000

Director of the Laboratory
D. L. Fisher, M.A., F.B.C.S., F.I.M.A.

From 18 June 1979
Tel: 0533 554455

PJH/AVD

20th July 1979

Dear Andy,

I am writing on behalf of the Numerical Analysts (although not one myself) here. It seems that a language without the ability to specify arrays of undefined bounds as formal procedure/function parameters cannot even be considered for replacing Fortran as it is then impossible to write generalised procedures/functions for dealing with arrays as is generally required. For this reason it would greatly aid our conversion to Pascal if such a standardised extension existed, and even more so if it were the same as that currently used by CDC 6000 Pascal 3.

Hoping this input is of use to you,

Yours sincerely,

Peter Humble.

STORAGE TECHNOLOGY CORPORATION

2270 South 88th Street / Louisville / Colorado 80027 (303) 666-6581 / TLX 4-5690

5 June 1979

Mr. Andy Mickel, Editor
Pascal News
University Computer Center: 227 EX
208 SE Union Street
University of Minnesota
Minneapolis, Minnesota 55455

Dear Andy:

I was delighted to meet you and Jim Miner in person at the ANSI/IEEE PASCAL Standards meeting in April in Boulder. Let me bring you and the readers of PASCAL News up to date on my professional involvement with PASCAL.

I am now working for Storage Technology Corporation in Louisville, Colorado. STC is a leading supplier of tape and disk devices in the IBM marketplace. STC has begun development of new products requiring software support. Our project has chosen PASCAL as a base for developing a system implementation language. The reasons for chosing PASCAL include the availability of a compiler (AAEC-IBM), the excellent characteristics of the language (syntax, sematics, programmer productivity, etc.), the ease of modifying the compiler, and the availability of expertise to support the language. Our intention is to maintain the proposed ISO standard for PASCAL as a proper subset of the language accepted by the compiler and to extend the language to aid the development of our project.

We are using as a base the Australian Atomic Energy Commission PASCAL compiler for IBM machines. Our experience with the compiler has been good, although we have encountered a number of minor bugs. I've been pushing our compiler group to report the bugs and fixes to the authors.

PASCAL distribution at the University of Colorado has changed since my departure. Steve Winograd carried on the distribution at the Computing Center from my departure in October until his in mid-May. In that time, he arranged for Wally Wedel at the University of Texas at Austin to distribute the CDC PASCAL compiler (Release 3) from the University of Minnesota. And he also arranged for Dr. William Waite of the Electrical Engineering Department to distribute the portable PASCAL compiler from Zurich and Per Brinch Hansen's Concurrent PASCAL. Thus the Computing Center is no longer associated with any PASCAL distribution activity.

In my spare time, I have worked on a number of large PASCAL programs. The first is a version of Adventure written in PASCAL. The original work was done on a CDC machine using the Release 2 Zurich compiler. Then I transported it to an IBM machine using our modified AAEC compiler. The IBM operating system is MVS with TSO. It took about two weeks of occasional work to accomodate the character set differences and compiler changes. Then the program executed perfectly on the first run. Even the interactive PASCAL solution used for the CDC system worked fine on the IBM system.

I believe there is a machine readable copy of my Adventure in Minneapolis. You have my permission to add it to the Release 3 distribution software if appropriate.

Another PASCAL program I've been working on is PASCAL-P. I've encountered a number of descrepancies between this compiler (and I assume the CDC compiler too) and the proposed ISO standard. The compiler does not restrict the usage of subrange variables passed thru VAR formal parameters. A subrange of integer variable may be used as an actual parameter for a VAR integer formal parameter. There will be no subrange assignment check within the procedure.

The other error is in passing elements of a packed structure thru VAR formal parameters. This is obviously impossible (and the CDC compiler prohibits) passing of a field which is less than a full word. However, the standard prohibits but the compiler allows passing a field that exactly occupies one word.

Other errors in the PASCAL-P compiler are as follows:

1) An element of a packed structure is passed thru a VAR formal parameter. A quick fix is to remove the word PACKED from line PASCP.127.

2) Although most compilers don't check identifiers to more than 8 or 10 characters, the identifier STRINGCONSTSY at line PASCP.813 should have the SY removed.

3) The three changes here are due to passing a subrange of integer variable thru a VAR formal parameter of type integer. Sometime an integer actual parameter is used.

```
   Line P.117:  Change INTEGER to ADDRRANGE
   Line P.166:  Change type of LSIZE from INTEGER to ADDRRANGE
   Line P.305:  Change type of LSIZE from INTEGER to ADDRRANGE
```

4) For bootstrapping on a CDC machine, the set range here is correct. But once on the target machine, change 0..58 to SETLOW..SETHIGH at line PASCP.2517.

5) This is not really an error but a limitation of the AAEC compiler. The static nesting of the PASCAL-P compiler is to deep for the AAEC compiler. This can be fixed by moving the procedure headings and declaractions for SIMPLEEXPRESSION and TERM to PASCP.2650 and PASCP.2705.

Other departures from the proposed ISO standard are as follows:

1) The sequence

```
     TYPE P       = @ INTEGER;
          INTEGER = REAL;
     VAR Q : P;
```

results in Q having type pointer to integer.

2) Assignments to FOR loop variables are not checked in even the most obvious cases.

3) (I) is not recognized as an expression when passed as an actual parameter for a VAR formal parameter.

4) File types are not implemented.

5) PACKED attribute is ignored so that use of the standard procedures PACK and UNPACK is impossible.

6) The tag field in variant records cannot be omitted.

I hope this information is of use to other uset of PASCAL.

Sincerely,

George H. Richmond

George H. Richmond
Storage Technology Corporation
P. O. Box 98, Mail Drop 93
Louisville, Colorado 80027
(303) 497-6375

THE ROCKEFELLER UNIVERSITY
1230 YORK AVENUE · NEW YORK, NEW YORK 10021

June 7, 1979

Dear PUG,

Enclosed please find dues for PUG membership, and also some extra for back issues of Pascal News.

We are a fairly recent Pascal installation. We have obtained OMSI's Pascal, running on DEC's RT-11 operating system for the PDP-11, and are extremely pleased with it. In addition, the University's central computer facility has a version of Pascal, running under Unix.

I will be teaching a course in Pascal in the fall. In the past, the staff of the computer center has given a FORTRAN course yearly, but their enthusiasm for this language seems to be waning. I hope that my course can fill the need for the novice computer user who wants to know how to handle his data (we tend to do a lot of data analysis, of various sorts).

In my fall course, I want to present "vanilla" Pascal (i.e., in its purest, most standard form). I have a question concerning the use of the PROGRAM declaration. It is not clear from the Report (or the standardization summary in Pascal News #14) whether this declaration is required or not (OMSI Pascal, for example, does not require it). Furthermore, is one required to declare FILE variables as PROGRAM parameters? I am not sure about the logic behind this - I can understand the PROGRAM as a kind of super-PROCEDURE, perhaps with its "parameters" coming (in some unspecified way) from the operating system (i.e. by assigning real files to file variables), but I am not sure this is correct. I would like to have a cogent explanation before I get asked the embarrassing question!

The problems about standardization that have appeared in Pascal News are very well taken, particularly as programs are exchanged between users. I recently received a copy of the tape prepared by the DECUS Pascal SIG. Besides being written using an obsolete tape format (so-called DOS format, rather than ANSI-standard, which both RI-11 and RSX support), with variable word-length blocks (even assuming you can read "raw" tape in Pascal, how could you handle different block sizes?), almost all of the utilities on the tape use some non-standard features which the OMSI compiler could not handle (the most common were the LOOP construct, and a Unix-like method of passing file names to the program, which is most opaque!). Some of these programs were adopted from Pascal News (with credit given) - why did the implementer (who was presumably making the program available, not for himself, but for others) choose to include these non-standard features? Please keep up the pressure to prevent a proliferations of pseudo-Pascals, which will only serve to fragment the user community.

I look forward to future Pascal (and Pascal News) developments.

Sincerely,

BSchor

Bob Schor

huntec
('70) LIMITED

25 HOWDEN ROAD,
SCARBOROUGH,
ONTARIO, CANADA
M1R 5A6
PHONE (416) 751-8055
TELEX 06-963640
CABLE HUNTOR, TORONTO

Huntec ('70) Limited,
c/o Room 431, B. I. O.,
Box 1006,
Dartmouth, Nova Scotia
Canada B2Y 4A2
June 29, 1979

Tony Addyman,
Department of Computer Science,
University of Manchester,
Oxford Road,
Manchester, England
M13 9PL

Dear Sir:

I am most interested in the application of Pascal to scientific and mathematical problems, especially time series analysis, on processors ranging from micros to large mainframes. I can see a problem in the application of the Pascal defined in the draft BSI standard to these areas. This is the absence of variable dimensions for arrays which are variable formal arguments of procedures. Object code libraries everywhere contain numerous Fortran routines which use this feature to perform operations on arrays and matrices of arbitrary size, and the very generality with which the capability is used argues for its serious consideration as part of Pascal, or as a conventionalized extension. I do not, however, believe that it should necessarily be coupled with variable dimensions for all arrays. This would introduce complications in the implementation which might be serious, especially on very small systems.

A second area which could use better definition is an EXTERNAL statement for seperately compiled procedures. Such procedures are almost universally used in actual production environments, and contribute to the modularization of software by seperating modules physically and psychologically. There are good arguments for allowing no communication of data to seperately compiled procedures except through the argument list. In most cases, the effect of the external procedure should not be changed by compiling it with the mainline routine.

I gather from Wirth's letter in Pascal News #13 p.82 that these matters are already under consideration. Add my name to the list of those who support these extensions.

Jack Dodds

cc. Rick Shaw
Pascal News

UNIVERSITY OF MINNESOTA
TWIN CITIES

University Computer Center
227 Experimental Engineering Building
Minneapolis, Minnesota 55455

1979 September 20

Ken Bowles
Institute for Information Systems - C-021
University of California - San Diego
La Jolla, CA 92093

Dear Ken,

Thank you for Newsletter #4 of the Pascal project at UC San Diego. It and the previous newsletters have provided us with news of your and your group's plans. Because subscribers to UCSD Pascal also have a need for general information about Pascal, I was wondering if you could insure that SofTech Microsystems will continue to send out Pascal User's Group ALL-PURPOSE COUPONS with the software package. Thanks!

Newsletter #4 carries the implication that UCSD Pascal in and of itself constitutes a viable "standard" for Pascal?! Also, suppose Niklaus Wirth trademarked Pascal, copyrighted the P2 portable Pascal system (on which UCSD Pascal was derived) and set up a marketing organization? There most probably would be no Pascal user community right now--nor, for that matter any UCSD Pascal!

Pascal-P is public-domain software. The least you could do is provide adequate recognition/credit/modification history in the comments of the source of the UCSD Pascal compiler. Right now I'm looking at a listing which has:

```
(*********************************************)
(*                                           *)
(*       UCSD  PASCAL  COMPILER              *)
(*    BASED ON ZURICH P2 PORTABLE            *)
(*    COMPILER, EXTENSIVLY  (sic)            *)
(*    MODIFIED BY ROGER T. SUMNER            *)
(*    SHAWN FANNING AND ALBERT A. HOFFMAN    *)
(*    1976..1978                             *)
(*                                           *)
(*    RELEASE LEVEL: I.3 AUGUST, 1977        *)
(*                   I.4 JANUARY, 1978       *)
(*                   I.5 SEPTEMBER, 1978     *)
(*                                           *)
(*    INSTITUTE FOR INFORMATION SYSTEMS      *)
(*    UC SAN DIEGO, LA JOLLA, CA 92093       *)
(*                                           *)
(*    KENNETH L. BOWLES, DIRECTOR            *)
(*                                           *)
(*    COPYRIGHT (C) 1978, REGENTS OF THE     *)
(*    UNIVERSITY OF CALIFORNIA, SAN DIEGO    *)
(*                                           *)
(*********************************************)
```

Contrast this to the entry in the PDP-11 Pascal compiler from Tampere, Finland produced by Jyrki Tuomi and Matti Karinen at the Tampere University of Technology:

2

```
(*******************************************************
*                                                      *
*                                                      *
*    STEP-WISE DEVELOPMENT OF A PASCAL COMPILER        *
*    ******************************************        *
*                                                      *
*                                                      *
*    STEP 5:   SYNTAX ANALYSIS INCLUDING ERROR         *
*              HANDLING; CHECKS BASED ON DECLARATIONS  *
*                                                      *
*    AUTHOR:   URS AMMANN                              *
*              FACHGRUPPE COMPUTERWISSENSCHAFTEN       *
*              EIDG. TECHNISCHE HOCHSCHULE             *
*              CH-8006 ZUERICH                         *
*                                                      *
*    ADAPTED   TO GENERATE CODE FOR A PDP 11 BY:       *
*              W. DE VRIES                             *
*              UNDER GUIDANCE OF DRS C. BRON           *
*              VAKGROEP  INFORMATICA                   *
*              TECHNISCHE HOGESCHOOL TWENTE ENSCHEDE   *
*              APRIL '75                               *
*                                                      *
*    CHANGED   TO RUN UNDER RSX-11M BY:                *
*              SEVED TORSTENDAHL                       *
*              TELEFONAKTIEBOLAGET LM ERICSSON         *
*              S-126 25  STOCKHOLM                     *
*              APRIL '77                               *
*                                                      *
*    MODIFIED  SLIGHTLY FOR PDP-11/70 UNDER IAS BY:    *
*              JYRKI TUOMI AND MATTI KARINEN           *
*              TAMPERE UNIVERSITY OF TECHNOLOGY        *
*              COMPUTING CENTER                        *
*              SF-33100 TAMPERE 10                     *
*              OCTOBER '77                             *
*                                                      *
*******************************************************)
```

The first 14 lines of this heading are those from the original Pascal-P2 compiler which give credit to Urs Ammann on whose back 90% of the Pascal compilers in the world are now riding, thanks to his labor and dedication.

Sincerely,

Andy

Pascal Standards

In this section are reports by Jim Miner, Rich Cichelli, and myself on this year's whirlwind of standards activity which has consumed so much of our time and was a major reason that this issue is late. We had wanted to provide a much-postponed report on the International Working Group on Pascal Extensions--Olivier Lecarme has written an excellent summary (in French) for the Bulletin of the AFCET Pascal Sub-Group. That will have to wait until issue #17 unfortunately, because the translation is not complete yet. Our current work in the Working Group about conformant array parameters is about to be pre-empted by the ISO Pascal Standards activities, and so Arthur Sale will have some information for us in issue #17. Information on the Validation Suite concludes this section.

Pascal Standards Progress Report

Jim Miner, with Tony Addyman, Andy Mickel, Bill Price, and Arthur Sale

This Report is divided into two main sections. The first deals with the international standardization effort, the second with national efforts, primarily in the United States.

One topic not addressed in this report is the political and organizational maneuvering which inevitably occurs in standards work. To get some ideas about this aspect read the pieces by Andy Mickel and Rich Cichelli following this report.

The ISO/BSI Standard

The history of the British Standards Institution (BSI) work on an international standard is covered in Pascal News #14 up through late 1978. Since then, the Working Draft 3 developed by BSI's DPS/13/4 was slightly revised and submitted to the International Standards Organization (ISO) subcommittee TC97 SC5. (See the accompanying glossary of standards group names.) The revisions to Working Draft 3 were mainly formalization of language (such as changing "is" to "shall be") and section renumbering. Working Draft 3 was printed in Pascal News #14 and subsequently in Software - Practice & Experience 9 (May 1979), pages 381-424.

The revised draft submitted to SC5 was given the document number "N462". (This document was published in the IEEE's Computer, April 1979, pages 68-82.) N462 was distributed in February by SC5 to its members for comment. Official comments were received by the British (through ISO channels) from several countries including Japan, the United States, Canada, the Netherlands, and Austria.

In addition to the "official" comments, DPS/13/4 has received a large volume of comments from the public. The massive task of examining these comments has been accomplished, and DPS/13/4 met this September to decide on changes to be included in the next draft (Working Draft 4). We expect this draft to be distributed in October through ISO for additional comments.

Working Draft 4 will be the subject of discussion at an ad hoc "Pascal experts group" meeting to be held in Turin, Italy in November. This group will advise SC5 (which meets at the same time) concerning further processing of the BSI working draft. It is not clear at this time what the outcome of the SC5 meeting will be, but the most likely result seems to be that the experts group will offer a revision of Working Draft 4 (with correction of errors) to SC5, and that SC5 will vote to register it as a Draft Proposal. If this occurs, the Draft Proposal will be circulated to SC5 member bodies for voting. The voting period is nomally three months, but precedent exists for fixing a longer period. Each SC5 member may vote "Yes", "Yes but please clarify ...", or "No because of ...". Negative votes must include specific objections. If these objections can be resolved then the "No" vote becomes a "Yes" vote. When a Draft Proposal is accepted by SC5 it goes into the next stage of voting as a Draft International Standard (DIS). When a Draft Proposal is not accepted, it will normally be revised and go through another round of voting.

Another possible outcome of the Turin meeting is agreement of the BSI to produce and circulate another Working Draft for comment only. This might significantly delay the international standard because SC5 does not meet often and business between meetings must be conducted by letter. Also, working drafts are not normally circulated before the Draft Proposal stage. The United States, which initiates most standards in this field, usually proceeds directly to the Draft Proposal stage. So, precedent firmly established by the United States in previous standards efforts argues against another Working Draft.

A third possible outcome is the establishment by SC5 of an international Working Group to attempt resolution of remaining problems in the Working Draft. This usually turns out to be expensive and time-comsuming.

A fourth possibility is that the BSI could postpone or even drop the ISO effort and concentrate on development of a British standard. The United States often develops an American National Standard before initiating ISO consideration. Unfortunately this is seen by some non-U.S. groups as coercion by the U.S. reflecting an unfriendly attitude to the rest of the world. This route would also result in a significant delay in obtaining an international standard.

Standards Activities in the United States

As reported in Pascal News #13, the American National Standards Committee on Computers and Information Processing (ANSI/X3) has established a Technical Committee on Pascal called X3J9. About the same time, the Institute of Electrical and Electronics Engineers (IEEE) established a Pascal standards project and committee called P770. X3J9 met initially in December 1978 in Washington D.C. (See the accompanying piece by Rich Cichelli about that meeting.) The IEEE committee met in January 1979 in San Francisco. Both of these meetings were primarily organizational.

Since then, both committees have met jointly in Los Angeles (February), Boulder (April), New York (June), and Houston (September). (In the rest of this report we will call this joint committee "X3J9".) Attendance at these meetings has averaged about 70 persons, perhaps half of which are official voting members. All such meetings are open to the public.

At the February meeting, discussion centered on the creation of an "SD-3" document. The SD-3 is a proposal to initiate a standards project, and outlines the nature of the standard desired, expected benefits of the standard and feasibility of its development, committee program of work, etc. X3J9 needed to submit such a proposal in order to work on an American National Standard, even if the result were identical to the ISO standard.

A final SD-3 proposal (printed below as subsequently modified by SPARC) was agreed upon at the April meeting. This document was submitted to X3 and SPARC for approval. Perhaps its most important feature is the stated intention that the (first) American National Standard should be compatible with the ISO standard.

A second immediate concern at the February meeting was the creation of a means for reviewing the British Working Draft then being circulated through ISO. X3J9 established a Technical Review Task Group (TRTG) under the direction of Bill Price to coordinate this review.

A third area of concern at the February meeting was the establishment of a mechanism for exploring extensions to Pascal. The proposed SD-3 mentioned above states this concern as seeking to "identify and evaluate common existing practices in the area of Pascal extensions." To create such a mechanism, X3J9 agreed to set up an Extensions Task Group (ETG) under the direction of Jim Miner. However, X3J9 also prohibited consideration of extensions during the initial review of the working draft (N462).

The April meeting was spent almost entirely on discussion of N462 and public comments on it which were received by X3J9. (The TRTG had met a week earlier in San Francisco to compile a draft response to the British.) After several exhausting rounds of discussion X3J9 agreed in principle to a response, but due to insufficient prior notice the committee was not able to generate an official response to the British.

By the time X3J9 met again in New York in June, more comments had been received. After another set of exhausting sessions X3J9 agreed on a final official response to the British draft: a 50+ page, very detailed document. (I think we are all indebted to Bill Price for the effort he put in on this review process!)

The June meeting also saw the development of proposed Procedures and Policy statements to guide the X3J9 extensions work.

In August, SPARC recommended to X3 that the X3J9 SD-3 be approved, but without provisions for developing an extended standard. In order to pursue an extended standard, X3J9 prepared a second SD-3 at its September meeting in Houston. Although not given final approval (because of lack of prior notice), it is expected that this document will be approved and sent to SPARC and X3 in November. The document tentatively agreed on in Houston is printed below.

X3J9 also came closer in Houston to agreement on procedures to cover extensions work. These procedures call for publicly soliciting proposals for extensions. The proposals may vary in content from merely stating an area of need for a capability in the language, up to a "formal" proposal including the following: a problem statement, specific revisions to the Standard Pascal document, syntax, semantics both in English and using some formal technique such as axioms, examples of use, implementation details, summary of experience using the extension, discussion of consistency with the existing language and expected benefit of the extension, and a list of related documents. Given the extensive detail needed in a formal proposal, I expect that most proposals will be relatively informal.

A library of "candidate extensions" will be maintained. These extensions will be those judged to be technically sound and desirable by X3J9. The library will be used later as the source of language features which may be included in an extended language. X3J9 has not established procedures for the synthesis of an extended language from these individual features.

Other National Standards Efforts

Several of us have been puzzled by the lack of official comments on N462 from several countries, including France and Germany. We have been told that Albrecht Biedl organized a technical committee which met in late May or early June to prepare some official German comments. Apparently the German standards organization (DIN) requires that such comments be reviewed by the next-higher committee before being submitted to ISO, and this committee will not meet until later this year.

We hope standards workers in more countries will report on their activities in future issues of Pascal News.

X3J9 Chair: Marius Troost, Sperry Univac

P770 Chair: Bruce Ravenel, Language Resources

Vice Chair (both committees): Scott Jameson, Hewlett-Packard

Secretary (both committees): Jess Irwin, Gould-Modicon

X3J9 International Representative: David Jones, Control Data

All correspondence with or about the committee may be addressed to:

Jess Irwin
c/o X3 Secretariat
CBEMA: Suite 1200
1828 L Street NW
Washington D.C. 20036

ISO - International Standards Organization.

ISO TC97 - ISO Committee on Computers and Information Processing.

ISO TC97 SC5 - ISO TC97 Sub-Committee on Programming Languages.

Draft Proposal (DP) - A document under consideration by ISO TC97 SC5.

Draft International Standard (DIS) - A document in a second stage of consideration by TC97 and all of ISO.

ANSI - American National Standards Institute.

ANS - American National Standard, which is a standard issued under the umbrella of ANSI.

dpANS - draft proposed American National Standard, a document on its way to becomming an ANS.

X3 - The committee recognized by ANSI for the area of Computers and Information Processing.

SPARC - Standard Planning and Requirements Committee, which advises X3 on functional and economic (not technical) aspects of new standards projects and review of proposed standards.

X3J9 - X3 Technical Committee on Pascal, which does the technical work on an American National Standard Pascal, and which advises X3 on the international standardization of Pascal.

IEEE - Institute of Electrical and Electronics Engineers.

IEEE Pascal Standards Committee - The committee established under IEEE standards project P770 to develop an IEEE Pascal standard.

JPC - Joint Pascal Committee, which is an unofficial term for the joint workings of X3J9 and the IEEE Pascal Standards Committee.

ANS Pascal SD-3 As proposed by X3J9 (X3J9/79-026) and amended by SPARC.
Subject to approval by X3.

Proposal for an American National Standard (ANS) Programming Language Pascal

1. IDENTIFICATION

1.1 Title:

ANS Pascal

1.2 Proposer:

Proposed by the X3 Technical Committee on Pascal (X3J9)

1.3 Date of Submission:

2. DESCRIPTION

2.1 Purpose:

The purpose of the standard is to provide an unambiguous and machine independent definition of the language Pascal.

2.2 Goal:

The goal is an implementable Pascal standard.

2.3 Nature of the standard:

A standard for a digital computer programming language.

2.4 Scope:

The programming language Pascal is a simple high-level language. It is a general-purpose rather than an all-purpose language. Pascal is being used increasingly in three areas:

1) The writing of system software

2) The writing of application software

3) The teaching of programming

2.5 Program of Work:

1) Maintain a liaison with the ISO, BSI and IEEE Committees to work toward a common working draft standard. This work should include review of those bodies' documents and forwarding of comments based on that review. The eventual draft proposed ANS Pascal shall be compatible with any ISO Pascal standard and identical in content with the jointly developed proposed IEEE Pascal standard.

2) Provide a means for review of all Pascal standardization activities.

3) Carry out the development of a Pascal standard.

4) Identify and evaluate common existing practices in the area of Pascal extensions.

5) Act as a liaison group with organizations interested in interpretation of ANS Pascal.

3. EXPECTED BENEFITS

3.1 Intrinsic:

Development of a standard Pascal reduces costs of extra training for a particular Pascal implementation and costs of conversion when transporting a program to a different machine.

3.2 Interchange:

A standard Pascal will facilitate portability.

3.3 Educational:

A standard Pascal enables production of educational documents or manuals usable with any standard implementation. Costs of re-education for a different implementation are reduced.

3.4 Economic:

While no estimates of economic impact are available at this time, it is felt that because of Pascal's widespread popularity, the economic benefits of a standard will be commensurately large.

4. DEVELOPMENT FEASIBILITY

4.1 State of the Art:

The most important factor in this proposal is the timeliness of the standardization of Pascal. Pascal has been implemented on a large number of different computers. If the problems relating to the definition of Pascal are not resolved in the very near future, there is a danger that the various implementations will become incompatible. The growth of a large number of incompatibilities would severely hinder any subsequent standardization activities.

The current lack of any significant incompatibilities should be seen as a good reason for standardization now.

4.2 Available Resources:

There are already three working groups concerned with the production of a Pascal standard. They are:

Pascal User's Group	(International)
DPS/13/4	(United Kingdom)
International Working Group on Pascal Extensions	(UK/USA)

These three groups are cooperating with each other and are corresponding with interested parties in the following countries: USA, Australia, Canada, Denmark, France, Germany, Poland, Sweden, and Switzerland. Many of these correspondents are suppliers of Pascal compilers.

Bibliography:

Jensen, K. and Wirth, N. (1978) Pascal - User Manual and Report, 2nd ed. (Springer-Verlag, New York)

Hoare, C.A.R. and Wirth, N. (1973), An axiomatic definition of the programming language Pascal, Acta Informatica 2, 335-55

Haberman, A.M. (1974), Critical comments on the programming language Pascal, Acta Informatica 3, 47-57.

Lecarme, O. and Desjardins, P. (1975), More comments on the programming language Pascal, Acta Informatica 4, 231-45

Welsh, J., Sneeringer, W.J. and Hoare, C.A.R. (1977), Ambiguities and insecurities in Pascal, Software-Practice and Experience 7, 685-96

Wirth, N. (1975), An assessment of the programming language Pascal, SIGPLAN Notices 10, 23-30

Wirth, N. (1971), The programming lnaugage Pascal, Acta Informatica 1, 35-63

Wirth, N. (1971), The design of a Pascal compiler, Software-Practice and Experience 1, 309-333

Wirth, N. (1972), The programming language Pascal and its design criteria, Infotech State of the Art Report 7: High Level Languages, 451-473

Hoare, C.A.R. (1973), Hints on programming language design, Stanford University Computer Science Dept. Report 403

Wirth, N. (1974), On the design of programming languages, North Holland Information Processing: Programming Methodology

Wirth, N. (1976), Programming languages: What to demand and how to assess them, and Professor Cleverbyte's visit to heaven, ETH Institute fur Informatik, Technical Report 17

4.3 Estimated Costs:

The cost of developing a Pascal standard will be borne by the sponsors of the membership. It is difficult to estimate the total cost as membership totals will undoubtedly fluctuate.

The total cost is expected to be on the order of $500,000.00

5. IMPLEMENTATION FEASIBILITY

5.1 Supplier Conformance Considerations:

In developing the Pascal standard, care will be taken to maintain machine independence. The final specification will encourage unambiguous interpretation. The above goals, in addition to the participation of many suppliers in the standardization effort, should provide an opportunity to achieve and/or determine conformance. Note that a suite of programs is currently being developed by groups based in Australia and the U.K. which could form the basis of a conformance test.

5.2 User Operational Considerations:

The current lack of widespread incompatibilities in existing practice should make conversion of existing programs a minimal expense.

5.3 Legal Considerations:

Preserving machine independence and compatibility with any ISO Pascal standard should prevent problems related to restaint of trade and public interest.

5.4 Estimated Costs:

Implementation may necessitate some modification of existing Pascal compilers and programs. No detailed cost figures can be developed at this time. However, the announced goals and constraints of this standardization effort should hold such necessary modifications to a minimum.

6. MAINTENANCE REQUIREMENTS

6.1 Extent and Frequency of Anticipated Changes:

X3J9 intends to provide interpretation and clarifications of the eventual ANS Pascal standard as the need arises.

The committee also intends to comply with the requirement that an ANSI standard be reviewed within a five year period.

6.2 Resources:

The committee accepts its responsibility to maintain the eventual standard and to continue this activitiy along with any revision efforts.

6.3 Cost:

The cost of maintaining the standard on an annual basis is estimated to be comparable to the original development cost.

7. CLOSELY RELATED STANDARDS ACTIVITIES

As mentioned previously, ISO is undertaking the development of a Pascal standard. The Technical Committee will maintain close liaison with this group to assure that the resulting standards define the same language.

The IEEE P770 Committee is developing the ANS Pascal standard jointly with X3J9.

8. RECOMMENDED TIME FRAME

Every effort will be made to submit a candidate standard to X3 by June 1, 1979.

*****

ANS EXTENDED PASCAL SD-3, September 14, 1979

X3J9/79-187
(Revised)

Proposal for an American National Standard (ANS) Extended Programming Language Pascal.

1. IDENTIFICATION

1.1 Title:

ANS Extended Pascal

1.2 Proposer:

Proposed by the X3 Technical Committee on Pascal (X3J9)

1.3 Date of Submission:

2. DESCRIPTION

2.1 Purpose:

The Extended Pascal standard is intended to define areas in which Pascal may be reasonably extended in a machine-independent and unambiguous manner consistent with existing practice.

2.2 Goal:

The goal is an implementable, internationally acceptable, Extended Pascal standard. The Extended Pascal standard is intended to replace the standard referred to in 7(a).

2.3 Nature of a standard:

The standard shall define extensions to the ISO Pascal standard and the corresponding ANS standard.

2.4 Scope:

The standard shall encompass those Pascal extensions found to be:

(a) compatible with the Pascal language referred to in section 7(a), and
(b) beneficial with respect to cost.

2.5 Program of work:

The program of work shall include:

(a) solicitation of proposals for extended language features;

(b) the critical review of such proposals;

(c) synthesis of those features found to be acceptable individually and which are mutually consistent into a draft proposed standard;

(d) interface with all interested standards bodies, both domestic and international;

(e) submission of draft as a dpANS and as an ISO draft proposal.

3. BENEFITS

3.1 Intrinsic:

Development of a standard Extended Pascal reduces costs of extra training for a particular Extended Pascal implementation and costs of conversion when transporting a program to a different machine.

3.2 Interchange:

A standard Extended Pascal will facilitate portability.

3.3 Educational:

A standard Extended Pascal enables production of educational documents or manuals usable with any standard implementation. Costs of reeducation for a different implementation are reduced.

3.4 Economic:

While no estimates of economic impact are available at this time, it is felt that because of Pascal's widespread popularity, the economic benefits of a standard will be commensurately large.

4. DEVELOPMENT FEASIBILITY

4.1 State of the Art:

There is growing sentiment in both consumer and producer communities that Pascal should be extended. A wide variety of extensions are available in currently existing language processors. Without a standard for an extended language, these processors will become increasingly incompatible.

There have been previous efforts on extensions by the UCSD Workshop on Pascal Extensions for Systems Programming and the International Working Group on Pascal Extensions. These efforts have shown that consensus can be reached on at least some extensions.

4.2 Resources:

The membership of X3J9 shall be a resource for this draft. In addition, cooperation and consultation with other standard bodies and Pascal experts shall be sought.

Bibliography:

Pascal News

ACM SIGPLAN Notices

Software Practice and Experience

4.3 Estimated Costs:

The cost of developing an Extended Pascal standard will be borne by the sponsors of the membership. It is difficult to estimate the total cost as membership totals will undoubtedly fluctuate.

The total cost is expected to be on the order of $500,000.00 per year.

5. IMPLEMENTATION FEASIBILITY

5.1 Supplier Conformance Considerations:

In developing the Extended Pascal standard, care will be taken to maintain machine independence. The final specification will encourage unambiguous interpretation. The above goals, in addition to the participation of many suppliers in the standardization effort, should provide an opportunity to achieve and/or determine conformance. Note that a suite of programs is currently being developed by groups based in Australia and the U.K. which could form the basis of a conformance test.

5.2 User Operational Considerations:

The expected growth in the use of extensions to Pascal suggests that costs incurred by users due to the timely adoption of an extended standard will be insignificant compared with the Benefits (section 3).

5.3 Legal considerations:

Preserving machine independence and compatibility with any ISO Pascal standard should prevent problems related to restraint of trade and public interest.

5.4 Estimated Costs:

Producers will face conversion costs. Effort will be made to ensure that extensions are efficiently implementable in language processors and may be used efficiently on existing hardware.

6. MAINTENANCE

6.1 Extent and Frequency of Anticipated Changes:

X3J9 intends to provide interpretation and clarifications of the eventual ANS Extended Pascal as the need arises.

X3J9 also intends to comply with the requirement that an ANSI standard be reviewed within a five year period.

6.2 Resources:

X3J9 accepts its responsibility to maintain the eventual standard and to continue this activity along with any revision efforts.

6.3 Cost:

The cost of maintaining the standard on an annual basis is estimated to be comparable to the original development cost.

7. CLOSELY RELATED STANDARDS ACTIVITIES

Related standardization efforts include:

(a) the development of an ANS Pascal by X3J9 as per X3J9/79-026 (proposed),
(b) the development (jointly with X3J9) of a proposed IEEE Standard for Pascal (IEEE Project P770), and
(c) the associated ISO standardization of Pascal.

These efforts have a different objective and a different time frame than the herein proposed effort, and thus should be carried to completion as planned.

8. RECOMMENDED TIME FRAME

June 30, 1981	-- End of public proposal initiation
December 30, 1981	-- Processing of proposals complete
June 30, 1982	-- Draft of proposed Extended Pascal document complete
December 30, 1982	-- End of public comment
June 30, 1983	-- Submission of proposed Extended Pascal Document for ANSI/IEEE/ISO consideration.

ANSI X3J9 Meeting of December 19, 1978

by Richard J. Cichelli

Most of the results presented here have been reported in the trade press. Behind the stuffy formality of the official news releases there is an undersurrent of the personalities and politics. And it's for big stakes. Pascal is viewed as a threat to the established order in computing.

The following report by John Knight of NASA and ACM's SIGPLAN gives most of the details.

> The X3J9 committee has been set up by ANSI to establish a standard for the programming language PASCAL. The first meeting was held on 19 December 1978 at the offices of the Computer and Business Equipment Manufacturers Association (CBEMA) in Washington D.C. This association will provide organisational and secretarial support for X3J9 but no technical or managerial support.
>
> To obtain membership of X3J9 it is necessary to apply in writing to the membership secretary at CBEMA. A Member is required to attend at least two out of three meetings and respond to at least every other letter ballot. There must be at least one and at most six meetings per year. The committee must prepare an SD3 document which is its justification for existence to ANSI.
>
> The convenor of this meeting was Justin Walker. Normally ANSI organises language specific subcommittees based on industrial and academic demand from inside the U.S.A. In this case X3J9 was established because of a request for support from the International Standards Organisation (ISO).
>
> It seems that none of the attendees of this meeting had applied for membership of X3J9 in writing as required so technically all attendees were observers. Thus this meeting was in a sense informal. ANSI requires a committee to elect a chairperson and secretary from within its membership. No chairperson was available because none of the participants were formal members of X3J9. The meeting was conducted by the convenor.
>
> The first surprise which occurred was an announcement by a representative of the IEEE that the IEEE had established its own PASCAL standards committee with the goal of producing a standard for the language. This announcement met with a lot of comment and considerable disapproval. The theme of the disapproval was that it is ANSI's job to establish standards and this would be a duplication of effort. Despite these comments, it is clear that the IEEE will continue its effort.
>
> Following the debate over the IEEE announcement, the discussion turned to organisational matters of X3J9. It was explained that four officials are required. They are:
>
> (1) Chairperson
> (2) Vice Chairperson
> (3) Recording Secretary
> (4) International Liason Officer
>
> The reason for the relatively high level of activity at the ISO is the current work being done by the British Standards Institute (BSI). The BSI has prepared a draft PASCAL standard and will submit it to the ISO. There is a high probability that it will be accepted (after revision) by the BSI and ISO. A move was made

at the X3J9 meeting to accept this draft standard as an ANSI draft standard. This was rejected on the grounds that few people had seen it. The meeting agreed to consider it at a later date after it had been circulated. The BSI document has been published by the PASCAL Users Group as PASCAL Newsletter no. 14. One point which generated a lot of debate and few conclusions is that the ISO has stated that its PASCAL effort will not involve any development of the language. ANSI has adopted the view that this is not necessarily its policy.

The next meeting of X3J9 will be hosted by UNIVAC in Irvine, California and will be held February 20 - 22. The proposed agenda is:

(1) Nomination of committee officials.
(2) Preparation of the SD3 document.
(3) Establishment of a review process.
(4) Review of written comment on the BSI/ISO document.
(5) Submission of proposals to the BSI and the ISO via the International Liason Officer.
(6) Action items.
(7) Report on ISO standard situation.
(8) Future meetings schedule.

- - - - - -

Some further clarification of the SIGPLAN's stand on the issues can be gained from Paul Abrahams' message to the SIGPLAN membership.

From the Vice-Chairman of SIGPLAN to SIGPLAN Members

I wòuld like to report to you on the recent upsurge of standardization activity with respect to Pascal, since I know that Pascal is a language that many of you are interested in. I am grateful to John Knight, our semi-official representative to committee X3J9, for providing me with the input for this report.

There àre three different groups currently interested in developing a PASCAL standard: the American National Standards Institute (ANSI), the IEEE, and the International Standards Organisation (ISO). A draft standard has been submitted to ISO by the British Standards Institute (BSI) (forgive the alphabet soup), and Niklaus Wirth, the author of Pascal, has expressed his wholehearted support of this draft. The BSI draft is likely to serve as an initial version for all the standardization efforts.
Meanwhile, back at the ranch, ANSI has established Technical Committee X3J9 on Pascal, and the committee will serve as technical advisory group to its ISO counterpart. Thus the ISO and ANSI standards will probably be developed in coordination with each other. X3J9 has already met once as of this writing, and its second meeting was scheduled for February 20-22. The first meeting had 70 potential members in attendance--surely a strong indication of interest. The IEEE Pascal Standards Committee has been established under the chairmanship of Bruce Ravenal, and its first meeting took place on January 29. No details about this meeting are available as of this writing.

It is probably not in anyone's interest to have three incompatible Pascal standards, and so the pressures for consolidation of the different efforts are likely to be strong. However, there are both technical and political obstacles to be oversome. The primary technical issue is whether the standard should involve any new development of the language. ISO's opinion is that it should not; ANSI wants to keep its options open; and IEEE has yet to express an opinion. The political issue is whether the IEEE and ANSI efforts can be merged; cooperation with ISO (at least from ANSI's viewpoint) is not at issue.

I suggest that any of you who would like more information on this subject contact John Knight (804) 827-3875/3026. In addition to being SIGPLAN's representative, he has a strong personal interest in Pascal and in the effort to standardize it.

- - - - - -

But it's not over yet!

On that fateful December 19 three more meetings occurred which I attended. There was the Linda Hecht/IEEE meeting, the combined dinner meeting and the ANSI organizers' after dinner meeting.

Try to appreciate the politics of the situation. The ANSI X3 committee's secretariate is CBEMA. X3 uses CBEMA facilities and personnel. CBEMA looks to many like an East coast mainframe manufacturers clique. Power in this clique is related to market dominance.

When X3 met to consider the PUG sponsered BSI/ISO activities, according to J.A.N. Lee who is ACM's representative on X3, the vote was taken to start a divergent competitive standards activity. This was done by deleting the "no language development" clause from the ISO work order. With this deletion a number of X3 members voted against starting X3J9. It is not a usual X3 policy to institute such a committee. Normally a committee of this sort approaches ANSI for recognition. As Lee reports it, this action was a direct rebuff to PUG and BSI.

How did the IEEE get involved? Believe it or not, the IEEE actually did some standardization on a numerical control "language", so there is a precedent for their activities. Most ACM affiliates regard this somewhat tenuous precedent as specious. However, if you consider that the IEEE is the professional home of many of those affiliated with West coast semi-conductor manufacturers and their kindred software technologists...

It's not hard to realize that the existing Pascal software support systems could help bridge the software gap between what established vendors provide and what the West coast upstarts need in order to sell their iron. It wouldn't hurt to tap the Pascal user community for customers as well.

As soon as X3J9 adjourned, Linda Hecht, the IEEE representative, invited me, Jim Miner (Univ. of Minnesota), Scott Jameson (H-P), Rick Shaw (SEL), Bruce Ravenel (Language Resources), and Gabe Moretti (Signetics) to a pre-arranged meeting place in Washington. Linda explained the advantages of an IEEE Pascal standard - namely, speed. There were only two problems. 1) ANSI and 2) such an IEEE committee gets carte blanche. We PUG members had some reservations about giving the language over to a committee one potential member of which asserted that he wanted to "fix Pascal so it would work for the engineer at his test bench." Linda's attitude was interesting: "Do it with us or we will do it without you." After I promised to solicit direct PUG membership response to the IEEE board of directors about this approach, she modified her position and we established Bruce Ravenel as liaison between IEEE and PUG.

While Hecht, Ravenel and Company are proposing a six month standards activity, DEC's representative at X3J9 is talking about a five year ANSI effort to fix Pascal for us.

The Pragmatics!

Pursuing the typical ANSI programming language standards activity over the usual five to seven years can cost a company or individual upwards of $30,000.

Some control of ANSI X3J9's activities can be had by using their constitution and bylaws. Duplication of work and production of conflicting standards is expressly forbidden. Consensus of all major interest groups is required. If PUG isn't a "major interest group" concerned with Pascal, I don't know what is. I believe the PUG membership at large should advise and consent to the standard. I have represented and defended this viewpoint at all meetings that I have attended. Incidentally the ANSI charter is designed to provide committees which formalize and recognize existing practice not formulate new designs.

After the IEEE meeting on the 19th, another meeting took place over dinner. Those from that meeting were joined by Justin Walker (NBS), Barry Smith (OMSI), Bill Price (Tektronix) and a few others. Confusion about the day's events reigned. Then, like a light breaking through the darkness, someone suggested that Ruth Richert (Burroughs) be made X3J9's chairperson. Brilliant! The idea and Ruth both! I was given the job of calling her and asking if she would accept such a responsibility. (She wasn't present at the X3J9 meeting.) I called her directly from the restaurant. She agreed provided her management approved. Ruth has coordinated similar activities within Burroughs and has a track record for success that is legendary. (Incidentally, it was Ruth who affectionatel awarded me the "order of the claw" - see PN #13 cover - at the UCSD workshop.)

The final meeting of the evening was with Justin Walker, Bruce Price, Barry Smith, and about half a dozen others. Those of us who were particularly disturbed by X3J9's failure to elect a chairperson (as required by Robert's Rules of Order which govern ANSI meetings) explained to Justin that the lack of a chairman allowed self appointed officials present at the speakers platform all through the meeting to effectively prevent the group from voting to restrict the standards committee work to reviewing, clarifying and formalizing the de facto standard. Justin felt overwhelmed by the events of that afternoon and felt someone with Ruth's organizational skill would better guide the X3J9 work.

No matter what happens, PUG is likely to have the final say on Pascal standards. I believe the important thing is to get the de facto core standard through ISO as soon as possible.

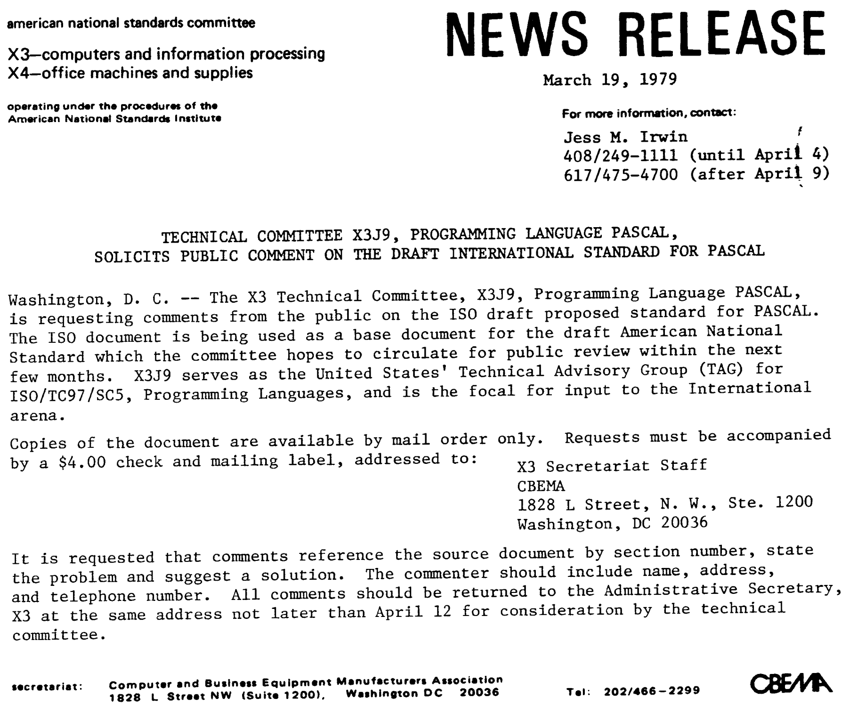
*****

Niklaus Wirth in a letter to me dated 8 December and received 12 December, stated:

"I have now also received a copy of Tony Addyman's proposal for an ISO standard, and I am impressed by the care and attention to details of this report. There is not much doubt that ISO will finally adopt it (or a later revision of it), and I therefore consider this document as of great significance. ..."

"...I wholeheartedly support the ISO draft, and perhaps you should exert your influence on implementors to at least follow that report. ..."

- Andy Mickel 78/12/13.



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

March 19, 1979

For more information, contact:

Jess M. Irwin
408/249-1111 (until April 4)
617/475-4700 (after April 9)

TECHNICAL COMMITTEE X3J9, PROGRAMMING LANGUAGE PASCAL,
SOLICITS PUBLIC COMMENT ON THE DRAFT INTERNATIONAL STANDARD FOR PASCAL

Washington, D. C. -- The X3 Technical Committee, X3J9, Programming Language PASCAL, is requesting comments from the public on the ISO draft proposed standard for PASCAL. The ISO document is being used as a base document for the draft American National Standard which the committee hopes to circulate for public review within the next few months. X3J9 serves as the United States' Technical Advisory Group (TAG) for ISO/TC97/SC5, Programming Languages, and is the focal for input to the International arena.

Copies of the document are available by mail order only. Requests must be accompanied by a $4.00 check and mailing label, addressed to:

X3 Secretariat Staff
CBEMA
1828 L Street, N. W., Ste. 1200
Washington, DC 20036

It is requested that comments reference the source document by section number, state the problem and suggest a solution. The commenter should include name, address, and telephone number. All comments should be returned to the Administrative Secretary, X3 at the same address not later than April 12 for consideration by the technical committee.

secretariat: Computer and Business Equipment Manufacturers Association
1828 L Street NW (Suite 1200), Washington DC 20036 Tel: 202/466-2299 CBEMA

*****

A Few Experiences at the Boulder Joint Pascal Committee Meeting 1979 April 26 & 27.

The main purpose of the Boulder meeting was to convene the TRTG chaired by Bill Price in order to produce an official American response to the BSI/ISO document N462. At the time the general feeling was that the Boulder meeting was a success although final agreement on the response by the whole JPC was delayed. In retrospect, the Boulder meeting was the most productive of the American standards effort. I was really impressed with the general quality of the technical discussion by most voting members at the meeting whereas my preconceptions were quite skeptical. The population of frustrated language designers which usually plague standards committees and which get their chance to ruin a language was fortunately small.

Also apparent was the positive influence of JPC co-chair Bruce Ravenel from the IEEE P770 Pascal Committee. The site of the meeting was the Computing Center at the University of Colorado, and Bruce naturally provided a historical continuity because he "cut his Pascal teeth" at the same university. One should not underestimate the significance of the joint standards effort (IEEE and ANSI) without which a protracted standards process would have been a certainty.

Last but not least, the meetings were principally chaired by the very able and jovial Marius Troost. I feel that the group benefitted greatly from Marius's experience and judgment, and we were indeed fortunate to have his services. Marius congratulated Bill Price for his hard work with TRTG.

Hey! Guess what I learned at Boulder? That there are people who work for computer companies whose sole job is to represent that company on standards committees. In other words, these people may know nothing about Pascal at all--never have written a program--and still they are there with considerable weight. Imagine my amusement when the DEC representative kept referring to the meeting as "X3J3" (the name of the ANSI FORTRAN committee). You could sure tell where she had been spending the last few years!

Reflections

I'd like to share some other information I've learned about the USA standards process in general. Actually I'm not even sure I have it all straight myself!

First of all, terminology and basic procedures are confusing. ANSI is a non-profit, private (non-governmental) body whose purpose is to aid standards development of all kinds. The ANSI committee in charge of the area of Computers and Data Processing is called X3. A look at the standing membership of X3 shows a predominance of computer manufacturers and large businesses--not ordinary users. Additionally there is NBS (the National Bureau of Standards), a governmental agency within the U.S. Department of Commerce which is completely separate from ANSI, and it or another agency handle Federal Standards for computing such as those which exist for COBOL and FORTRAN.

One strange term you hear is "secretariat." The duty of carrying on the communications, document-copying and distribution, and scheduling of meetings, etc. for each standards committee is performed by the secretariat. The member of X3 which happens to perform the secretariat of X3 is CBEMA: The Computer Business Equipment Manufacturers Association. As the name implies, you know who controls this group! And guess who is the secretariat of ISO? ANSI!

Suppose we (PUG) had decided to get an official Pascal standard adopted by ANSI. Roughly, the correct procedure is to make an application to X3's SPARC (Standards Planning and Requirements committee) to get them to consider forming a committee to consider creating a standards committee! This can take about a year if you are successful.

Now the conventional view of some people in the US (and indeed some PUG members) was that we should have of course approached ANSI for a standards effort, because it has undertaken standards efforts for other programming languages and this represents a kind of precedent.

This line of thought totally ignored the fact that other language standards efforts undertaken by ANSI have produced unsatisfactory results: in other words bad precedents! Look at the size and complexity of COBOL produced by X3J4; the original designers of BASIC are still crying in their sleep over the work of X3J2; and I won't waste any more words about FORTRAN and X3J3 (see David Barron's editorial on page 3 of PN #13).

These were all committee efforts dominated by representatives of the large computer manufacturers and the US government and took many, many years. Why did we have to make these mistakes?

Fortunately we didn't. Although there was an attempted move at the first X3J9 meeting in Washington to not even consider the work on a Pascal Standard already done by PUG and BSI and to undertake an effort from scratch, it was fortunately defeated. It was also simply amazing that so many of the attendees of this meeting were not even PUG members! We may be only lucky that the real reason we were able to defeat such a chauvinistic American move (in the face of a cooperative international initiative) was that we users were organized through PUG and informed through Pascal News.

So everything has turned out fine so far and people ask me why I was so worried and sure that things would go wrong. Well, there was a lot at stake: there were no guarantees about avoiding a long, misguided effort directed by the manufacturers instead of the users, and we knew that the international effort was already underway. My hope was expressed in a letter to SPARC on page 86 of PN #13: ANSI had an opportunity to reciprocate its respect with ISO--several ISO standards are one line saying "see ANSI standard xxx" and for Pascal, a language with European origins, the standardization whould be left to Europeans.

Before the December X3J9 meeting in Washington, the BSI/ISO proposal caught X3 off guard and several SPARC steps were skipped over and X3J9 was immediately set up and then this first meeting was set (wasn't that easier than the regular procedure?). I was still personally very angry that only afterwards did the secretariat inform PUG. Why didn't they check with us for information? No matter that PUG already existed and represented the majority of Pascal users! Anyway, at the December meeting, Justin Walker of NBS chaired X3J9 temporarily and several committees were set up: one produced the SD-3 reproduced above--a document outlining the goals of X3J9 similar to documents existing for the BSI and ISO Pascal initiatives.

Jess Irwin was selected by the group as secretary, who has the important task of indexing, reproducing, and distributing documents. These documents range from announcements (and pronouncements) from X3 to papers discussing technical issues. So far the Joint Pascal Committee has over 200 documents, and even the document register (index) itself is a numbered document!

The people attending the Washington meeting with the intention of representing PUG were Jim Miner, Rich Cichelli, and Rick Shaw. Because Rich and Rick wanted to also represent their organizations (ANPA/RI and SEL respectively), they weren't allowed to do this. Thus Jim became PUG's representative and I became his alternate.

Fortunately the standards activity is a public process, but unfortunately the resources required by the attendees are immense in order to pay for the time, lodging, and travel expenses. This greatly favors individuals representing big corporations with expense accounts (tax deductible, no doubt). In fact the longer the computer manufacturers can drag out the standards proceedings, the more power their representatives have toward the end of the process because they will be practically the only ones there! So standards activities, supposedly in the best interests of the users, effectively exclude user participation!

Jim Miner, in fact, has gone to 2 meetings on his own money, and we both went to the Boulder meeting on our own money. Finally NBS is helping Jim pay for plane fares to upcoming meetings.

- Andy Mickel 79/08/31.

☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆

Did you know that pascal has already been standardized?
One ISO SI Pascal is a newton/m^2

Validation Suite

The University of Tasmania

Postal Address: Box 252C, G.P.O., Hobart, Tasmania, Australia 7001

Telephone: 23 0561. Cables 'Tasuni' Telex: 58150 UNTAS

Dear Pascal User,

In the past you have asked about the availability of a Pascal validation suite of programs, or I have reason to suspect that you are interested in this topic.

I enclose therefore a copy of a press release concerning Release 2.0 of this package (the first public one) as at 13th July 1979. Should you wish to receive a copy of the Validation Suite, contact your nearest distributor. Only handling charges will be levied to cover the average cost of a magnetic tape, postage, and follow-up information.

Any comments on the package and its use will be welcomed, though as I anticipate a number of letters, I may not be able to acknowledge each one personally.

Distribution Centres

In the USA and the Americas:
Richard J Cichelli — Phone (215) 253-6155
ANPA/RI
P.O.Box 598 — Fee US $ 50.00
Easton, Pa. 18042
USA

In Europe:
Brian Wichmann — Phone (01) 977-3222
National Physical Laboratory
Teddington, Middlesex — Fee not known
England TW11 OLW
United Kingdom

In Australia, New Zealand and Japan:
Pascal Support — Phone (002) 23-0561 X435
Department of Information Science
University of Tasmania — Fee Aus $ 50.00
Box 252C G.P.O.
Hobart, Tasmania 7001
Australia

Other places:
Choose the nearest distributor.

Addresses for suggestions or complaints:

Sept 1979 .. Feb 1980	March 1980 on
Prof A.H.J.Sale c/o Computer Studies Group The University Southampton England SO9 5NH United Kingdom	Prof A.H.J.Sale Department of Information Science University of Tasmania Box 252C G.P.O. Hobart, Tasmania 7001 Australia

The distribution format convenient to each distributor varies, so please enquire before sending money.

Yours sincerely,
Arthur Sale

Arthur Sale

PRESS RELEASE

PASCAL VALIDATION SUITE AVAILABLE

Pascal has joined the select group of languages, which include COBOL, which have a validation set of programs to check that compilers and machines conform to the requirements of the Standard. Released on Friday 13th July by Arthur Sale at the University of Tasmania, the validation suite is expected to find wide use almost immediately. Many machine suppliers and software houses have been waiting for its release in order to assist them in developing compilers for Pascal that will be acceptably correct.

The present release, numbered 2.0 as there was a previous unreleased version, contains 283 separate programs. About 150 of these are tests to check that compilers and machines conform to the requirements of the Pascal Standard, and about another 70 check that the system does not deviate outside its requirements. The remainder explore the requirements of the Standard in areas defined to be errors or implementation-defined, or attempt to assess the quality of the compiler in various areas.

Release tapes can be obtained from a number of distribution centres around the world, for basically handling charges. Further information is obtainable from the Department of Information Science, University of Tasmania, Box 252C G.P.O., Hobart, Tasmania 7001.

The validation suite was developed by Brian Wichmann in the U.K. and Arthur Sale in Tasmania under the auspices of the Pascal Users Group. The intention of the package is to encourage a very high degree of portability of Pascal programs (even higher than presently exists), and to provide users with a mechanism to assure themselves that vendors' products comply with the Standard. It is expected that validation reports on compilers will shortly be published in Pascal News: three are already complete. Such reports will encourage suppliers to enhance the quality of their products.

The announcement again highlights the rapid development of Pascal as a serious programming language for use in the computing marketplace, and not simply another academic toy.

COMING SOON

Validation Suite

Pascal News #16

Implementation Notes

Portable Pascals

PASCAL-P

Pascal-P ordering information has changed. In North and South America, order from:

William Waite
Software Engineering Group
Electrical Engineering Department
University of Colorado
Boulder, Colorado 80309
Phone (303) 492-7204

In Australia, order from:

Tony Gerber
Basser Department of Computer Science
University of Sydney
Sydney, New South Wales 2006
Australia
Phone 61-02-692-3756 (Gerber), 61-02-692-2541 (Dept Sec)

Tony reports that his Pascal-P distribution costs are now A$20 for an unconfigured tape and A$40 for a configured tape. Of course Chris Jacobi is still distributing Pascal-P in Europe, Africa, and Asia from ETH, Zurich.

Arthur Sale reports that he may embark on producing a Pascal P5 which will implement the forthcoming ISO Standard Pascal, when he knows what it is.

{For those that don't know, Pascal-P is the parent of many of the present crop of Pascal compilers - not very useful by itself but modifiable to other target machines by supplying a changed code-generator. The bugs in Pascal-P are very widely distributed! }

PASCAL-E

A new portable Pascal compiler has been under development for some time at Vrije University in Amsterdam by Andrew Tanenbaum and his co-workers. This compiler was initially derived from Pascal-P2 and generates an intermediate code called EM-1. EM-1 (for Experimental Machine) is an optimal stack machine architecture for stack languages such as Pascal.

The PDP-11 implementation of Pascal-E comes with an EM-1 code optimizer which produces a final compiler in only 20k bytes. This compiler has been covered in Pascal News #11 p87 under DEC PDP-11. The system runs under UNIX and Andrew Tanenbaum described the system at the UNIX Conference in Toronto in June.

His address is: Computer Science Group, Vrije University, De Boelelaan 1081, 1007 MC, Amsterdam, The Netherlands (020-5482410).

Pascal Variants

TINY PASCAL

Supersoft { What does that make you think of? } have announced a Tiny Pascal fpr TRS-80 and North Star. It is supposed to run at least 4 times faster than Basic and requires a Level II TRS-80 with 16k and a 24k North Star. Tiny Pascal is { of course } a subset of Pascal, and apparently includes:

"recursive procedures/functions, if-then-else, repeat/until,
peek and poke, while, case, & more"

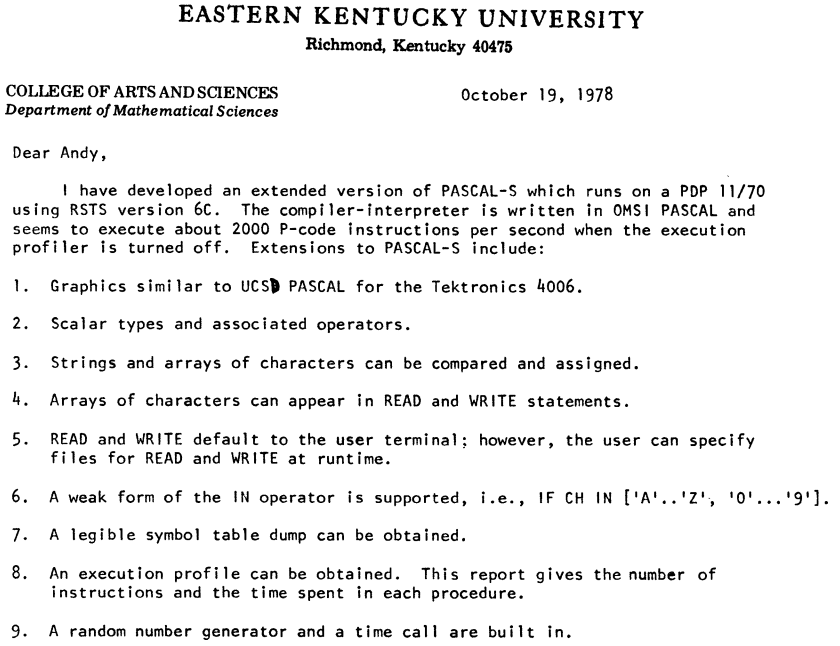
Cost: $40, from

Supersoft
P.O.Box 1628
Champaign, IL 61820
(217) 344-7596

{ Lie back, relax, and let Supersoft Pascal take care of your troubles. PUG makes a free gift of the above slogan. }

PASCAL-S AND PASCAL-I

We have some new information on an implementation of Pascal-S for the PDP-11 presented below. Rich Cichelli sent an update for Pascal-I (see article in this issue), the very successful implementation of Pascal-S designed for highly interactive use. Note that we put Rich's previous checklist under CDC 6000 in Pascal News #11 p82.



EASTERN KENTUCKY UNIVERSITY
Richmond, Kentucky 40475

COLLEGE OF ARTS AND SCIENCES
Department of Mathematical Sciences

October 19, 1978

Dear Andy,

I have developed an extended version of PASCAL-S which runs on a PDP 11/70 using RSTS version 6C. The compiler-interpreter is written in OMSI PASCAL and seems to execute about 2000 P-code instructions per second when the execution profiler is turned off. Extensions to PASCAL-S include:

1. Graphics similar to UCSD PASCAL for the Tektronics 4006.
2. Scalar types and associated operators.
3. Strings and arrays of characters can be compared and assigned.
4. Arrays of characters can appear in READ and WRITE statements.
5. READ and WRITE default to the user terminal; however, the user can specify files for READ and WRITE at runtime.
6. A weak form of the IN operator is supported, i.e., IF CH IN ['A'..'Z', '0'...'9'].
7. A legible symbol table dump can be obtained.
8. An execution profile can be obtained. This report gives the number of instructions and the time spent in each procedure.
9. A random number generator and a time call are built in.

10. All programs are given a DAY, DATE, and TIME stamp.

Current symbol table size is 120; code vector size is 1000, and the runtime stack size is 1500; consequently, the system's primary use is educational.

The code section compiles into a little over 16K words with the syntax analyzer and interpreter overlaying each other. This leaves about 12K words for variable storage and 10 Buffers.

Extensions 1 and 2 are essentially due to Don Baccus of OMSI; however, the bizarre way our system handles control characters and carriage returns necessitated extensive reworking of the graphics system. Extension 8 was adapted from Matwin and Missala (PUG #12).

I would like to correspond with and/or trade implementation details with the other PASCAL and PASCAL-S users. Enclosed is a sample program which finds knights tours of a chessboard.

Sincerely yours,

Jerry

Dr. Jerome H. LeVan
Associate Professor of Mathematical Sciences

0. DATE/VERSION: PASCAL-I, 30-MAR-79, Release 2.03

1. IMPLEMENTATOR/DISTRIBUTOR/MAINTAINER:
Richard J. Cichelli, 901 Whittier Drive, Allentown, Pa. 18103
J. Curtis Loughin
John P. McGrath

2. MACHINE: Machine independent. 25 installations on CDC, DEC, IBM, and other computers. Written entirely in PASCAL using some features of PASCAL 6000 (segmented files for terminal I/O to flush buffers and read past EOF on terminal input).

3. SYSTEM CONFIGURATION: Developed under SCOPE 3.4 with INTERCOM using the CDC segmented loaded. Installed on many others.

4. DISTRIBUTION: 600' magnetic tape. SCOPE internal format, 7 track, 800 bpi, or 9 track 800 bpi ASCII or EBCDIC. Pascal-I isn't in the public domain. Price - $100. Make check payable in U.S. dollars drawn on a U.S. Bank to Richard J. Cichelli.

5. DOCUMENTATION:
System Level: Very readable code (guaranteed)
User Level: Machine readable users manual
System explains itself in response to the HELP command (full details - oriented towards novice programmers.)

6. MAINTENANCE: Accepting bug reports.

7. STANDARD: Supports PASCAL-S. Differences from standard PASCAL - files - only INPUT and OUTPUT, no sets, pointer variables, case variants, labels, goto's or with statements. Any PASCAL-S/PASCAL-I program is a valid PASCAL program.

8. MEASUREMENTS: Interpreter and overlayed. The compiler forms the largest overlay segment and runs at 33,000 (octal) words. The editor segment runs in about 24,000 (octal) words. PASCAL-I will compile and interpret PASCAL-S programs of up to about 500 lines as the system is currently configured.

9. RELIABILITY: Runs just great.

Implementation Notes

10. DEVELOPMENT METHOD: Started with PASCAL-S and Wirth-Jensen I/O routines. Built suitable data structures for storage of compressed program source and interpreter code. Modified PCSYSTM to fully recover from user aborts and system timeouts. Also added file access primitives and moved stack and heap to low core to enable the segmented loader to vary field length. The system is about 7500 lines of tightly formatted PASCAL.

Implementor responsibilities:

Curt Loughin - Editor, Formatter, PASCAL-S compiler rewrite, PASCAL-S interpreter rewrite, and Immediate code routines.

John McGrath - I/O routines rewrite, HELP command, PCSYSTM mods.

Richard Cichelli (project leader) - Post mortem dump and other run-time control and status routines.

CONCURRENT PASCAL

Note: We have had no word from Per Brinch-Hansen on the survey of users of Concurrent Pascal promised for this issue. Perhaps in PN #17...

Österreichische Studiengesellschaft für Atomenergie Ges.m.b.H.
Lenaugasse 10 • A-1082 WIEN • Austria

Current State of the RSX11M Implementation of Concurrent Pascal

We have moved P.B. Hansen's Concurrent and Sequential Pascal compilers from the Solo operating system to RSX11M (and RT11) so that we could develop Concurrent and Sequential Pascal programs in a customary timesharing environment.

This was done about 2 years ago.

In the meantime we have developed a new Concurrent Pascal Kernel which differs from the original Kernel in some points.

The main differences are:

- The system can run on all types of PDP11.
- An interactive trace facility can be used to make program flow and process switching visible on a terminal.
- The number of processes is only restricted by the available memory space. Process switching is very fast. A process needs only 9 words system overhead. We had a pilot project using 60 concurrent processes.
- The process scheduling strategy is a simple demand scheduling (no time slicing or "round robin" scheduler)
- The kernel runs as a single task under RSX11M. No memory management directives are used.
- The interface to the operating system is simple. The kernel communicates with RSX11M only via a few QIO/AST statements. At the moment the Concurrent Pascal kernel supports only terminal I/O. Other devices may be connected in the same way.

- At the moment the loading and executing of sequential programs in a Concurrent Pascal program is still not supported.
- Only one process at a time can execute a "WAIT"-instruction.
- A "powerfail restart" facility can be used by a Concurrent Pascal program in the same way as a device. A process performing an I/O operation on the power fail device is suspended until power fail restart occurs.

The trace facility is very useful for demonstration purposes and program testing. The following lines show a sample trace output of P.B. Hansen's "realtime scheduler":

```
   >; USE THE INTERACTIVE TRACE FACILITY
   >;
   >CER SC1
   *** CONCURRENT PASCAL KERNEL START ***
↑T CER>HP 4            - set upper limit of process numbers to be traced
↑T CER>LL 273 HL 282 - set range of line numoers to be traced
↑T CER>EVENT IO OFF
↑T CER>PRINT ON
   EXIT ROUTINE      IN PROCESS 00002. AT LINE 00279.
   EXIT  MONITOR     IN PROCESS 00003. AT LINE 00277.
   EXIT  MONITOR     IN PROCESS 00004. AT LINE 00276.
   EXIT ROUTINE      IN PROCESS 00003. AT LINE 00278.
   EXIT  MONITOR     IN PROCESS 00002. AT LINE 00276.
   EXIT  MONITOR     IN PROCESS 00003. AT LINE 00279.
   EXIT ROUTINE      IN PROCESS 00003. AT LINE 00279.
   EXIT  MONITOR     IN PROCESS 00004. AT LINE 00277.
↑T CER>ENTER EXIT MENTER MEXIT DELAY CONTINUE OFF LINE ON
   NEW LINE          IN PROCESS 00003. AT LINE 00281.
   NEW LINE          IN PROCESS 00004. AT LINE 00278.
   NEW LINE          IN PROCESS 00003. AT LINE 00276.
   NEW LINE          IN PROCESS 00004. AT LINE 00279.
   NEW LINE          IN PROCESS 00004. AT LINE 00280.
   NEW LINE          IN PROCESS 00003. AT LINE 00277.
   NEW LINE          IN PROCESS 00002. AT LINE 00278.
   NEW LINE          IN PROCESS 00004. AT LINE 00281.
↑T CER>LINE OFF DELAY CONTINUE ON
↑T CER>CONTINUE OFF
↑T CER>LP 3 HP 4
↑T CER>LL 0 HL 0
   DELAY             IN PROCESS 00004. AT LINE 00160.
   DELAY             IN PROCESS 00004. AT LINE 00139.
   DELAY             IN PROCESS 00003. AT LINE 00160.
   DELAY             IN PROCESS 00003. AT LINE 00139.
   DELAY             IN PROCESS 00004. AT LINE 00160.
   DELAY             IN PROCESS 00004. AT LINE 00139.
   CER>LP 0 HP 0 CONTINUE ON
   CONTINUE          IN PROCESS 00002. AT LINE 00145.
   ........          IN PROCESS 00003. AT LINE 00139.
   DELAY             IN PROCESS 00002. AT LINE 00160.
   CONTINUE          IN PROCESS 00005. AT LINE 00166.
   ........          IN PROCESS 00005. AT LINE 00324.
   DELAY             IN PROCESS 00002. AT LINE 00139.
   CONTINUE          IN PROCESS 00003. AT LINE 00145.
   ........          IN PROCESS 00004. AT LINE 00139.
   DELAY             IN PROCESS 00003. AT LINE 00160.
↑C
   PROGRAM TERMINATED AT LINE 00277. IN PROCESS 00004.
```

```
PROGRAM HISTORY:
........          IN PROCESS 00003. AT LINE 00139.
........          IN PROCESS 00004. AT LINE 00139.
DELAY             IN PROCESS 00003. AT LINE 00160.
........          IN PROCESS 00004. AT LINE 00139.
DELAY             IN PROCESS 00003. AT LINE 00160.
........          IN PROCESS 00004. AT LINE 00139.
DELAY             IN PROCESS 00003. AT LINE 00160.
........          IN PROCESS 00004. AT LINE 00139.
*** CONCURRENT PASCAL KERNEL END ***
>
>
```

This system has been used successfully in an industrial process control application under RSX11S. It will probably run under IAS and RSX11D, too. The complete software package is available for 5.000,- Austrian Schilling (∼ 350 US$).

The main drawback of the Concurrent Pascal compiler is that it produces relatively slow threaded code (PDP11-Fortran is about 2.5 times faster). To overcome this disadvantage we plan to build a Concurrent Pascal precompiler for the highly efficient OMSI Pascal compiler.

Nevertheless the current system is an excellent programming tool for non time critical or I/O-bounded tasks. Compared to RSX11-realtime-multitask applications the Concurrent Pascal system is many times faster, since task switching and eventflag synchronisation is a very slow process in RSX11.

Yours sincerely,

Dipl.Ing. Konrad Mayer

MODULA

Modula is an experimental attempt to build a real-time programming language with structure. We reproduce the abstract page of the Modula-2 report by Niklaus Wirth, which is an attempt to put Pascal back into Modula. The other abstracts in this section relate to work done by York University on Modula-1, and their implementation. Write to them for copies or distribution tapes.

Modula-2 by N.Wirth
Institut fur Informatik, ETH, CH-8092 Zurich, December 1978.

Abstract

Modula-2 is a general-purpose programming language primarily designed for systems implementation. This report constitutes its definition in a concise, though informal style.

Note: No compiler is available for distribution at this time.

UNIVERSITY OF YORK
HESLINGTON, YORK, YO1 5DD
TELEPHONE 0904 59861

12 January 1979

Dear Mr Mickel

University of York Modula Compiler Second Release

The second release of the Modula (UNIX/PDP-11) compiling system will be made during February 1979. In comparison with the first release the following changes are incorporated in the second release:

* all known compiler errors will be corrected,

* the VALUE clause (for the load-time initialisation of level 0 variables) and the standard functions 'off' and 'among' will be implemented,

* optional run-time checks for CASE expression out of range, array index out of range and a procedure exceeding its stated depth of recursion will be implemented. The recursion depth of procedures inside Device Modules will not be checked,

* the portability/bootstrapping interface between passes 2 and 3 of the compiler will be brought into line with the description in Wand(1978), and

* the set of test programs will be extended and improved.

The only language restriction remaining in this release will be 'declaration before use'.

Users of the first compiler release who received a magnetic tape from York are requested to return the tape for the second release. No charge will be payable for existing users of the compiler who wish to update to the new release. Our charges to new users are 300 pounds to commercial customers and 50 pounds to educational and research institutions not in the United Kingdom.

Suggestions from users (and others) for longer-term enhancements are most welcome. At the present time the following seem the most likely:

* an alternative 'back-end' producing code for one of the new 16-bit microprocessors. This will probably be one of the set [68000, Z8000, 8086],

* a User Guide, and

* facilities for separate compilation.

At present the University of York has no plans to produce versions of the Modula compiling system that run under different PDP-11 operating systems, although it is hoped that versions which run under RSX-11M and RT-11 will be developed by collaboration with other UK Universities.

We would be interested in hearing from any Modula user about their experiences with the language or with the York compiler. Of course we would be delighted to hear from anyone who would like to take delivery of their first Modula compiler!

Yours sincerely

Ian D. Cottam

I C Wand I D Cottam

(* Note: we have reports that Jeff Tobias has modified this compiler to produce code for the Intel 8086. Jeff is at the AAEC Research Establishment, Private Mail Bag, Sutherland 2232 N.S.W. Australia. Also Steve Bruell, Pete Zechmeister, David Boone, and others are working with John Collins at 3M in St. Paul, Minnesota to modify the compiler to produce code for the Motorola 6809. John is at 3M Center, Bldg 235 F247, St. Paul, MN 55101, phone: (612) 736-0778. *)

Reference

I C Wand, 'MCODE: A description of the bootstrapping interface of the University of York Modula compiler', Report Number 14, Department of Computer Science, University of York (1978)

ABSTRACT OF "MCODE"

by Ian Cottam, Dept of Computer Science, University of York, Heslington, York YO1 5DD, England. Phone (0904) 59861.

"The front-end of the York Modula compiler is a two-pass compiler that translates Modula (Wirth 1977) source programs into an object program for a hypothetical target processor. In this document we will call this object code MCODE and the hypothetical processor, the MMACHINE. The architecture of the MMACHINE has been designed so that MCODE can be mapped without undue difficulty onto existing mini and microcomputer hardware.

It should be emphasized that the MMACHINE is only suitable for the realization of Modula programs and that it contains many primitives, eg DOIO, which directly reflect the operations required in a Modula run-time environment."

{ We apologize for the capitalization in the above abstract, but the introduction was written that way. }

Holden, J. and Wand I.C., *An assessment of Modula*, York Computer Science Report No 16, November 1978, 41 pages.

Abstract:

Wirth has recently published a new programming language called Modula which he suggests is suitable for the programming of process control systems, computerised laboratory equipment and input/output device drivers. The authors have written a compiler for Modula running on a PDP-11 and generating object code for the same machine. Their experience in writing device drivers for a number of PDP-11 devices is reported, including simple mains frequency clocks, disks, CAMAC and a graphics processor. Some difficulties arose during the writing of these programs; these are investigated and solutions proposed, either within the existing language or by minor modifications to the language. The study shows the extent to which Modula meets the requirements for a general purpose real-time/systems implementations programming language; areas of deficiency are noted.

Cottam, I.D., *Functional specification of the Modula Compiler*, York Computer Science Report No 20, March 1979, 69 pages. (Release 2 for PDP-11/UNIX systems)

Abstract:

This document is the functional specification of the University of York Release 2 PDP-11 MODULA compiler. It is assumed that the reader is familiar with the defining document for the programming language MODULA:

"N.Wirth; MODULA, A language for modular multiprogramming. Software - Practice and Experience 7 No.1, 3, (1977)"

York MODULA conforms closely to standard MODULA as defined in [1]. Differences between the two versions are detailed in Section 3. As well as being the specification against which the compiler is written and tested [5], this document serves as a programmer's reference manual.

The York MODULA compiler operates under the control of the UNIX operating system and in conjunction with the standard UNIX PDP-11 assembly language processor "as".

Rumours Department

Kees Smedema in North American Philips is believed to be working on a Modula compiler for the LSI-11 written in Pascal. Kees's address is Philips, 345 Scarborough Rd, Briarcliff Manor, NY 10510 (Phone 914-762-0300).

Wendy DuBois, Zilog Corporation, 10460 Bubb Rd, Cupertino, CA 95014 (408-446-4666) has not kept us informed about the York Modula written in C at Zilog.

Modula for Z-80: Gerd Blanke, Postbox 5107, D-6236, Eschborn, Germany, may have a system for Zilog MCS with 64k under RIO. Phone (06198) 32448.

PASCAL-PLUS

A new entry. Pascal-Plus is a set of extensions to Pascal making up an experimental language which provides concurrency and modularity. We reproduce the abstract of a report received on Pascal-Plus. A working compiler for ICL 1900 computers is available from Belfast (address below), and we understand that a Pascal-Plus-P is in preparation.

Hardware Notes

A new section; devoted to retailing gossip and news of Pascal's influence on new hardware. Marginally relevant is the discovery of an instruction in the DEC VAX 11/780 which MUST have been influenced by Pascal. It is even called the CASE instruction. How's that, Tony Hoare, even an instruction named after your invention!

UDS-470

A new microcomputer is being marketed by Control Systems Inc, 1317 Central, Kansas City, Kansas 66102 (931-371-6136), also Minneapolis & Williamsburg. This is a microcomputer development system offering UCSD Pascal(TM), but with special features for putting the developed code into ROM/PROM. Designed for fast development of prototypes, one-off systems, etc, in industrial environments.

Western Digital MicroEngine

Probably everyone has heard of the Western Digital chip set which implements a 16-bit microcomputer based on the highly modified version of P-code generated by Ken Bowles' compilers. Naturally it runs a lot faster than an interpreter, and provides super speed when it works (and if you can get one). The race is now on between Western Digital's direct frontal attack on the speed issue in microcomputers, their competitors heading in the same direction, and the highly optimizing compilers generating native code for the older micros and their strange architectures. Watch this with interest, it should be fun. So, Pascal, cut another notch in your belt: even specially designed computers have come so you're right up there with Algol 60 (the Burroughs large machine range) and Fortran (the Control Data crunchers).

S-100 Bus

Digicomp Research Corp., Ithaca, N.Y., have developed a processor board which incorporates the WD MicroEngine(TM) and which plugs into an S-100 bus. The board is said to run at least 2 times faster than the interpreter system on a PDP-11/34, and complies with the IEEE S-100 Standard. Price: around $995.

Pascal/8002

A Pascal/8002 Universal Program Development Package has been designed for use with Tektronix's 8002 Microprocessor Development Laboratory. It provides editor, compiler, assembler, linker, etc. Contact Pascal Development Co, Suite 205, 10381 S DeAnza Blvd, Cupertino, California 95014, with your ready $2000.

National Semiconductor

We are watching with interest National's efforts to support Pascal on a micro chip set (based on their 16-bit 2903A and 2910A microprocessors)
better than their competitors. It is certain that most of the current micro architectures are unsuitable for any software, so it is not hard to do better. But wouldn't it be nice to have a computer architecture which was as elegant as Pascal?

Feature Implementation Notes

James B Saxe and Andy Hisgen
c/o Pascal User's Group
University Computer Center
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455

Montréal, March 26, 1979

Dear James and Andy,

I read with great satisfaction your paper in PN #13 describing "Lasy Evaluatin of the File Buffer for Interactive I/O". I arrived exactly to the same solution when making an ASCII version of Pascal 6000 compiler for CDC Cyber 173 at Université de Montréal in April 1976. I used it with real pleasure and without problem since that time.

I hope this solution be widely accepted and I suggest Pascal standard stick to it (cf PN #14).

Serge Froment
Université de Montréal
Projet C.A.F.E.
Case Postale 6211, succursale "A"
Montréal (Québec) H3C 3Y9

R. K. Ridall & Co. Inc. ≈ 620 Tanglewood Lane, Devon, Pennsylvania 19333 ≈ (215) 647-4212

1979 January 26

Dear Andy:

We have been using the University of Lancaster's P4 Pascal for the Data General NOVA series computers for some time now. It Is quite good for its purpose -- teaching programming. What is so tantalizing about this system is that it is almost complete enough for writing sophisticated applications, but not quite. I offer the following "wish list" as a guide to Pascal implementors:

1: Full ASCII character set, especially lower case.
2: Sets of 128 members, to accommodate SET OF CHAR.
3: Date and time of day routines, for labelling reports.
4: Elapsed time function, so that one could use the instrumentation program AUGMENT in Pascal News #12.
5: Real numbers of 12..16 significant digits (in addition to ordinary real, not instead).
6: Full output formatting of real numbers (of the form WRITE(X:10:2) as in standard Pascal).
7: Random access files with records from 16..512 bytes in length, not just two fixed sizes. The record size should be deduced from the RECORD type declaration.

Pete Goodeve's assembly language interface makes it possible to do 3 and 4, but it would be much more convenient to have these "built in" to the compiler.

Yours truly,

William G. Hutchison, Jr.
Consultant

March 28, 1979

John Fluke Mfg. Co., Inc. / PO Box 43210 / Mountlake Terrace WA 98043 / (206) 774 2211

To: All Pascal Implementors

Having used many different Pascals on different machines, and having had the opportunity to study some forthcoming and as yet unannounced compilers, I notice a disturbing trend in some of the more recent implementations: that of embedding program semantics in the compiler directives to increase the "power" of the language and to compensate for laziness on the part of the implementors.

My suggestion: a compiler directive is acceptable as long as it does not affect the semantics of a program. A program should run correctly independently of directives. This means the following are acceptable:

a. Listing Control (including titling, underlining of keywords, prettyprinting, the printing of warnings).
b. Optimization Control (as long as the optimizations will not affect the correctness of the program).
c. Acceptance or rejection of language extensions.

The following are definitely not acceptable because they hinder transportability and are often implemented because of sheer laziness on the part of the implementor.

a. Options changing the meaning of functions or operations (e.g. turning i/o checks on and off) that a programmer could use to affect the correctness of program execution. Even if a programmer utters the names of seven demons in the right order, he should not be given a "window to hell" or other access to magical powers.

b. Selective Compilation (I could really take off here). Selective compilation is used where it is known at compile time that certain code is not needed. I assert that the following examples show how this may be done in an alternative way if the compilers are a little more intelligent:

```
const debugversion = false;
...
if debugversion then writeln( output, '...' );
{an intelligent compiler can eliminate the above}

const outputformatversion = 3;
...
case outputformatversion of
...
end; {case}
{an intelligent compiler can select the right
   alternative and compile it in-line}
```

It's not as if this is particularly difficult: at least one existing compiler can incorporate the above with a minimal additional effort. Another compiler that is under implementation incorporates a complicated meta-language embedded in the comments; if that were eliminated and the above implemented (the implementors say there will be extensive optimization too...), the compiler would be so much simpler and better.

The dinosaurs are extinct (well, almost. There is still PL/1.) so let's keep it that way.

K. S. Bhaskar

K.S. Bhaskar
Engineering Systems
Programmer / Analyst

IMPLEMENTATION FEATURE NOTE

PROBLEM

The user of Pascal is entitled to rely on the features of the language being correctly implemented, however difficult this may be. The abstraction takes precedence over implementation convenience.

In one problem I have observed, the for-loop fails to carry out the expected action if the second limit expression evaluates to maxint and the statement has the to form. (In some processors the downto form will similarly fail if the second expression evaluates to -maxint.) For example, the statement:

```
for i := (maxint-2) to maxint do writeln(i);
```

has been known to print

```
 32765
 32766
 32767
-32768
-32767
 .....
```

and so on. This is of course entirely erroneous behaviour and should not be tolerated. The problem is, of course, that the value of the for-control-variable has overflowed the integer representation, and in the case cited the overflow is simply ignored.

If the overflow causes a program abort, the user might be slightly more satisfied at knowing of the implementation deficiency, but will still note that perfectly correct Pascal statements are not acceptable ... (Reducing maxint by one is an ugly solution.)

SOLUTIONS

In some computers, for example the Burroughs B6700, the architecture makes it easy to avoid this problem. However, in most mini- and micro-computers it may appear to be very difficult.

One solution is to substitute a "trip-counter" in the implementation as the loop-controlling value; another is to use the code-template:

```
Source statement
    for v := e1 to e2 do body;
Code template
    temp1 := e1;          {a temporary location}
    temp2 := e2;          {another}
    if (temp1<=temp2) then begin
       v := temp1;
       goto 22;           { violates Pascal rules }
       repeat
          v := succ(v);
       22:
          body;
       until (v = temp2);
    end;
```

Recently, I noted a very simple solution which is applicable to a large class of hardware architectures, notably those that use the condition-code and conditional-branch structures. The equivalent code template in pseudo-Pascal is:

```
temp1 := e1;
temp2 := e2;
v := temp1;
while (v <= temp2) do
   body;
   v := succ(v);
until overflow;
```

In one PDP-11 implementation which had the straightforward while test at the top of the generated code, this was achieved by simply replacing an unconditional branch (BR) at the end of the loop body code by a branch if overflow had not been set (BVC). The net cost in execution speed and space to do it right — nil!

Of course, optimizing compilers that use highly transformed versions of the basic for-statement (for example by moving the test to the end of the loop to save one branch instruction every loop iteration) will need to inhibit the optimization if they cannot determine that the second limit expression cannot ever be maxint. Of course this is not a problem with enumerated types, and may act as a minor encouragement to programmers to use subranges more than type integer - a practice they ought to be employing anyway. (Doing the right thing for the wrong motives still reaps the rewards of virtue...)

ACKNOWLEDGEMENT

The technique reported here is due to Barry Smith, Oregon Software, and is used in (at least) the Pascal-1 X1.2 compiler. Its discovery was prompted by the Pascal Validation Suite.

1979 September 15

Arthur Sale

Arthur Sale

Checklist

0. DATE. Of the information provided.

1. IMPLEMENTOR/MAINTAINER/DISTRIBUTOR. Whatever, but give a person, an address and a phone number. If the source of information is not the person named, give the source too.

2. MACHINE. Obvious.

3. SYSTEM CONFIGURATION. Any known limits on the configuration or support software required, eg operating system.

4. DISTRIBUTION. Who to ask, how it comes, in what options, and at what price.

5. DOCUMENTATION. Specify whatever there is.

6. MAINTENANCE. Is it unmaintained, fully maintained at a profit, or what?

7. STANDARD. How does it measure up to standard Pascal? Is it a subset, or extended? How? Quality?

8. MEASUREMENTS. Of its speed or space, or relative to other systems.

9. RELIABILITY. Any information about field use, or sites installed.

10. DEVELOPMENT METHOD. Outline: to tell what parentage it had and what it is written in.

11. LIBRARY SUPPORT. Any other support for the compiler in object linkages to Fortran, source libraries, etc.

NOTE: Pascal News publishes all the checklists it gets. Implementors should send us their checklists for their products so that the 1000s of committed Pascalers can judge them for their merit. Otherwise we rely on the rumours.

Machine-Dependent Implementations

{ This section summarizes the information we have on Pascal implementations since the last issue, in checklist format where possible. }

Apple Computer: Apple II (Cupertino)

1. IMPLEMENTOR/MAINTAINER/DISTRIBUTOR. Apple Computer Inc, 10260 Bandley Drive, Cupertino, California 95014 (Calif 800-622-9238, other States 800-538-9696).

2. MACHINE. Apple II incorporating 6502 processor.

3. SYSTEM CONFIGURATION. Minimal is Apple II, 48k RAM, Apple Language Card and one mini-floppy disk drive. Works better with two.

4. DISTRIBUTION. Apple dealers. Suggested price $495.

5. DOCUMENTATION. Full set of manuals included in distribution.

6. MAINTENANCE. Supported by Apple Computer Inc.

7. STANDARD. Based on UCSD Pascal(TM), with a reasonably full implementation but several non-standard extensions.

8. MEASUREMENTS. None provided.

9. RELIABILITY. Good, but little field experience as yet. Number of field sites and systems on order not reported.

10. DEVELOPMENT METHOD. Extensively modified from Pascal-P2 via a portable system involving interpretation of a modified P-code instruction set.

11. LIBRARY SUPPORT. Editor provided (written in Pascal), and FILER. Support for graphics and string manipulation.

BESM - 6 (Moscow)

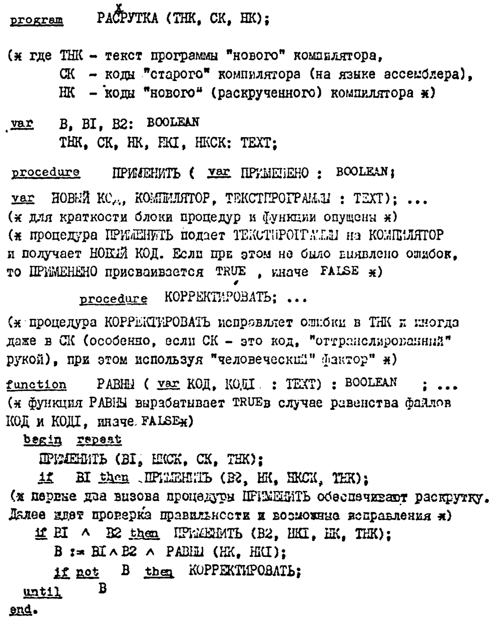
We have obtained a few more details on S. Pirin's Pascal implementation on the BESM-6 from the proceedings of a May 10-15, 1976 conference on Programming Methodology and Program Verification held in Dresden, Germany.

S. Pirin describes how the BESM-6 compiler was derived from the ETH Zurich compiler for the CDC 6600 by changing the code generators to produce BESM-6 assembly code.

The paper describes the advantages of Pascal for programming and its efficient implementation, and describes the bootstrap process. The bootstrap process is itself described by a Russian Pascal program which we reproduce below. The compiler compiled itself in 24 secs, producing 105653 bytes of assembler text. The assembler takes 36 secs to produce the object code of 21507_8 words.

The total bootstrap process thus takes 60 secs. The compiler was made operationally available as Pascal-BESM-6 in the Computer Center in early 1976.

The author of the paper was S. Pirin, USSR Academy of Sciences Computer Center, Moscow. The paper was printed in the proceedings of the Thematischen Konferenz KNWWT, Methodik der Programmierung und Programmverifikation, 10-15 May 1976, Dresden (Technische Universitat Dresden, DDR).



```
program    РАСРУТКА (ТНК, СК, НК);

(* где ТНК - текст программы "нового" компилятора,
       СК  - коды "старого" компилятора (на языке ассемблера),
       НК  - коды "нового" (раскрученного) компилятора *)

var    B, BI, B2: BOOLEAN
       ТНК, СК, НК, НКI, НКСК: TEXT;

procedure    ПРИМЕНИТЬ ( var ПРИМЕНЕНО : BOOLEAN;
var  НОВЫЙ КОД, КОМПИЛЯТОР, ТЕКСТПРОГРАММЫ : TEXT); ...
(* для краткости блоки процедур и функции опущены *)
(* процедура ПРИМЕНИТЬ подает ТЕКСТПРОГРАММЫ на КОМПИЛЯТОР
и получает НОВЫЙ КОД. Если при этом не было выявлено ошибок,
то ПРИМЕНЕНО присваивается TRUE , иначе FALSE *)

        procedure  КОРРЕКТИРОВАТЬ; ...

(* процедура КОРРЕКТИРОВАТЬ исправляет ошибки в ТНК и иногда
даже в СК (особенно, если СК - это код, "оттранслированний"
рукой), при этом используя "человеческий" фактор" *)
function    РАВНЫ ( var КОД, КОДI  : TEXT) : BOOLEAN    ; ...
(* функция РАВНЫ вырабатывает TRUEв случае равенства файлов
КОД и КОДI, иначе FALSE*)
  begin  repeat
    ПРИМЕНИТЬ (BI, НКСК, СК, ТНК);
    if   BI then ПРИМЕНИТЬ (B2, НК, НКСК, ТНК);
(* первые два визова процедуры ПРИМЕНИТЬ обеспечивают раскрутку.
Далее идет проверка правильности и возможные исправления *)
   if BI  ^  B2 then  ПРИМЕНИТЬ (B2, НКI, НК, ТНК);
     B := BI ^ B2  ^  РАВНЫ (НК, НКI);
     if not   B  then  КОРРЕКТИРОВАТЬ;
  until    B
end.
```

BTI-4000, 5000, 8000

We would appreciate ANY information anyone has about these Pascal implementations. Well, how about it?

Burroughs B5700 (Edinburgh)

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Prof Balfour, Head, Dept of Computer Science, Heriot-Watt University, 37-39 Grassmarket, Edinburgh, Scotland. (Information provided by David Cooper, CACI Inc, Keizersgracht 534, Amsterdam, Netherlands.)

2. MACHINE. Burroughs B5700.

3. SYSTEM CONFIGURATION. Not known.

4. DISTRIBUTION. Reported sites at HQ US Army Electronic Command, Fort Monmouth, New Jersey 07703 (Bob Bebeki); Union College, Schenectady, New York, N.Y. 12308 (Nancy Croll).

5. MAINTENANCE. Not known.

6. DOCUMENTATION. Not known.

7. STANDARD. Allows 94-element sets, corrects several errors in earlier version from Oslo.

8. MEASUREMENTS. Claimed considerably faster at compilation than earlier Oslo version.

9. RELIABILITY. "in constant use at Heriot-Watt, both by staff and students. Has been used extensively for projects such as a MODULA compiler, an error- detector-corrector, a frequency analyser and a Diplomacy game."

10. DEVELOPMENT METHOD. Not known. Written in XALGOL.

11. LIBRARY SUPPORT. Not known.

Control Data 6000, Cyber 70, Cyber 170 (Zurich, Minneapolis)

0. DATE/VERSION. Pascal 6000 Release 3; 79/01/01.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER.

Distributors:

(Europe, Asia and Africa)
Ric Collins
UMRCC
Oxford Road
Manchester M13 9PL
England, UNITED KINGDOM
(061) 273-8252

(North and South America)
Wally Wedel
Computation Center
University of Texas-Austin
Austin, TX 78712
U. S. A.
(512) 472-3242

(Australia and New Zealand)
Tony Gerber
Basser Dept. of Computer Science
University of Sydney
Sydney, N. S. W. 2006
AUSTRALIA
61-02-692-3756 or 692-2541

Implementor:
Urs Ammann
Institut fur Informatik
E. T. H. Zentrum
CH-8092 Zuerich
SWITZERLAND

Maintainer:
John Strait / Andy Mickel
University Computer Center
227 EX
University of Minnesota
Minneapolis, MN 55455
U. S. A.
(612) 376-7290

2. MACHINE. Control Data Corporation 6000, Cyber 70 and 170 series.

3. SYSTEM CONFIGURATION. Minimum central memory-32K words. Operates under SCOPE 3.4, NOS/BE 1, KRONOS 2.1 or NOS 1.3 under ASCII subset or CDC scientific character sets and 63- or 64-character sets.

4. DISTRIBUTION. Tape format is binary SCOPE internal, 7/9 track, unlabelled, 800/1600 bpi. Distribution tape includes installation notes, source for compiler, library, software tools and machine- retrievable documentation. Contact the distributor nearest to to you for more information. A release agreement must be signed and the cost is 50 pounds sterling (Manchester), $100.00 (Texas) or $A30.00 (Sydney).

5. DOCUMENTATION. One printed copy each of the following: 70 page supplement to <u>Pascal User Manual and Report</u>, 60 page description of the extended library routines and 60 pages of documentation that describes the various software tools included on the release tape. Machine-retrievable copies of all of this documentation are included on the release tape.

6. MAINTENANCE. Will except bug reports at Minnesota for forseeable future.

7. STANDARD. Nearly full standard. Restrictions include: standard procedures and functions cannot be passed as actual parameters; <u>file of file</u> is not allowed. Extensions include: segmented files and predefined procedures and functions. Extensions new in release 3 include: conformant array parameters; an <u>otherwise</u> clause in <u>case</u> statements; a variable initialization facility (<u>value</u>); a text-inclusion facility for building source libraries and full specificiation of parameters to formal procedure and function parameters. New features in release 3 include: a new post-mortem display; pointers to files; numerous compiler option enhancements; improved run-time tests; more descriptive error messages; interactive support for INTERCOM and TELEX/IAF; many code generation optimizations; numerous bug corrections and improved installation procedures.

8. MEASUREMENTS. Compilation speed: 10800/5800 characters per second on a Cyber 74/Cyber 172. Compilation size: 45K (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 1.7K, compared with FORTRAN minimum of over 7.5K.

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 by 11 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. Run-time system was completely rewritten for release 3.

11. LIBRARY SUPPORT. Allows calls to external Pascal routines, assembler subprograms and FORTRAN (FTN) subroutines. The library supplied on the release tape contains many procedures and functions in addition to the standard Pascal ones. A number of library routines have been added in release 3 including a tangent routine, sorting routines, random number generators, plotting packages, formatted-read routines, double-precision routines, etc.

Data General Eclipse

DG Eclipse (Medical Data Consultants)

PRODUCT DESCRIPTION

MDC PASCAL Version 4 (BLAISE) is an efficient PASCAL compiler and runtime support system designed for the execution of PASCAL programs in a mini-computer environment. The development criteria are as follows:

1. To support interactive I/O in a reasonable way.
2. To be compatible with, as far as possible, existing MDC ECLIPSE RDOS PASCAL Compilers.
3. Close agreement with the P4 'standard'.
4. A reasonable integration into RDOS. (We support background/foreground, subdirectories, and a simple command-line form of activation).
5. Version 4 features high-speed compilation as well as efficient execution.

DATE/VERSION

MDC ECLIPSE RDOS PASCAL Version 4 (BLAISE) January, 1979.

DISTRIBUTER/IMPLEMENTOR MAINTAINER

Ted C. Park
Director, Systems Development
Medical Data Consultants
114 Airport Drive, Suite 105
San Bernardino, CA 92408

MACHINE

Data General - any ECLIPSE-line computer

SYSTEM CONFIGURATION

ECLIPSE must have FPU or EAU
Minimum of 24K words user memory
RDOS REV 6.1 or greater

DISTRIBUTION

Executable object modules and documentation are supplied on 9-track 800 BPI tape in RDOS 'dump' format. The cost is $150.00 to cover our mailing and duplicating costs.

DOCUMENTATION

Machine readable documentation and operating procedures are supplied on the tape, however, it is recommended that the user obtain his own copy of Pascal Users Manual and Report.

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.

STANDARD

PASCAL P4 subset

MEASUREMENTS

Compilation Speed:	300 chars/sec (400 lines per minute)
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 the Data General FORTRAN V compiler
Minimum Memory Needed:	24K words

RELIABILITY

MDC PASCAL Compilers are in use worldwide, and are performing very satisfactorily. At present no known bugs exist.

DEVELOPMENT METHOD

Developed from PASCAL P4. The heart of Version 4 consists of approximately 30K bytes of near optimum coding of the Standard PASCAL-P4 P-CODES. A small but powerful interpreter which executes the P-CODES allows the entire compiler to occupy less than 17K words of memory thus alleviating the necessity of overlaying, swapping or any other virtual memory scheme. An efficient post-processor along with standard Data General utilities and a run-time library supplied on the tape combine to produce an executable core image file.

LIBRARY SUPPORT

The system is totally self-contained so that no Data General libraries are needed.

DG Eclipse (Gamma Technology)

Dear Andy: March 14, 1979

Gamma Tech is happy to announce the completion of our effort to convert the University of Lancaster PASCAL Compiler (RDOS) to Data General's new AOS (Advanced Operating System) on their ECLIPSE and M600 series.

I enclose some information we are getting ready to send to the press, PASCAL contacts and customers, and a copy of the 8-page document for the AOS PASCAL Compiler. Pete Goodeve in Berkeley is responsible for the conversion and is working with Gamma Technology on its distribution and maintenance. The compiler itself and the math routines are the same Lancaster versions in this release. We are committed to a major update as detailed in the enclosed bulletin.

Also I enclosed a checklist for the PUG News, plus some other miscellaneous PASCAL items that have come our way.

Yours sincerely,

Alice Dawson
Gamma Technology, Inc.

AOS PASCAL Bulletin

Gamma Technology, Inc. now has available an AOS implementation of PASCAL based on the Lancaster compiler.

The distribution package presently consists of sources and binaries on 9-track, 800 bpi magnetic tape, an 8-page document and one copy each of the RDOS "User's Guide" and source manuals (for background information). The compiler itself and math routines have not been altered in this release.

We plan to do a major revision of the AOS compiler by July. This release will include:

- fixing known P4 compiler bugs
- conversion to hardware floating point arithmetic
- expansion of the character set to the full ASCII set
- more complete documentation

Feedback from Release I users will also be included in the update.

The pricing schedule for the AOS Lancaster/Berkeley PASCAL Compiler is as follows:

Release I (immed. delivery)	$250.00
Release II update to Release I customers (7/79)	50.00
Release II to new AOS customers (7/79)	300.00

Less $40.00 for previous purchasers of the Lancaster Compiler sources (we are passing on the savings to those customers who have already paid Lancaster's royalty).

Release I for Lancaster RDOS source customers	$210.00
Release II update to Release I customers (7/79)	50.00
Release II for Lancaster RDOS source customers (if Release I has not been purchased)	260.00

Once again, we ask that California customers add the appropriate state tax or enclose a resale certificate form. Foreign customers (except Mexico and Canada) should add $5.00 for additional mailing costs.

0. Date: March 1979
   Version: 1.00

1. Distributor: Gamma Technology, Inc.
   2452 Embarcadero Way
   Palo Alto, CA 94303

   (415) 856-7421
   TWX: 910-373-1296

   Implemented and maintained by Pete Goodeve

2. Machine: Data General Corp. ECLIPSE and M600 Series machines

3. System Configuration: AOS Rev. 2.00 or later
   96 K core memory
   Floating Point Hardware

4. Distribution: $300 package includes sources and binaries on 9-track, 800 bpi magnetic tape in AOS dump format and documentation (see point 5).

5. Documentation: Currently includes 8 page AOS PASCAL document and keysheet. Also included are one copy each Lancaster (RDOS) "User's Guide" and internals manual for reference. User purchase of Manual and Report is strongly urged. PASCAL.DOC and PASCAL.KEY are machine-retrievable.

6. Maintenance Policy: Gamma Technology is committed to a major update of this compiler (extending character set to full ASCII set, math routine conversion, fixing P4 Compiler bugs). We encourage bug reports and will distribute fixes and modifications.

7. Standard: PASCAL P4 subset accepted. Compiler itself is currently unchanged from Lancaster's RDOS version.

8. Measurements: Since AOS is a multi-user/process system, all time measurements are subject to change depending on what is going on in the system. These measurements were done on a quiet system, e.g. PASCAL was the only user.

Program	Source Size (in bytes)	Executable Prgm. File Size (bytes)	Approximate Compilation Time (sec)	Approx. P-code Conversion and Assembly time
Begin/End Program	26	10240	6	12
Graph (Output)	301	10240	10	16
RGCD (example in User's Manual and Report)	330	10240	14	16
Countchars (Input, Output)	727	10240	11	14
Roman # Conversion (Output)	765	10240	10	17
Primes (Output)	1154	10240	14	23
Life (Input, Output)	3060	12288	22	44
P4Compiler	116515	57344	10:33	13:14 (min:sec)

Program	Execution Time (sec)
Begin/End	2
Graph	4
RGCD	2
Countchars	Using Graph as Input - 3 Using Life as Input - 5
Roman	2
Primes	2

Execution Space - The default setting of the compiler allocates 4K bytes for the stack and heap space. This can be changed at either compile or run time by using command switches. Options range from a minimum of 2K bytes to the maximum space available.

All of the small programs executed above were compiled with the minimum stack/heap space. At run-time they all took 6 pages of unshared memory. A page is 2K bytes. AOS allocates memory to processes in page increments. In comparison, SCOM (compare 2 ASCII files), an AOS utility program, takes 3 shared and 5 unshared pages of memory.

Compilation Space - The PASCAL compiler under AOS is a 32K Word swappable process.

As the space and timing figures demonstrate, the larger programs are, the more efficient PASCAL becomes. For example, a lower to upper case converter in PASCAL runs in 6K while a similar program in PL/I needs over 25K.

9. Reliability: The first site has been running for about 3 months. There are now 5 sites. We anticipate that the system will be fairly solid because it is based on University of Lancaster's RDOS implementation (now over 130 sites worldwide).

10. Development Method: P4 Compiler (Wirth) used is same as Lancaster version. The interpreter (DG assembly) was rewritten for AOS. ALGOL libraries no longer required as AOS itself is now the run-time monitor. Effort took about one person-month by a very experienced person.

11. Library Support: External procedures and libraries can be compiled separately and later bound in with a main program. Intermediate P-code, object binary, load map, and symbol table files can be retained. AOS provides library file editors.

DG Eclipse (Rational Data Systems)

Rational Data Systems

21 June 1979

245 West 55 Street New York City 10019 212-757-0011

Dear Andy,

Enclosed is a copy of our 14-page brochure describing our Pascal implementations for Data General computers. It is available free of charge to anyone who writes to us requesting a copy. Feel free to duplicate any portions of it for any purpose you please.

We have five different implementations for various Data General configurations. I have attempted to summarize them per your standard format:

0. DATE/VERSION
New. Availability of the various versions as follows:

AOS:	7/79
RDOS/DOS Single User:	8/79
RDOS/DOS Multi-Terminal:	9/79
RDOS Multi-User (via remapping):	10/79
RDOS/DOS Multi-User (via swapping):	11/79

1. DISTRIBUTOR/IMPLEMENTOR/MAINTAINER
Rational Data Systems
245 West 55th Street
New York City 10019 USA
212/757-0011

2. MACHINE
Data General Eclipse, Nova or microNova.
All configurations and optional instruction sets supported.

3. SYSTEM CONFIGURATION
AOS, RDOS or DOS operating systems.
Single-User DOS will run with floppy disks.
All others require standard system hard disk.

4. DISTRIBUTION

Media: a. 9-track 800bpi Magnetic Tape
b. Data General Floppy Disk
c. 5M byte Top-Load Disk ($200 extra)

Version	License	S.S. Renewal
AOS	$ 3,500	$ 400
RDOS/DOS Single User	2,500	250
RDOS/DOS Multi-Terminal	3,000	300
RDOS Multi-User (Remap)	4,000	500
RDOS/DOS Multi-User (Swap)	4,000	500

5. DOCUMENTATION

User Manual. Distributed both hardcopy and machine-readable. The current version describes differences from J&W and proposed standard as well as operational details. The manual will evolve to eventually become a complete language reference manual.

6. MAINTENANCE POLICY

Initial license includes one year subscription to software updates and fixes. Renewable at the above prices. These are fully supported products. All bug reports accepted. Enhancements already underway. We will be dependent upon customer and marketplace feedback to help determine direction.

7. STANDARD

Used Jensen & Wirth and proposed standard as guide. Extensions include STRING and DECIMAL data types, READONLY and APPEND file accessing, random file positioning via SEEK procedure, TERMINAL files for interactive applications, CLOSE and PURGE procedures to control file disposition, DATE and TIME procedures, generalized procedure SYSCALL for host system interfacing, SEGMENT procedures/functions for automatic load-on-call handling of large programs. See #10 for insight into other changes.

8. MEASUREMENTS

Compilation speed:	355 chars/sec (AOS Eclipse S/130)
Compilation space:	Compiler compiles self with 16kb avail.
Execution Speed:	Compiler compiles self in 8 minutes.
Execution Space:	Interpreter (with all transcendentals, etc.) less than 12k bytes. P-code is byte oriented.

9. RELIABILITY

Excellent (but still new). As of 6/21/79, two test sites for AOS version. All known bugs fixed.

10. DEVELOPMENT METHOD

We began with the UCSD Pascal (TM) compiler which was based upon P2. We made major changes, enhancements and deletions. The hypothetical p-machine has been greatly modified. Our first step was a cross-compiler running on a UCSD-based Z-80 microcomputer. This compiler compiled an Eclipse version which was then moved in object form to the Eclipse. Finally the source version was moved. The interpreters were developed on the Eclipse.

The process has required 14 person-months to date. The impelementors have had previous experience in language implementation and compiler design. The compilers are all written in Pascal.

We have secured proper licensing arrangements for the UCSD Pascal compiler through Softech Microsystems, Inc. Please note that this is NOT the complete UCSD Pascal (TM) System which includes an operating system, text editors and other utilities. We simply used their (very good) compiler as a starting point in the development of our systems.

11. LIBRARY SUPPORT

We offer no assembler language interface or library capability at this time. Both may be influenced by customer reaction. The speeds of the compilers are such that the INCLUDE facility we provide is an adequate substitution for a subroutine library.

A major feature is that compiled code is immediately ready for execution. There is no use of any binder, loader or linkage-editor utility. These utilities are often slower than the compilers themselves. The compiler can compile itself in 8 minutes (see #8) and the output is immediately ready to run.

All five versions are source and p-code compatible thus permitting full cross-compilation capabilities.

Thanks again for your great work.

Sincerely,

Doug

Douglas R. Kaye
President

Digital Equipment DEC PDP-11, LSI-11

{--See also entry under Zilog Z-80, Darmstadt--}

DEC PDP-11 (Berkeley)

Mike O'Dell reports on 79 June 5 that William Joy of Berkeley UNIX Pascal is rewriting it for the new portable code generators of the C compiler. This will mean that Pascal, C, and Fortran are all code compatible and share the same library.

DEC PDP-11 (Stanford Systems Corporation)

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER. Stanford Systems Corporation, Suite 1020, 525 University Avenue, Palo Alto, California 94301 (415-321-8111).

2. MACHINE. DEC PDP-11.

3. SYSTEM CONFIGURATION; 4. DISTRIBUTION; 5. DOCUMENTATION; 6.MAINTENANCE. Not known.

7. STANDARD. "Significant syntactic generalizations: ELSE clauses in CASE statements, embedded assignments in expressions, substitution of expressions for constants, labeled END's for error-checking, relaxation of parameter- passing restrictions, return of additional function value types." { Some of these hardly seem good generalizations... }

8. MEASUREMENTS; 9. RELIABILITY; 10. DEVELOPMENT METHOD; 11. LIBRARY SUPPORT. Not known.

DEC PDP-11 (UCSD Pascal(TM))

Events have again overtaken UCSD Pascal. The name has now been registered as a trademark of the Regents of the University of California, and has been licensed to a single commercial profit-making firm. The address for UCSD Pascal matters is now

SofTech Microsystems, Inc.
9994 Black Mountain Road, Building 3,
San Diego, California 92126 (Phone not known)

All of the UCSD's regular services in support of the UCSD Pascal System have been transferred to SofTech Microsystems, but the University will continue to work in distinct, but related areas.
{ Information derived from UCSD Institute for Information Systems Newsletter #4, popularly known as the Swansong }

DECUS

This is a brief report on DECUS Pascal SIG, for Digital's Pascal users. The current Pascal SIG Chairman is John R. Barr, Dept of Computer Science, University of Montana, Missoula, Montana 59812. The SIG has information on a selection of DEC-10/20 compilers, PDP-11 compilers, and PDP-8 compilers. The Chairman's phone number is (406) 243-2883.

The Pascal SIG Newsletter has a new editor: Charles A Baril, PO Box 1024, University of New Orleans, New Orleans, Louisiana 70122, or Pascal SIG c/o DECUS, One Iron Way, MR2-3/E55, Marlboro, MA 01752. The SIG held a symposium in New Orleans in April, and was addressed by Kathleen Jensen (of Jensen & Wirth fame) on "Why Pascal?", based on her experiences with Wirth and Ammann. There was also a presentation on Pascal for the VAX series. (See Bill Heidebrecht's report in the Here and There Conferences Section.)

In Vol 3 No 1 of the SIG Newsletter we discovered the following highlights

In a letter from the SIG Chairman: "DIGITAL has not yet committed to offer a Pascal compiler for any of their machines. ... Digital is interested in new languages which will provide better programming environments, but is committed to supplying a complete environment including libraries, debuggers and other programming aids. When Ada, the DoD embedded systems language, is defined, DIGITAL will be required to implement complete programming environments for that language. The amount of work required to implement any new language may prevent DIGITAL from offering both Ada and Pascal." If this is so, we echo Gordon Bell's comments: Pascal users on DEC machines will have to do it themselves. What about some concentration on tools now we have a lot of good compilers floating around?

The Pascal SIG Library tape is maintained by Bill Heidebrecht, TRW DSSG, One Space Park, Redondo Beach, CA 90278 (213-535-3136). The library contains "Swedish Pascal" and "NBS Pascal" for PDP-11s, and a number of utility programs. Bill makes a plea for DEC users to check with the Local User Group first for a copy, otherwise check to see if someone nearby has a copy you can borrow, and only in last resort to ask the DECUS library or him for a copy. You can understand why.

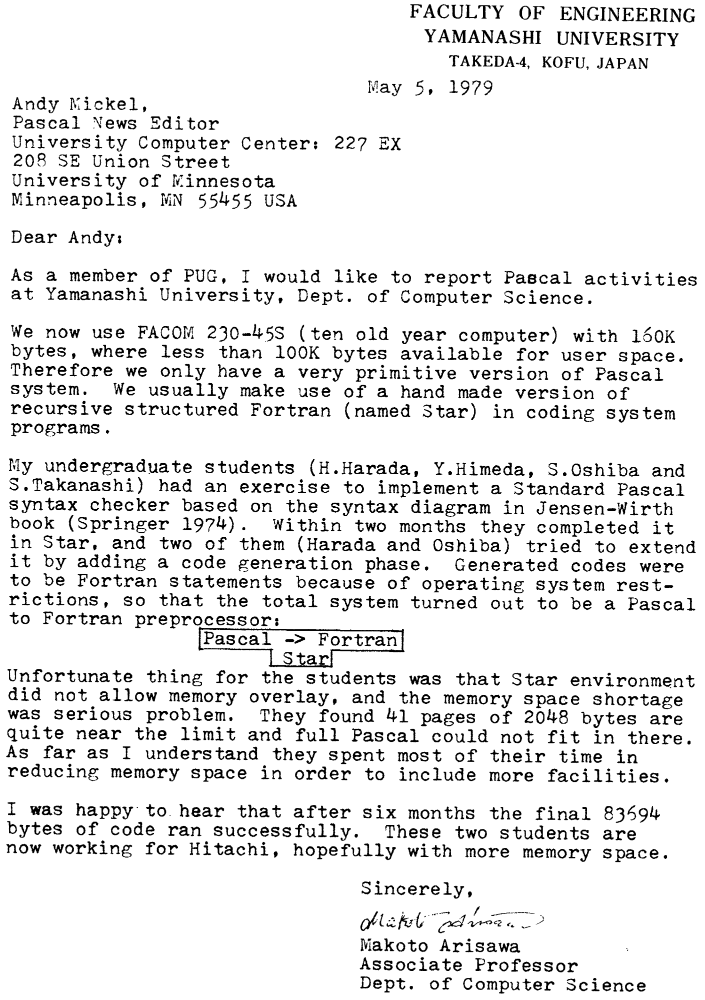
PUG and the DECUS SIG cross-reference each other as a service to Pascal users; after all we are here to help. However, we were perturbed to read in the DECUS SIG Newsletter (Vol 3 No 1 Feb 79) that Bill Page, responsible for Fortran, APL, and other languages such as Pascal on mid-range DIGITAL computers, large PDP-11s and VAX-11, "did not see Pascal in its present form as a language suitable for implementation." {!!!} He "cited the lack of I/O capabilities similar to Fortran's as one drawback." Perhaps the 1000 DECUS SIG members will educate DIGITAL, especially as they are faced with the N machine architectures by M operating systems problem.

Digico Micro 16E

See entry for GEC 4082 (Keele).

Facom 230-45S

The following news of the use of Pascal in Japan may be of interest, especially the target language the compiler generates. { I always said that Fortran was a medium-level assembly language. }



FACULTY OF ENGINEERING
YAMANASHI UNIVERSITY
TAKEDA-4, KOFU, JAPAN

May 5, 1979

Andy Mickel,
Pascal News Editor
University Computer Center: 227 EX
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455 USA

Dear Andy:

As a member of PUG, I would like to report Pascal activities at Yamanashi University, Dept. of Computer Science.

We now use FACOM 230-45S (ten old year computer) with 160K bytes, where less than 100K bytes available for user space. Therefore we only have a very primitive version of Pascal system. We usually make use of a hand made version of recursive structured Fortran (named Star) in coding system programs.

My undergraduate students (H.Harada, Y.Himeda, S.Oshiba and S.Takanashi) had an exercise to implement a Standard Pascal syntax checker based on the syntax diagram in Jensen-Wirth book (Springer 1974). Within two months they completed it in Star, and two of them (Harada and Oshiba) tried to extend it by adding a code generation phase. Generated codes were to be Fortran statements because of operating system restrictions, so that the total system turned out to be a Pascal to Fortran preprocessor:

Pascal -> Fortran
Star

Unfortunate thing for the students was that Star environment did not allow memory overlay, and the memory space shortage was serious problem. They found 41 pages of 2048 bytes are quite near the limit and full Pascal could not fit in there. As far as I understand they spent most of their time in reducing memory space in order to include more facilities.

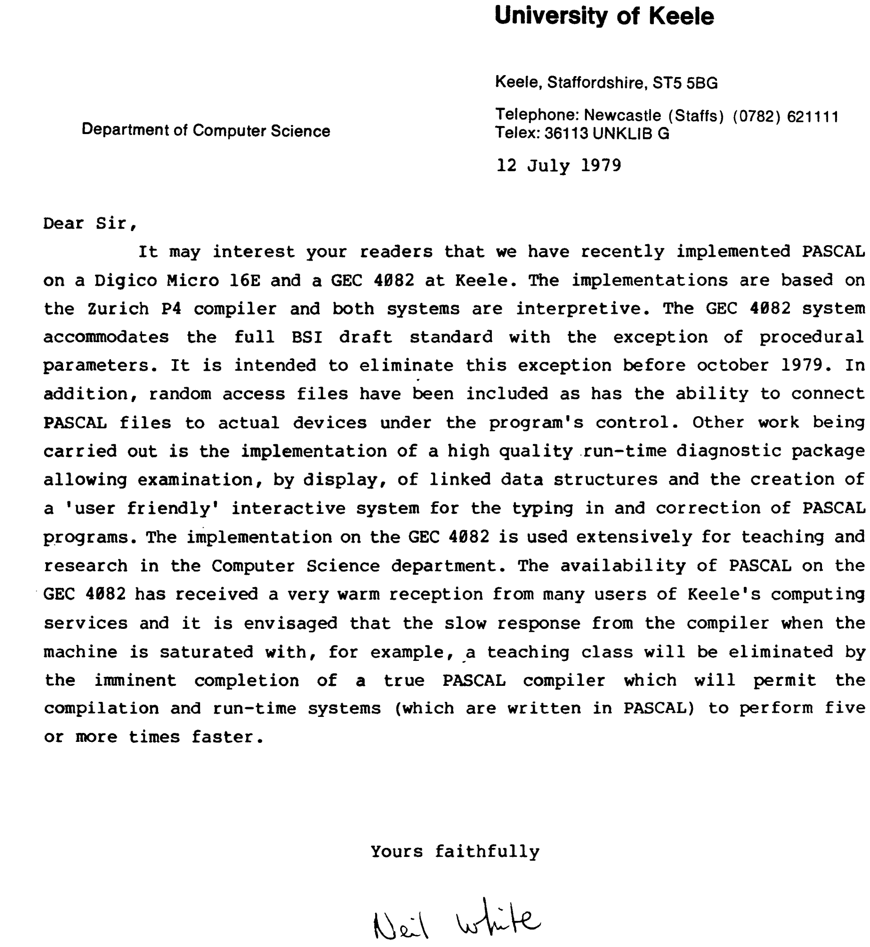
I was happy to hear that after six months the final 83694 bytes of code ran successfully. These two students are now working for Hitachi, hopefully with more memory space.

Sincerely,

Makoto Arisawa
Associate Professor
Dept. of Computer Science

General Electric GEC 4082

{ Are there any more machines waiting to be conquered? Sometimes it seems as though there are no more mountains to climb! }



University of Keele

Keele, Staffordshire, ST5 5BG

Department of Computer Science

Telephone: Newcastle (Staffs) (0782) 621111
Telex: 36113 UNKLIB G

12 July 1979

Dear Sir,

It may interest your readers that we have recently implemented PASCAL on a Digico Micro 16E and a GEC 4082 at Keele. The implementations are based on the Zurich P4 compiler and both systems are interpretive. The GEC 4082 system accommodates the full BSI draft standard with the exception of procedural parameters. It is intended to eliminate this exception before october 1979. In addition, random access files have been included as has the ability to connect PASCAL files to actual devices under the program's control. Other work being carried out is the implementation of a high quality run-time diagnostic package allowing examination, by display, of linked data structures and the creation of a 'user friendly' interactive system for the typing in and correction of PASCAL programs. The implementation on the GEC 4082 is used extensively for teaching and research in the Computer Science department. The availability of PASCAL on the GEC 4082 has received a very warm reception from many users of Keele's computing services and it is envisaged that the slow response from the compiler when the machine is saturated with, for example, a teaching class will be eliminated by the imminent completion of a true PASCAL compiler which will permit the compilation and run-time systems (which are written in PASCAL) to perform five or more times faster.

Yours faithfully

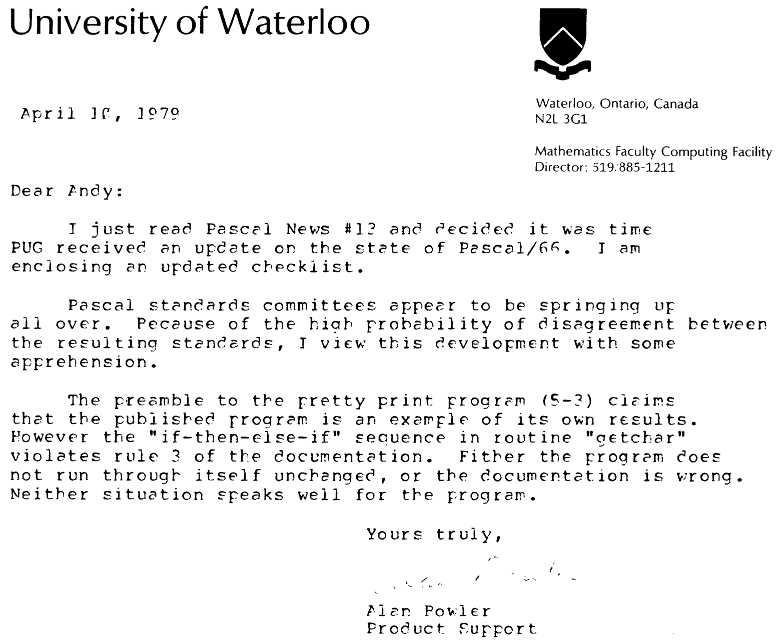
Neil White

Honeywell Level 6

An "extended Pascal compiler" has been developed for Honeywell Level 6 minicomputers by California Software Products Inc (CSPI), Suite 300, 525 North Cabrillo Park Drive, Santa Ana, California 92701. Speeds up to 2000 lines/minute are reported. Estimated cost $6500. However, their last Pascal did not have pointers according to our information. We hear that the people at Oregon Software also may have a compiler. (See entry under DEC PDP-11.)

Honeywell 6000 / Series 60 Level 66 (Waterloo)

On 79 May 13 Peter Rowley sent us a note saying:
"As an undergrad at the Univ of Waterloo who had to struggle with Pascal Version 5, I appreciated the comments of J.Q. Arnold in #11. Pascal 6 is, however, quite pleasant to use and fairly reliable. There are times, though, when one is reminded of the strong influence of the language B on the compiler; this influence sometimes makes portability a problem. (eg the 'procedure main' convention and dynamic file opening."



University of Waterloo

Waterloo, Ontario, Canada
N2L 3G1

Mathematics Faculty Computing Facility
Director: 519/885-1211

April 1C, 1979

Dear Andy:

I just read Pascal News #13 and decided it was time PUG received an update on the state of Pascal/66. I am enclosing an updated checklist.

Pascal standards committees appear to be springing up all over. Because of the high probability of disagreement between the resulting standards, I view this development with some apprehension.

The preamble to the pretty print program (5-3) claims that the published program is an example of its own results. However the "if-then-else-if" sequence in routine "getchar" violates rule 3 of the documentation. Either the program does not run through itself unchanged, or the documentation is wrong. Neither situation speaks well for the program.

Yours truly,

Alan Bowler
Product Support

0. **Date/Version**
   Release 6.1 of Pascal/66 was distributed in January 1979.

1. **Distributor/Implementor/Maintainer**
   Pascal/66 is distributed by Honeywell Information Systems. Actual development and maintenance is done by the University of Waterloo.

   Contact: Dr. W. Morven Gentleman
   Director, Math Faculty Computing Facility
   University of Waterloo
   Waterloo, Ontario, Canada
   N2L 3G1

2. **Machine**
   Pascal/66 runs on Honeywell Series 6000 (with EIS) and Series 60 Level 66 machines.

3. **System Configuration**
   Pascal/66 runs under the GCOS III operating system (release 3/I or later) in timesharing or in batch. The compiler needs 31 or 32k words for most programs, but may grow larger depending on the program being compiled. Compiled programs may be as small as 6k words.

4. Distribution

Pascal/66 is distributed on magnetic tape as a save of the files, programs and documentation necessary to run Pascal. Installation time is estimated at less than 1 man hour.

Pascal/66 is available on a purchase basis. For price information contact your local Honeywell representative.

5. Documentation

A machine readable supplement to the Pascal User Manual and Report is provided. Also included are a set of documentation files for library routines, support programs, and other useful information. A program is provided to allow convenient access to these files from a timesharing terminal.

6. Maintenance

Maintenance is included in the purchase price. Bug reports are accepted no matter how they arrive, but those submitted via the normal Honeywell System Technical Action Requests are guaranteed a reply.

Pascal/66 is undergoing active development to improve its functionality and performance. Current development is aimed at making the B library available to Pascal users. This will give the Pascal user easy access to the full capabilities of the full GCOS III operating environment, and greatly enhance Pascal's usability as system development language.

7. Standard

As with most implementations there are some deviations from the standard.

Violations:

- The keyword "program" and the corresponding "end." (with a period) are not currently implemented. We have not yet invented an interpretation of the program parameters that is meaningful in the GCOS III environment.
- "nil" is a predeclared identifier rather than a reserved word.
- The construct "file of file" is not supported.
- Anonymous tag fields are not yet supported.
- Functions of indeterminate type such as "abs" may not be passed as arguments.
- The words "forward" and "extern" are reserved.

Extensions:

- String constants are adjusted in the obvious manner to conform in type to the variable they are used with in compares or assignments.
- Constant valued expressions (e.g. n+1) are valid wherever a constant is allowed.
- There is an "else" option on case statements and variant records.
- Value ranges are accepted on variant and case labels.
- Null record sections and field lists are allowed.
- Procedures "read" and "readln" will read variables of type "packed array of char".

9. Reliability

Release 6.1 corrected all known and reported bugs. It is considered very reliable.

10. Development Method

This compiler is an independent implementation written in the system programming language B. It is about 11000 lines. It uses an LALR(1) parser implemented using the YACC parser generator. It compiles machine code in standard relocatable object decks. The library is written in B and assembler. The present library is being revised to merge with the standard B library; at present it uses a non-standard B library.

11. Library support

Pascal programs may be linked with separately compiled procedures written in Pascal, Fortran, B or assembler. These routines may be included as object decks or loaded from standard libraries. Facilities are provided in the package to allow easy creation and maintenance of libraries.

Source text inclusion facilities are not presently provided, this is partially because such capability is easily available in the GCOS III environment.

12. Notable features - Details often missed

- Sets are not restricted to a maximum size (other than the availability of address space on the machine). Thus Pascal/66 will run the first 2 versions of Hoare's prime sieve program given in chapter 8 of the Pascal User Manual.
- There is a compile time option to decide if the compiler is case sensitive to identifiers and reserved words.
- Predeclared procedures of fixed type, such as "sin" and "cos" may be passed as arguments.
- Non-local goto's are supported.
- All standard functions, procedures and identifiers are supported.
- Procedures "read" and "write" work with non-text files as per the corrected printing of the Pascal User Manual and Report.
- Procedures are provided to dynamically attach and detach a file.
- Procedures "new" and "dispose" work by managing a free storage list, avoiding the extra overhead and unpredictable behaviour of a garbage collector.

IBM Series 1

Thanks to Neil Bauman of Healtham, and William Hutchison of Ridall & Co, Inc., we now know that both previously reported Series 1 Pascal efforts are defunct: specifically those of Gus Bjorklund and SPAN management.

But new rumours exist. Robin Kasckow and Peter Farley of Decision Strategy Corp., 708 Third Ave, New York, NY 10017 (212-599-4747) have indicated that they may attempt a Series 1 implementation since none seem to be around. Also, IBM itself seems to have partially awakened and has approached the University of Southern California, UC San Diego, University of Minnesota, and finally the University of Illinois about doing an implementation.

IBM 360 or 370

{--Introduction--}

Ever wonder what THEY are THINKing about Pascal? IBM policy is that they have not offered, recommended, or endorsed Pascal. In their view Pascal is a recently developed programming language for instructional applications that generates many questions of availability from university customers. The Pascal expert at IBM seems to be Loren Bullock, Public Sector Marketing (Education Industry), 10401 Fernwood Road, Bethesada, MD 20034 (301-897-2102). Perhaps it would help if we wrote to IBM about PASCAL instead of Pascal?

{--The AAEC compiler running at Amdahl--}

The following letter relates to getting the Australian Atomic Energy Commission compiler up and running on an Amdahl system. The User Guide referred to was received by PUG, so is presumably available on request to Amdahl.

April 30, 1979

J. M. Tobias, G. W. Cox
Australian Atomic Energy Commision
Systems Design Section
New Illawara Road
Lucas Heights, N.S.W. Australia

Dear Jeffrey and George,

Thank you for the tape containing the Pascal 8000 system.

I had very little difficulty bringing the compiler up under VM/370 on our Amdahl system. I made a few minor changes to the run-time system and added a front end that handles the CMS command interface.

I'm sorry, but I don't have any bugs to report. The only difficulties I encountered were due to the somewhat limited support VM/CMS provides for OS macros and services.

While installing the system, I attempted to keep to a minimum the changes to the compiler itself as well as to the run-time system. I did this in the hope that I can install any future version with a minimum of work.

I'm enclosing a copy of the "User's Guide" I put together and a summary of what I did to install the system.

Sincerely,

Robert S Lent

Robert S Lent

Amdahl Corporation
Department of Computer Architecture
1250 East Arques Avenue
Sunnyvale, CA 94086

cc: Pascal User's Group, c/o Andy Mickel

{--A new IBM implementation: Michal Iglewski, Poland--}

Dear Mr. Mickel 28 February 1979

At the end of 1978 we have obtained the implementation of Pascal for IBM 360/370. The System Pascal 360 is derived from the Pascal Compiler developed by Wirth and Amman at ETH Zurich. The preliminary version has been distributed to several European centers. It is also used in some Polish universities. Below we enclose some information about our system and about the way of its distribution.

Yours sincerely,

MIglewski

Michał Iglewski

0. Date/version: 1.11.1978 Pascal 360 release 1.0
1. Distributor/Implementor/Maintainer:
   Implementors: Krzysztof Anacki, Michał Iglewski, Artur
   Krępski, Marek Missala
   Institute of Computer Science
   Polish Academy of Sciences
   Programming Methods Department
   00-901 Warsaw, PKiN, P.O. Box 22
   tel. 200211 (2225)
   telex: 813556
   Maintainer: Distributor:
   M. Iglewski A. Krępski
   address as above address as above
2. Machine: IBM 360 and IBM 370 - compatible machines (The implementation is done on a 360/50)
3. System configuration: operates under OS. The monitor may be modified with minimal effort to run under VS,MVS etc. Minimal required memory is 110K. Standard OS object modules are generated.
4. Distribution: the Pascal 360 system is distributed on a magnetic tape at the density of 800 or 1600 bpi.
   On the tape there are:
   - description of the installing procedure
   - source version of the system (Pascal and assembly code)
   - binary version of the system
   - program to update Pascal programs.

   The tape should be supplied by the user. The Pascal 360 system is distributed free of charge with the right of exploitation till the end of 1981. After that period it is possible to prolongate this permission to unlimited time.
5. Documentation: a supplement to the Revised Report (not available in machine retrievable form)
6. Maintenance policy: The system will be in distribution at least till 1980 by ICS PAS. At the beginning of 1980, the release 2.0, taking into account the users remarks, is expected. We deeply appreciate any critical remarks and comments concerning our system.
7. Standard (accepted language)
   Basic restrictions:
   - files cannot be assigned, passed as value parameters, or occur as components of any structured type; disposition packed for files is ignored; it is not permitted to declare file variables in procedure (functions) activated recursively,
   - sets are limited to x..y where $0 \leqslant ord(x) \leqslant ord(y) \leqslant 63$
   - standard procedures and functions are not accepted as actual parameters
   - the program heading must contain the formal parameter output.

   Technical restrictions:
   - the maximum number of elements of an enumeration type is 256
   - only the first 8 characters of identifiers are significant
   - the length of the object code of a procedure (or of a main program) cannot excess 8192 bytes
   - the types of an actual parameter and of the corresponding formal variable parameter must be the same.

   Additional specifications:
   - the file name in the Pascal program and the name of the corresponding DD card must be the same
   - for every procedure (function) being a formal parameter, the types of its parameters must be specified.

   Extensions:
   - external procedures can be declared
   - the procedure pack and unpack enable the data transfer between two unpacked arrays, too
   - the additional predefined procedures and functions are: date, time, halt, message, clock, expo, linelimit, release, assert.
8. Measurements:
   - compilation speed: about 1670 chars/sec on IBM 360/50
   - compilation space: 160K for small programs

175K for medium programs
225K for selfcompilation

It is possible to reduce the required compilation space by means of overlays. The decrease of compilation space
a) by 19K implies the decrease of compilation speed by 3 %
b) by 51K implies the decrease of compilation speed by 12%.
- execution speed: comparable with Fortran G as shown in the following table

program \ compiler	Fortran H (op=2)	Fortran G	Pascal 360 (T-)	Pascal 360 (T+)	Algol F (T-)	Algol F (T+)
matrix multiplication	1	1.58	1.97	2.95	1.55	1.84
recursive program	1	1.10	0.99	1.16	14.68	15.31
sorting of table	1	2.50	2.30	3.72	5.44	6.31
character count on file	1	1.10	0.25	0.35	2.24	2.39

- execution space: about 3K plus the size of the compiled code, stack and heap.

The compiler generates re-entrant code and may be shared among all users.

9. Reliability: current reliability is moderate to good.

10. Development method: the compiler was developed from Ammann's Pascal CDC 6200 Compiler and transported via cross-compilation (CDC 6200) to IBM 360.
The Pascal 360 system consists of
a) compiler written in Pascal 360 (8600 lines)
b) monitor written in 360 Assembler (3K)
c) monitor support procedures written in Pascal (535 lines) and in 360 Assembler (6K).
During 5 years work (1974 - 1978) on the compiler other smaller software projects have been realized, e.g. the Pascal-P for the IBM 370 and SMAPS - the system of macros and procedures for structured programming in the O.S. 360 Assembler (monitor is written using SMAPS). The actual work on the Pascal 360 system deals with
- improvement of compilation process
- extension of the Pascal file concept to the other O.S. file organizations
- dynamically called procedures
- program generating the profile of Pascal user work
- system for testing Pascal programs

11. Library support: the Pascal 360 user can form a library of subprograms and then use (link) them by means of:
- separate compilation
- call of external procedures (e.g. Fortran) preserving the IBM conventions.

The Pascal 360 utility library (including among others update program, dynamic profile, cross-reference program) has been prepared and will be developped in the future.

{--See also Zilog Z-80 entry (Darmstadt)--}

I.C.L. -- INTRODUCTION (Slightly Revised)

PCHICL - 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;
- Organization of meetings of Pascal users and implementors;
- Acting as a "User Group" to negotiate with Pascal 1900 and 2900 suppliers.

There are currently about 70 people on PCHICL's mailing list, mainly in Computer Science Departments and Computing Centres of UK 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
Inglemire Avenue
Hull HU6 7LJ
England (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 may have that may be of use or interest to other users.

ICL 1900 Series

PASQ Issue 3
This compiler is most suitable for ICL 1900s operating under George 4 and for those with large core store (256k say) operating under George 3. This is the compiler described under the implementation checklist in Pascal News. It incorporates a Diagnostics Package (written by D Watt & W Findlay of Glasgow University) and a source library facility. It takes 44k to compile most programs, 60k to compile itself.

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 compiler lacks some of the facilities of Issue 3, but compiles most programs in 36k, 40k for itself.

XPAC Mark 1B
This compiler is suitable for all ICL 1900s and 2903/4s with at least 32k of core. The language processed is Pascal Mark I, the language of the ORIGINAL report. The compiler takes 24k to compile most programs, 32k to compile itself.

ICL 1900 (Belfast)

0. DATE/VERSION. Updated this issue from letter March 1979.

1. IMPLEMENTOR/MAINTAINER/DISTRIBUTOR. Jim Welsh, Colum Quinn & Kathleen McShane, Dept of Computer Science, Queens University, Belfast BT7 1NN, Northern Ireland (0232-45133). Enhancements by David Watt & Bill Findlay, Computer Science Dept, University of Glasgow, Glasgow G12 8QQ, Scotland, UK (041-339-8855).

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, and diagnostic package only practicable under George 3 or 4.)

4. DISTRIBUTION. Free: send 9-track 1600bpi PE or 7-track 556bpi NRZI tape to Belfast.

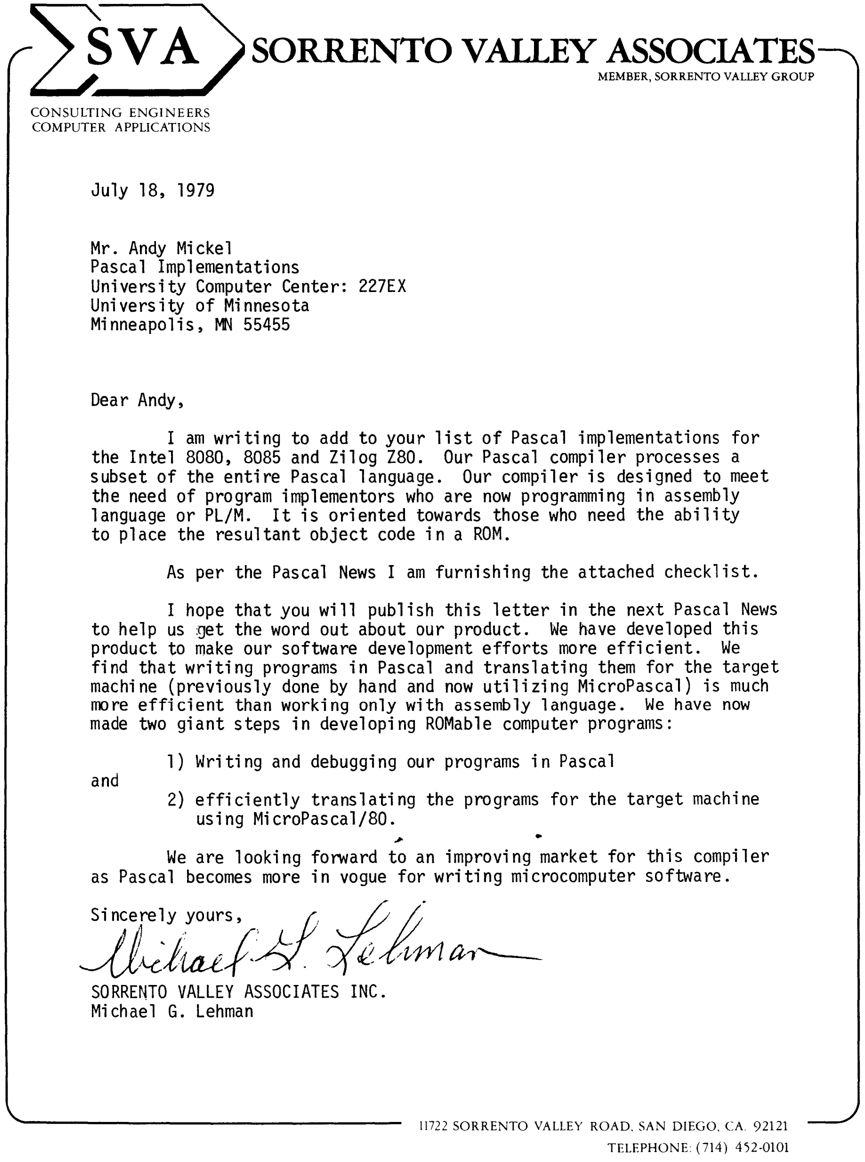
5. DOCUMENTATION. Belfast Users Guide (Supplement to Pascal User Manual & Report) and implementation documentation is distributed with the compiler.

6 - 10. See Pascal News #13; unchanged.

11. LIBRARY SUPPORT. Pascal source library facility.

Intel 8080, 8085, 8086, etc

Intel 8080, 8085, Zilog Z-80 (Sorrento Valley Associates)



SVA SORRENTO VALLEY ASSOCIATES
MEMBER, SORRENTO VALLEY GROUP
CONSULTING ENGINEERS
COMPUTER APPLICATIONS

July 18, 1979

Mr. Andy Mickel
Pascal Implementations
University Computer Center: 227EX
University of Minnesota
Minneapolis, MN 55455

Dear Andy,

I am writing to add to your list of Pascal implementations for the Intel 8080, 8085 and Zilog Z80. Our Pascal compiler processes a subset of the entire Pascal language. Our compiler is designed to meet the need of program implementors who are now programming in assembly language or PL/M. It is oriented towards those who need the ability to place the resultant object code in a ROM.

As per the Pascal News I am furnishing the attached checklist.

I hope that you will publish this letter in the next Pascal News to help us get the word out about our product. We have developed this product to make our software development efforts more efficient. We find that writing programs in Pascal and translating them for the target machine (previously done by hand and now utilizing MicroPascal) is much more efficient than working only with assembly language. We have now made two giant steps in developing ROMable computer programs:

1) Writing and debugging our programs in Pascal

and

2) efficiently translating the programs for the target machine using MicroPascal/80.

We are looking forward to an improving market for this compiler as Pascal becomes more in vogue for writing microcomputer software.

Sincerely yours,

Michael G. Lehman

SORRENTO VALLEY ASSOCIATES INC.
Michael G. Lehman

11722 SORRENTO VALLEY ROAD, SAN DIEGO, CA. 92121
TELEPHONE: (714) 452-0101

MicroPascal/80 Implementation Specification

- 0 - Date: July 19, 1979
Version: MicroPascal/80
Release 1.0

- 1 - Distributor/Implementor/Maintainer

Distributed and Maintained by Sorrento Valley Associates
11722-D Sorrento Valley Road
San Diego, CA 92121
(714) 452-0101

Implemented by: Michael G. Lehman

- 2 - Machine: Intel 8080/8085 and Zilog Z80

- 3 - System Configuration:

The compiler executes under the UCSD Pascal system and thus is portable across a wide variety of systems.

It generates assembly language code in one of two forms:

either a) compatible with the UCSD assembler/linker
or b) compatible with the Digital Research CP/M MAC macro assembler

In either case (a or b) only the run-time routines which are actually used by the user's program are actually included at assembly time.

For interfacing to CP/M we provide a program to transfer files from UCSD file format to CP/M file format.

- 4 - Distribution:

The MicroPascal/80 compiler is distributed on 2-8" floppy diskettes (single density) which contain:

1. Compiler object code
2. Run-time object code for using UCSD linker
3. Run-time source code for using UCSD assembler

Note: These disks utilize UCSD directory format.

Optionally the user may request a third diskette which contains:

4. (In CP/M format): the CPMRTP.LIB file containing the run-time source code.
5. The UCSD to CP/M file transfer program

The disk utilizes CP/M directory format and executes only on an 8080/8085/Z80.

Cost of the above package is $500.00

Source for the compiler is not available for purchase.

- 5 - Maintenance Policy

We will fix bugs promptly for a user for one year from date of purchase.

In the future we are working on versions of this compiler for the DEC PDP-11, Intel 8086 and Zilog Z8000.

- 6 - Standard

MicroPascal/80 does not implement the full standard for Pascal.

This was done to allow efficient code to be generated for a processor like the 8080.

MicroPascal/80 is a pure subset of the UCSD language and contains the following omissions from UCSD Pascal (I.5, II.0):

No LABEL declaration (and therefore no GOTOs).

TYPE declarations for ARRAYs only (to allow passing arrays as parameters).

No RECORD declarations.

No FILE support (because most systems which would utilize this will not have a disk to need support).

Only singly dimensioned ARRAYs.

PACKED is ignored on BOOLEAN ARRAYs.

PROCEDUREs and FUNCTIONs not allowed as parameters.

ALL VARiables and procedure parameters

No STRING data type

No UNIT capability.

- 7 - Measurements

Compilation speed (executing on a 4MHz Z80) is 1000 chars/sec (note this number was derived from 400 Lines/Min * average of 15 chars/line.

Compilation space is a minimum 56K byte system.

Execution speed is estimated to be from 3x to 5x the execution speed of the same program executing interpretively under UCSD system.

Execution space is a minimum of 1.5K bytes and grows from there depending upon the user's program and run-time routines needed.

Compactness of the code is from 2x to 5x as large as the UCSD P-code but the tradeoff point comes at about 24K bytes since MicroPascal/80 does not need an interpreter or operating system to support programs.

- 8 - Reliability

The stability of the system seems good to us at this point. We (and our customers) have been using the compiler for about two months with no major problems.

First release to a customer's site was 79/06/05.

- 9 - Development method

This compiler was written from scratch in Pascal. The total effort to implement was approximately 4 person-months. The implementor had previously implemented about a dozen different compilers for various languages.

- 10 - Library Support

We supply no library of support routines but the user can by using EXTERNAL procedures build a library of supporting routines. We have successfully used MicroPascal/80 to generate "assembly language" subroutines for use in a library.

Prospective users should note that since the compiler produces assembly language, MicroPascal/80 can be used to generate "subroutines" as well as complete programs.

We have developed this product to make our software development efforts more efficient. We find that writing programs in Pascal and translating them for the target machine (previously done by hand and now utilizing MicroPascal) is much more efficient than working only with assembly language. We have now made two giant steps in developing ROMable computer programs:

1) Writing and debugging our programs in Pascal

and

2) efficiently translating the programs for the target machine using MicroPascal/80.

MicroPascal/80 Language Definition

* Legal Constructs:

CONST
TYPE (ARRAY's only)
VAR
PROCEDURE
FUNCTION
IF... THEN... ELSE
CASE... OF
WHILE... DO
REPEAT... UNTIL
FOR... TO... DO
FOR... DOWNTO... DO

* Complete expressions

including the operators:
+,-,*,DIV,/,MOD,AND,OR,NOT

* Single dimensioned ARRAYs

* Integer, Character, Boolean and Real data types

Intel 8080A (DMC Division of Cetec Corporation)

DMC

DMC a Division of Cetec Corporation
2300 Owen Street
Santa Clara, California 95051
(408) 249-1111

November 22, 1978

Dear Dr. Wirth:

It is with pleasure I write to you announcing the release of a new software product by DMC Division of CETEC Corporation.

Our software development staff has produced a PASCAL compiler to run on our 8080A microcomputer floppy disk system, the CommFile. The details are:

1. Implementation

Marketing Department
DMC Division of CETEC Corp.
2300 Owen Street
Santa Clara, CA 95051
(408) 249-1111

2. Machine

8080A

3.	System Configuration	DMC CommFile 130 with 44K bytes of RAM and dual floppies.
4.	Distribution	DMC CommFile 130 with 44K bytes of RAM, dual floppies, and PASCAL compiler retails for $6320.00 U.S.
5.	Documentation	PASCAL Users Manual and Report, second edition. DMC PASCAL Operators Manual.
6.	Maintenance Policy	Full maintenance.
7.	Standard	PASCAL Users Manual and Report, second edition.
8.	Measurements	Not yet available.
9.	Reliability	Stability excellent.
10.	Development	Recursive Descent Compiler.
11.	Library Support	Standard PASCAL Procedures and Functions.

You will be kept informed as we develop PASCAL further at DMC.

Very truly yours,

Phil Devin
Manager
Marketing Support

Phil Devin/md

Intel 8080, 8086, Zilog Z-80, Z-8000 (Microsoft)

The Microsoft Pascal is to be compatible with UCSD, ANSI and ISO Pascal. The target processors are 8080, Z-80, 8086, Z-8000 and LSI-11, and will run under CP/M on 8080 and Z-80, and is expected early in 1980.

There appear to be some un-needed extensions; the following list is selected from some documentation we received:

- predefined type WORD (16-bit unsigned integer) {??}
- attributes for variables:
  STATIC, INITIAL, ORIGIN, REGISTER, INTERNAL, EXTERNAL
- capabilities from the C language {!!}
  embedded assignment operator
  increment and decrement operators
- control structure extensions { when we have too many already }
  BREAK and CYCLE in FOR, WHILE & REPEAT
  RETURN statement
  FOR variable IN set DO statement
- address functions PEEK and POKE

Fortunately, the language will be structured in levels, and at the best level looks rather like Pascal ought to look. At the "Extended" level and the "System" level these rather useless and dangerous features are enabled, according to the manual to give "the ability to easily do in Microsoft Pascal those operations that are easy in assembly language". We always thought that Pascal was supposed to preserve us from undesirable practices and lead us away from temptation. Readers of the News may like the following two examples from the SYSTEM level of the Microsoft Manual; we do not:

```
    ALPHA[I.=(BASE+INCR(Q))]:=ALPHA[I*2-1]+J
    FOR IX:=1 TO J.=(LIMIT + 2 * INCR) DO ...
```

Apart from these additions, the standard level of Microsoft Pascal looks like being a good job.

Intel 8080 (TSA Software ASP)

TSA SOFTWARE, INC

203 261-7963
39 WILLIAMS DR., MONROE, CT. 06468

79.3.9

Dear Andy, and fellow Pascal - Ligraphers

(caligraphy is the art of fine hand-writing and
Pascal is the....................)

As you can see from the date of my PUG renewal check (78.11.7), this letter has been a long time in the finishing, I hope it is useful.

It is important that the reader understands the machine environment I work in, because it is very different from the usual Pascal environment. I work primarily on systems programs for micro-computers. We deal with "BIG" micros - 32K Bytes or more, at least a mini-floppy disk (80K)and usually a video display terminal and printer. We sell operating systems and related support software, with occasional applications projects.

The net result is an machine environment with:

(1) Very limited memory
(2) Very limited and slow disk storage
(3) Medium speed but totally unaided processor 8080/Z80 (no I/O or auxiliary processors)
(4) Minimal operating system support, of the CP/M variety. (no protected anything - memory or I/O)
(5) Very low budget projects, with no or minimal institutional support
(6) Absolute reliability requirement (business software) with very naive users.

All in all, a rather harsh operating environment. As a result, most programming is either assembler or assembler. Business software is done primarily using a rather poor selection of Basics.

I've been using Pascal as a design language since 1975 when Pascal - P2 came out, but haven't had a compiler to actually use. When USCD Pascal came out, I had hopes for it, however it does't run within our software environment. It is interpretive and does not provide escape to assembly code when necessary. At that point I broke down and initiated our "ASP" project. "ASP" (a small/system Pascal, TM -TSA Software) is a full compiler, and outputs 8080 assembler for use with our 8080 linking assembler. (much to most people's amazement, most micro computer assembly code is still written with absolute non-linking assemblers.) It is detailed in the attached implementation checklist.

The discussions herein are related to our experience with our compiler and using Pascal in a general system environment. In some cases, our own solutions are discussed; in others, a plea for suggestions is made.

I find the current discussion in the popular computing periodicals abut Pascal, rather amusing; since I see a vast difference in the place of Pascal vs Basic. Pascal is not a friendly language, in fact to be so, would fail it's primarily requirement: To allow the programmer to produce functional, reliable, maintainable programs. Basic, on the other hand, is appropriate to an environment where laxity and interactive processing is more appropriate. The problem as to when a program crosses the dividing line and how to place it in the correct environment initially is the critical item, but beyond the scope of this letter.

Implementation Checklist

The TSA Software 'ASP' (tm) compiler is a minimal implementation of Pascal. It is intended to be the bottom end of a line of compilers. 'ASP' - A small Pascal or a system Pascal provides basic functions for system programming and acts as a basis for application programming.

0. Date / Version: 79.2.5; ASP/1 version x00.14

1. Implementor: Richard Roth
TSA Software, Inc.
39 Williams Drive
Monroe, Connecticut 06468
(203) 261-7963

2. Machine: 8080 / Z80 / 8085 Micro Processor

3. Configuration: 32K..64K Bytes
At least one floppy disk
Running CP/m, CDOS, IMDOS, TSA/OS
or any other compatable operating system

4. Distribution: ALPHA test copies only being supplied

5. Documentation: 40 pages of test notes, and library calling sequences, 10 sample programs

6. Maintenance: Not defined yet

7. Standard: Major subset of Pascal
(A) All program structures except CASE, WITH
(B) Only scalar variables and arrays.
Pseudo--Structures using 'CONST' offsets and 'type casting'. Value procedure parameters only

Extensions:
Text file include
External and module declaration
Static data initialization
In-line machine code
String functions: CONCAT, SUBSTR, etc.
Bit-wise boolean on integers

8. Measurements: Compile: 230 line/min. to 8080 Macro assembler
Total: 24 line/min. to linked executable code
10K Bytes for compilier
Execution: Full 8080 machine code
Library size: String- 1600 bytes
I/O- 6200 bytes
Real- 1800 bytes
General- 260 bytes

9. Reliability: Still in development
Rev X00.00 since September 78
2 Alpha test sites since December 78

10. Development
Recursive decent technique
Coded in 8080 machine code
Outputs macro's, table driven for different macro formats of assembler code
Approximately 70K Bytes of source code (2K lines)
3-4 man-months of super programmer time.

11. Library / Support
Linkable support library for:
Variable length strings
32 Bit / 16 bit integers,12 digit reals
Sequential and block random I/O,recursive coding.
Source file include with some supplied external declarations
Utilities: Symbol cross-reference, Documentation comment printer

Interdata

See Perkin-Elmer (change of company name).

Modcomp II & IV

Larry D Landis, United Computing Systems, 2525 Washington, Kansas City, MD 64108 reports that Syd Weinstein (a co-worker) says that the University of Illinois School of Medicine has a ModComp Pascal. No other details. (78 Nov 17)

Also Eugene N Miya, Pascal Development, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91103 (213-354-4321) reports that JPL is undertaking an effort to come up with a Pascal compiler for the ModComp II and IV. (79 Mar 08)

Motorola 6800

Control Systems Inc, Kansas City, KS, seem to have a 6800 version of Pascal. Sorry, no more information do we have.

Nord-10 & Nord-100

Terje Noodt
Computing Center, University of Oslo
Pb. 1059, Blindern
Oslo 3, Norway

May 14, 1979

Dear Andy,

Could you please send me another copy of Pascal News number 13? In my copy pages 85 to 94 are missing. done.

The work you have done for PUG and Pascal has been tremendous - I can understand that you feel you've had the burden long enough. I only pray that PUG doesn't die.

We have now finished a new version of Pascal for the Nord-10 and the recently announced Nord-100. A description is enclosed, together with a copy of the User Manual.

Yours sincerely,

T. Noodt

Terje Noodt

<u>Nord-10 and Nord-100 Pascal</u>

0. DATE/VERSION. 79/04/23

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER.
   Implementors: P. Gjerull and T. Noodt,
   Computing Center, University of Oslo
   Pb. 1059, Blindern
   Oslo 3, Norway
   Distributor: Norsk Data A. S.
   Pb. 4, Lindeberg gård
   Oslo 10, Norway
   Maintainer: The implementors and distributor in collaboration.

2. MACHINE. Nord-10 and Nord-100.

3. SYSTEM CONFIGURATION. Nord-10 or Nord-100 running SINTRAN III. A Pascal program may use up to 128K of virtual memory.

4. DISTRIBUTION. From Norsk Data A.S. on floppy disks.

5. DOCUMENTATION. User Manual (40 pages) describing use of Pascal system, restrictions and extensions. Machine retrievable.

6. MAINTENANCE. Norsk Data grade A (highest level).

7. STANDARD. Restrictions: Declaration of file variables in main program only. MARK and RELEASE implemented instead of DISPOSE. Extensions: Initialization of main program variables. Files may be opened dynamically. Separately compiled Pascal and FORTRAN procedures may be called. Several minor extensions and utilities.

8. MEASUREMENTS. Performance comparable to Nord FORTRAN (estimated).

9. RELIABILITY. Good.

10. DEVELOPMENT METHOD. Developed from the TRUNK compiler. Produces standard relocatable code (BRF).

11. LIBRARY SUPPORT. A set of external utility procedures to interface with the operating system.

Perkin-Elmer 7/16 (Melbourne)

{ running Brinch-Hansen's "Sequential Pascal" }

TELEPHONE
345 1844

TELEGRAMS
UNIMELB PARKVILLE

University of Melbourne

DEPARTMENT OF COMPUTER SCIENCE

Parkville, Victoria 3052

7th June, 1979.

Dear Andy,

I am writing in response to queries in the Pascal User's Newsletter concerning Pascal on the Interdata 7/16. You and some of your readers may be interested to know that we have had Brinch Hansen's Sequential Pascal running on our 7/16 since mid-1977. I have included a description of our system in the form of implementation notes, and will welcome any inquiries that are made as a result of these notes.

Yours sincerely,

Enc.

<u>Joe Longo</u>.

Ø VERSION:

Brinch Hansen's Sequential Pascal

1 IMPLEMENTORS:

JOSEPH LONGO,
DEPT. OF COMPUTER SCIENCE,
UNIVERSITY OF MELBOURNE,
PARKVILLE, VICTORIA, 3105,
AUSTRALIA.

2 MACHINE:

Interdata 7/16, with high-speed ALU and 64 Kb memory

3 SYSTEM CONFIGURATION:

Home-grown "Hynos" disk-oriented operating system provides the host environment, but its support functions can be easily provided in a stand alone environment.

4 DISTRIBUTION:

The original distribution tapes and documentation from which this implementation has been derived can be obtained from the distributor for a total cost of $US60.

5 DOCUMENTATION:

"Sequential Pascal Report", per Brinch Hansen, Alfred C. Hartman, Cal.Inst.Tech., July 1975 (comes with the distribution tapes and notes.) "The Architecture of Concurrent Programs, per Brinch Hansen, Prentice-Hall.

6 STANDARD:

Sequential Pascal is a subset of Pascal. Some of the differences/ limitations are:

- no "goto" statements (and therefore no "labels")
- maximum set size: 128 elements
- no nested procedure definitions
- non-standard input-output: I/O defined at compilation time through "prefix procedures"
- procedure names can not be passed as parameters in procedure calls.

7 MEASUREMENTS:

The seven-pass Sequential Pascal Compiler compiles at a rate of approx. 6 lines per second, but is 30% I/O bound within the Hynos operating system. The compiler requires a 16-17Kb program space and 12-13Kb data space.

Code produced by the compiler is interpretive. The average execution time of a virtual instruction is about 40 micro-secs.

8 RELIABILITY:

Very good.

9 DEVELOPMENT METHOD:

Sequential Pascal is an interpretive language developed by Brinch Hansen for use in writing utility programs for and as the job-control language of Concurrent Pascal Programs. The original interpreter was written in PDP-11 assembly code and was transferred to the Interdata 7/16 with about one man-month of effort. Translation of the interpreter from the PDP-11 into 7/16 assembly code was relatively simple. The difficulty encountered arose from trying to implement Sequential Pascal outside of its Concurrent Pascal environment. Not only did we have to make our operating system respond to the system calls as would Concurrent Pascal, but also we found it necessary to investigate, at a very basic level, the operations of the Concurrent Pascal Compiler in maintaining the working environment for program execution. These operations are transparent to the Sequential Pascal programs and unfortunately none of this work for implementing Sequential Pascal on its own is documented by the developers. Finally, the size of the Interdata Interpreter is about 4Kb (compare this to 2Kb for the PDP-11) but includes all of the virtual instructions needed for interpreting Concurrent Pascal code also.

10 LIBRARY SUPPORT:

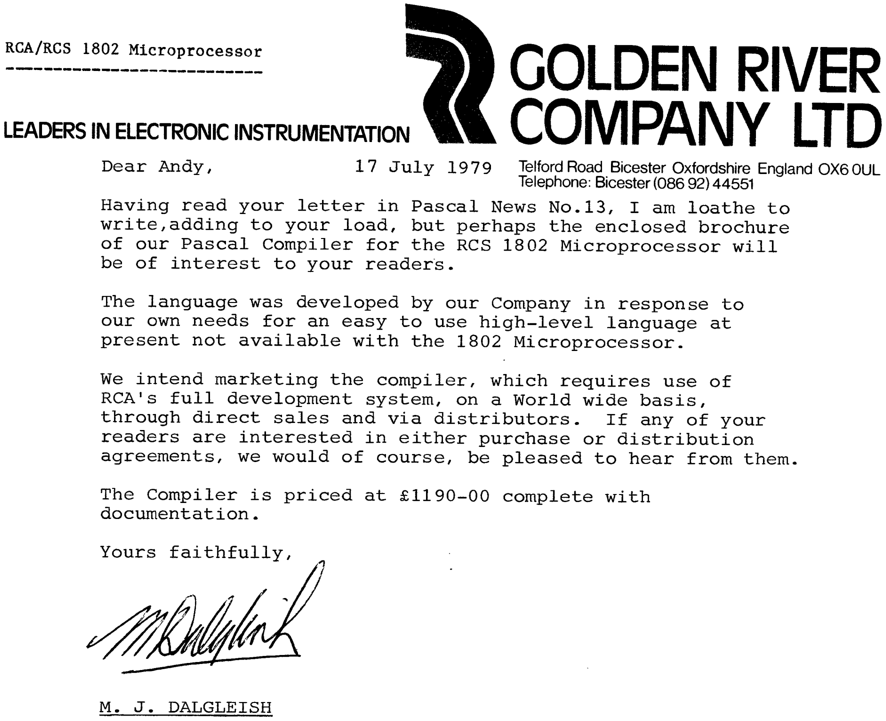
One of the features of Sequential Pascal is that all library routines are defined as "prefix procedures" at compilation time. This feature has been used extensively to enable our Sequential Pascal programs to exploit a number of facilities available in the host environment. This means that, apart from the basic procedures described in Brinch Hansen's book (see 5 above), all other library routines are entirely implementation dependent. It is conceivable that this facility may be used to link to FORTRAN programs, but we have no intentions of doing so.

One of the prefix procedures defined by Brinch Hansen, called "RUN", enables a Sequential Pascal program to execute another sequential program. It is not an overlay in that, to the calling program, it appears like a normal procedure call, but it is a very useful method for linking separately compiled programs at execution - rather than at load-time. In fact this is what makes the running of the seven-pass compiler feasible.

Perkin-Elmer 3220 (Champaign)

Roger L Gulbranson, Nuclear Physics Research Laboratory, University of Illinois, 23 Stadium Drive, Champaign, IL 61820 (217-333-3190) reports that he is writing data acquisition software (to perform at a rate of 10000 samples/second) on his new 3220 written in Concurrent Pascal. He will also be improving the efficiency of the kernel and the Pascal compiler's code generator.

RCA/RCS 1802 Microprocessor



GOLDEN RIVER COMPANY LTD

LEADERS IN ELECTRONIC INSTRUMENTATION

Telford Road Bicester Oxfordshire England OX6 0UL
Telephone: Bicester (086 92) 44551

Dear Andy, 17 July 1979

Having read your letter in Pascal News No.13, I am loathe to write,adding to your load, but perhaps the enclosed brochure of our Pascal Compiler for the RCS 1802 Microprocessor will be of interest to your readers.

The language was developed by our Company in response to our own needs for an easy to use high-level language at present not available with the 1802 Microprocessor.

We intend marketing the compiler, which requires use of RCA's full development system, on a World wide basis, through direct sales and via distributors. If any of your readers are interested in either purchase or distribution agreements, we would of course, be pleased to hear from them.

The Compiler is priced at £1190-00 complete with documentation.

Yours faithfully,

M. J. DALGLEISH

{ Oxfordshire }

0. DATE. 1979 July 17

1. DISTRIBUTOR. Golden River Company Ltd, Telford Rd, Bicester, Oxfordshire, OX6 [illegible] England. (08692-44551)

2. MACHINE. RCA 1802 Development System.

3. CONFIGURATION. 20k RAM, CDP18S Dual floppy drives, RS232-compatible terminal.

4. DISTRIBUTION. 1190 pounds sterling for licence of nominated system only. Distribution medium: floppy disk.

5. DOCUMENTATION. Printed User Manual (not machine retrievable).

6. MAINTENANCE. For forseeable future.

7. STANDARD. Pascal subset implemented. No reals, enumerated or subrange types, no variant records, no binary i/o, no integer or real i/o to text files, no nested procedure declarations, 64-element set limit, maxint=32767, no file declarations, packed not implemented.

8. MEASUREMENTS. Compiles in 17k bytes, run-time support requires 2-3k byte kernel. No speed given.

9. RELIABILITY. Not known.

10. DEVELOPMENT METHOD. 3-pass compiler with intermediate results to disk.

11. LIBRARY. None specified.

Siemens 7-748

See also Zilog Z-80 (Darmstadt) entry

Southwest Technical Products SWTP6800

Lucidata

oosteinde 223 voorburg
telephone 070-862387
bank: a.b.n. voorburg
account 516610384
registration no. 86871
the hague chamber of commerce

7th June, 1979

Dear Sir

Please include the enclosed CheckList in your next Newsletter.

Sincerely,

Dr. N.W. Bennée

P-6800 PASCAL - CHECKLIST FOR PUG NEWSLETTER

0. DATE/VERSION

Version 1 released May 1979.

1. IMPLEMENTOR/DISTRIBUTOR/MAINTAINER

Lucidata,
Oosteinde 223,
Voorburg,
Holland.

2. MACHINE

South-West Technical Products 6800 or equivalent.

3. SYSTEM CONFIGURATION

Mini floppy disc with 12K + 4K bytes memory as a minimum configuration, using the Technical Systems Consultants mini FLEX or FLEX 2 Operating System.

4. DISTRIBUTION

Lucidata.
The cost is 300 Dutch Guilders (approx. 150 US dollars) for the compiler, the run-time system, utilities and demonstration programs on a floppy disc, together with the documentation.

5. DOCUMENTATION

User manual. (Not machine retrievable).
Gives details of the PASCAL subset, sufficient information on the run-time system to permit building of customised/specialist systems, and specimen programs. A list of PASCAL books is included, and the address of PUG!

6. MAINTENANCE

Matters requiring attention should be reported to Lucidata. Subsequent releases will include any corrections which may be necessary.

7. STANDARD

Version 1 is a self-compiling subset of PASCAL. Principal omissions are records and pointers, with certain restrictions on type declarations. Version 2 (planned for late 79 release) will include more features.

8. MEASUREMENTS

Compilation speed: depends on the amount of memory in the configuration, but is independent of program size. A page mode (which is about half as fast as normal mode) is invoked automatically if there is insufficient memory for any program (e.g. the compiler) and its stack space.

Speeds measured for self-compiling the compiler on a 1 MHz system with SWTP MF-68 dual floppy discs are as follows:

32K bytes : 78 characters/second (130 lines/minute)

24 + 4K	: 44 characters/second (74 lines/minute)
20 + 4K	: 42 characters/second (70 lines/minute)
16 + 4K	: 32 characters/second (54 lines/minute)

Execution speed: finds all 92 solutions to the Eight queens problem in 58 seconds, using the recursive alogrithm given in "Algorithms+Data Structures=Programs", by N. Wirth.

Execution space: between 3K and 4K bytes for the run-time system, depending on the number of different P-codes to be executed, plus space for the P-code instructions for the programs - typically 12 bytes per line of source PASCAL, plus stack space.

9. RELIABILITY

So far, excellent - but insufficient use by non-professionals to make a meaningful claim.

10. DEVELOPMENT METHOD

Two pass recursive descent compiler which generates P-code in fixed length 4 byte format, executed by the run-time system.. Bootstrapped up from a much smaller subset of PASCAL.

11. LIBRARY SUPPORT

Separately assembled routines may be linked in.

Sperry-Univac V77 (Irvine)

Sperry Univac Minicomputer Operations has announced Summit, a multi-task operating system for V77-800 & V77-600 minicomputer systems, supports Pascal as a component. Prices seem to be $6000 for Summit and $2000 for Pascal.
Write to Sperry Univac Minicomputer Operations, 2722 Michelson Drive, Irvine, California 92713 (714-833-2400 X536) or London, NW10 8LS, England or 55 City Centre Drive, Mississauga, Ontario L5B1M4, Canada.

Tandy Radio Shack TRS-80

A UCSD Pascal System has been announced by FMG Corporation (PO Box 16020, Fort Worth TX 76133 Phone: 817-294-2510) for the TRS-80. The package costs $150 and requires a 48k system with two disk drives.

Texas Instruments 9900

Ticom Systems (10100 Santa Monica Blvd, Suite 862, Los Angeles, CA 90067, Phone 213-552-5328) have announced a version of Pascal for the TI 9900. Our blurb from Michael Hadjioannou was not in the form of a checklist and contained no technical details.

Univac

See Sperry-Univac

Zilog Z-80

Zilog have announced Z-80 Pascal at $950 from Zilog at 10340 Bubb Road, Cupertino, California 95014. Very little more is known at PUG HQ.

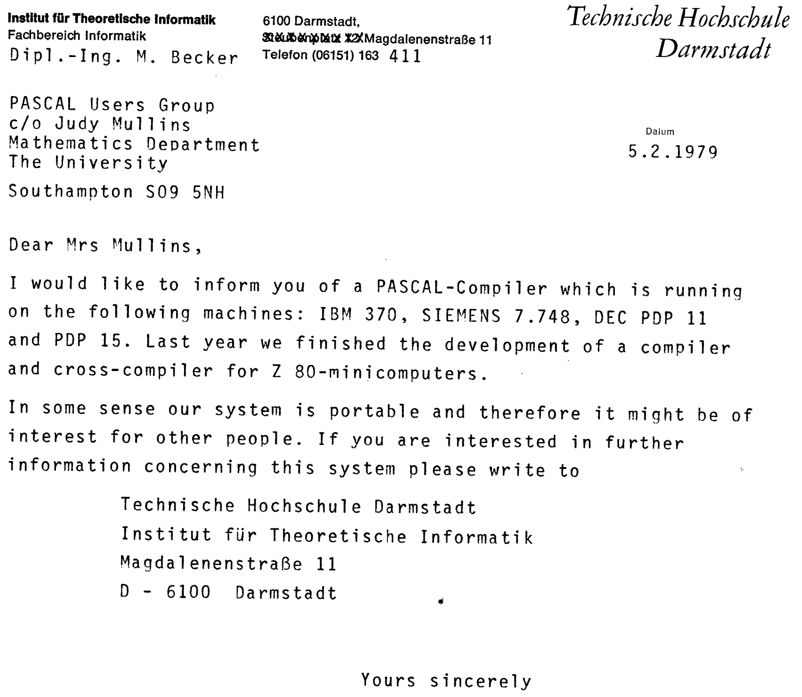
See also Intel 8080 (SVA, Microsoft).

Zilog Z-80 (Ithaca Audio Pascal-Z)

Ithaca Audio, P O Box 91, Ithaca, NY 14850 (607-257-0190) have announced "the first Pascal compiler for the Z-80, and the fastest Z-80 Pascal ever is now ready" (Byte, 79 July). The compiler requires the Ithaca Audio K2 operating system and 48k memory. The output is native assembly code for the Z-80, which has to be assembled through the [illegible]. Price: $175.00; distribution: 8" K2 floppy disk.

Zilog Z-80 (Darmstadt)

The following letter was received by a PUG member on 79 Feb 5, from Dipl-Ing M. Becker.



Institut für Theoretische Informatik
Fachbereich Informatik
Dipl.-Ing. M. Becker

6100 Darmstadt,
~~Steubenplatz 12~~ Magdalenenstraße 11
Telefon (06151) 163 411

Technische Hochschule Darmstadt

PASCAL Users Group
c/o Judy Mullins
Mathematics Department
The University
Southampton SO9 5NH

Datum
5.2.1979

Dear Mrs Mullins,

I would like to inform you of a PASCAL-Compiler which is running on the following machines: IBM 370, SIEMENS 7.748, DEC PDP 11 and PDP 15. Last year we finished the development of a compiler and cross-compiler for Z 80-minicomputers.

In some sense our system is portable and therefore it might be of interest for other people. If you are interested in further information concerning this system please write to

Technische Hochschule Darmstadt
Institut für Theoretische Informatik
Magdalenenstraße 11
D - 6100 Darmstadt

Yours sincerely

M. Becker

Zilog Z-8000

See Intel 8080