ACM

# SIGPLAN Notices

A Monthly Publication of the

SPECIAL INTEREST GROUP ON PROGRAMMING LANGUAGES

Volume 11, Number 7, 1976 July

## Contents:

CORRESPONDENCE: Jim C. Warren, Jr., John Hopkinson, Ben Consilvio 1

NOTICES: X373 Toward IF-THEN-ELSE in Current Revision, Forthcoming meetings of X3J3, Fortran Tutorial at ACM 76, Module protection in SIMULA. 5

NEWS OF PAST EVENTS: Recent Meetings of the San Francisco Peninsula SIGPLAN Chapter 7

OFFICIAL SIGPLAN MESSAGES: NOTE FROM YOUR TREASURER 9

PUBLICATIONS: compiled by Ronald L. Lancaster 12

TECHNICAL CONTRIBUTIONS:

Julius A. Archibald, Jr. Towards a Dynamic (FORTRAN) Programming System.

Dennis Allison, happy Lady, & friends. Design Notes for Tiny Basic. 25

Richard K. Bennett. Build: A Primitive Approach to the Design of Computer Languages and their Translators. 34

Robert E. Brown. Toward a Better Language for Structured Programming. 41

Ronnie G. Ward. A Non-Iterative Situation Case Statement. 55

M. H. Williams. A Question-Answering System for Automatic Program Synthesis. 63

# SIGPLAN NOTICES

## special interest group on programming languages

SIGPLAN *Notices* is an informal monthly publication of the Special Interest Group on Programming Languages (SIGPLAN) of the Association for Computing Machinery.

Membership in SIGPLAN is open to ACM members or associate members for $11 per year. It is also open to others whose major professional allegiance is in a field other than information processing or computing for $17 per year. All SIGPLAN members receive SIGPLAN *Notices*, are given discounts at SIGPLAN-sponsored meetings, and may vote in the Group's biennial elections. ACM members of SIGPLAN may serve as officers of the group.

Institutional or Library subscriptions to SIGPLAN *Notices* are available for $25 per year, and the regular back issues of the *Notices* may be purchased for $3 per copy from ACM Headquarters. SIGPLAN symposium and conference proceedings issues of the *Notices* are available, at different prices, from ACM Headquarters.

Members of the Special Technical Committee on APL (STAPL), a SIGPLAN-affiliated users' group, receive the *APL QUOTE QUAD*, an informal quarterly publication of interest to APL users.

All correspondence concerning STAPL should be directed to the committee chairman, Professor Garth Foster, Department of Electrical Engineering, 101 Link Hall, University of Syracuse, Syracue, New York 13210.

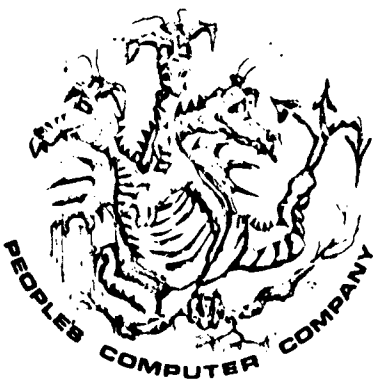
CHANGES OF ADDRESS AND OTHER MATTERS PERTAINING TO THE SIGPLAN MAILING LIST should be sent to ACM Headquarters: ACM SIGPLAN, 1133 Avenue of the Americas, New York, New York, 10036. Phone 212/265-6300.

CONTRIBUTIONS TO SIGPLAN *Notices* may be sent to the Editor. They should be camera-ready and typed with single spacing. Contributed papers 15 pages or longer will be returned to authors for retyping on model paper and ultimate photoreduction. Announcements or copies of relevant company reports and academic theses are requested for inclusion in the Publications section. Those interested in editing a special issue devoted to a specific topic are invited to contact the Editor to discuss the matter. Letters to the Editor of SIGPLAN *Notices* will be considered as submitted for publication unless they contain a request to the contrary.

Technical papers appearing in this issue are unrefereed working papers, and all contributions are ordinarly to be construed as personal rather than organizational statements. Authors planning to submit to a journal any manuscript appearing in SIGPLAN *Notices* should make certain that the journal version differs sufficiently in content to satisfy the editor of that journal. All questions regarding journal policy on possible duplicate publications should be directed to the editor of the journal in question.

PROSPECTIVE ORGANIZERS OF SIGPLAN TECHNICAL MEETINGS are referred to the Volume 8, No. 4, (April 1973) issue of SIGPLAN *Notices* in which the "Suggested Guidelines for SIGPLAN Technical Meetings" were published.

# correspondence




MENLO PARK, CA ■ 94025
P. O. BOX 310 ■

Greetings,

There is a viable alternative to the problems raised by Bill Gates* in his irate letter to computer hobbyists concerning "ripping off" software. When software is free, or so inexpensive that it's easier to pay for it than to duplicate it, then it won't be "stolen."

Example: There are at least five versions of Tiny BASIC up and running on at least three processors. A cassette containing Tiny BASIC for the Intel 8080 is available for five bucks. A version for the Motorola and AMI 6800 also costs $5, including complete user documentation. If the price is still too high, complete user documentation and implementation details for one of the 8080 versions has already been published. This includes complete annotated source code. Anyone is welcome to retype it and reassemble it. No one will yell, "thief." All details of a second version will be published before the end of April. Several more versions will be published shortly thereafter, including a cross-assembled version created using the macro facilities of the IBM 360 Assembler. Versions are expected shortly thereafter for the MOS Technology 6502, and Signetics 2650. Note: Tiny BASIC is, essentially, BASIC sans array and floating-point operations, although one of the versions has array operations, and another uses a calculator chip to obtain floating-point capabilities. It is explicitly designed for minimal memory micros.

Example: Gary Kildall, who built the PL/M compiler for Intel and the PLuS compiler for the Signetics 2650, is making an entire floppy-disc operating system available. He plans to sell a disc and complete documentation for not much more than what it would cost to duplicate them.

Example: A complete alpha-numeric music system, including amplitude control, has been designed and made available. The documentation costs only $2, including complete schematics for the minimal hardware that must be added.

Information on all of these systems–and much more–is being published in a new, reference journal for home computer users (and anyone else interested in micros), *Dr. Dobb's Journal of Computer Calisthenics & Orthodontia*. The *Journal* is publishing all available details. For instance, the first issue contained: complete design details for Tiny BASIC, complete user documentation for the first 8080 version, complete details for using a calculator chip to obtain mathematical and floating-point functions, and a 16-bit, binary-to-decimal conversion routine.

The second issue included: complete implementation details and annotated source code for the first version of tiny BASIC, complete documentation and source code for a simple music program for Altair 8800s, design notes on a forthcoming high-level language for 8008/8080s, two articles on a $1K phoneme generator kit for micros that allows unlimited English speech synthesis, and a quick note on the 6800 version of Tiny BASIC.

The third issue will include complete details and code for the second 8080 Tiny BASIC which includes 1-D arrays, a simple debugger for the 6502, a keyboard loader for octal code, details of a contest to generate public-domain graphics software for CroMemCo's TV Dazzler, and much more. The *Journal* is also reprinting carefully selected, good stuff from the growing multitude of computer club newsletters. Additionally, it is publishing complete indices to all major computer hobbyist publications and selected articles from other publications, lists of hobbyists and their equipment, used equipment sources, clubs and organizations, computer stores and distributors, etc. Finally, it is actively pursuing a consumer advocacy role relative to the home computer user.

The point is that all of this information–systems software, design notes, schematics, etc–is being made available for little more than the cost of reproduction. The *Journal* came into being, explicitly to aid creation and distribution of that information. In some ways, it creates a sort of manufacturer-independent user's group.

It is reasonable to expect that free and inexpensive software will become increasingly available to and through the hobbyists' community. This is true, in spite of the failure of such SHAREing in the business and industrial communities.

1. Hobbyists are developing home-grown hardware and software, just for the fun of it. Since it's "fun" rather than "work," they have shown a great willingness to share and distribute what they develop. This is not

*See Minicomputer News, March 11 for Gate's letter.

an unknown phenomenon. It is the usual practice in most other hobby environments, and is certainly true in the academic environment.

2. As with the industrial mini and micro markets, hobbyists have learned to be wary of purchasing hardware from manufacturers who provide no software support. Through common sense, and by observing Mr. Gates' experience, those who wish to sell software for significant sums of money must realize that there is only one group that can practically be expected to pay for it: the hardware manufacturers. They need it to enhance their products in a highly competitive marketplace.

3. Concerning quality: A significant minority of computer hobbyists are also experienced computer professionals. It's their (our) play as well as work. The competency level is more than sufficient for the design and implementation of excellent systems software.

4. Finally, the approach used in producing the Tiny BASICs will be continued and expanded, a sort of modified Chief Programmer Team approach: An experienced pro does the overall design and outlines the implementation strategy (via the *Journal* and other hobbyist publications). Following those directions, the more experienced amateurs do the necessary hack-work (exciting to them, but drudgery for the "old pro"). Since it is a symbiotic effort, the implementors are almost certain to share their work with the designers, and hence, with the larger community of home computer users.

It's amazing how much "good stuff" becomes available when the producers think of their labor as "play" instead of "work." All who wish to do so are invited to join with the publishers of *Dr. Dobb's Journal* in the pursuit of realizable fantasies.

Jim C. Warren, Jr., Editor
*Dr. Dobb's Journal of Computer Calisthenics & Orthodontia*

P.S. *Dr. Dobb's Journal* is published by People's Computer Company, Box 310, Menlo Park CA 94025. Subscriptions are $10 per year. PCC is an established publisher of *PCC* newspaper (devoted to computers in education, and computer games), and of numerous computer books.

**North Staffordshire Polytechnic Department of Computing**

Director J. F. Dickenson BSc(Eng), PhD, CEng, FIMechE

Head of Department
H. L. W. Jackson PhD, MSc, AInstP, DipEd, FIMA, FBCS

Computer Centre
North Staffordshire Polytechnic
Blackheath Lane, Stafford
Telephone 0785 53511

24th March, 1976

To the editor:

I have just read the first paper in the Proceedings of the ACM SIGMINI/SIGPLAN Interface Meeting on Programming Systems in the Small Processor Environment in the April (1976) Edition of SIGPLAN Notices and I found somewhat serious omissions in the languages mentioned, namely RTL/2 and Coral 66. Both of these are currently available in the U.K.

Coral 66 has the appearance of being a subset of Algol 60 with a few real-time features added. Computers purchased by the U.K. Ministry of Defence are supposed to have a Coral 66 compiler available. It has been reported that the U.S. Ministry of Defence are interested in Coral 66 also.

RTL/2 is a newer language and has a more modern appearance. It was developed by I.C.I. for its own purposes and is now marketed by S.P.L. International (London).

Both languages have several compilers currently available and are quite widely used in the U.K.

Further details can be found in the references given:

Barnes, J.G.P., Real-time languages for Process Control. Comp.J. Vol. 15, No. 1, Feb. 1972, pp 15-17.

Software for Control, I.E.E. Publication.

Yours faithfully,

John Hopkinson

**COMPUTER SCIENCES CORPORATION**
SYSTEM SCIENCES DIVISION (301) 589-1545
8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

April 13, 1976

**To the editor:**

The continuing publication of articles on the form and style of programming languages has reached a point where some ground rules are, I think, in order.

1. In order to compare the various proposed alternative forms, or even alternative languages, the common practice of typesetting should be avoided. Despite its undeniable advantages when used with prose, poetry or algorithms, the use of italics, boldface, underlining, etc., in an language example obscures the subject and makes comparison difficult. Lower case, italics, boldface, underlining are not generally available for compilation or preprocessor listings for any language or machine. The authors of papers in this area should restrict their examples to copies of actual computer-printed outputs. They should avoid examples on coding sheets or ones "enhanced" by the printers. In this way their audience can more directly compare and evaluate language suggestions in the form most likely to appear in practice.

2. The selection of examples should not be limited to the various structured (and unstructured) control forms. All useful languages have I/O commands of some sort, and probably other forms which are commonly used. The examples selected should show all the basic features of a language form, not just its most (or least) elegant forms.

Very truly yours,

Ben Consilvio

# NOTICES

X3J3 'FAVORABLY DISPOSED' TOWARD IF-THEN-ELSE IN CURRENT REVISION

In an unofficial action, the Fortran Standards Committee (X3J3) expressed its inclination in favor of a proposal to incorporate an IF-THEN-Else structure into the current revision of Standard Fortran. This was expressed by an informal vote during the X3J3 meeting at NBS Gaithersburg Md (March 31 - April 2), and was no doubt influenced by public response at the Fortran Forum earlier in the week. A number of members of the committee felt concerned because of the "lateness of the hour," but the prevailing sentiment was that the importance of this feature warranted giving some extra consideration to its adoption. The proposal, which was presented by Walt Brainerd and is of course subject to further revision prior to its possible formal adoption, was essentially as given below. Public comment on this proposal is solicited, and should be directed to Lloyd W. Campbell, BRL-CSD Bldg 328, Aberdeen Proving Ground, MD 21005.

FORTHCOMING MEETINGS OF X3J3

Meetings of X3J3 are open to the public; however, facilities are limited. Non-members who wish to attend a meeting as observers should inquire concerning arrangements from Martin Greenfield, MS 831a, Honeywell Info Sys, 300 Concord Rd, Billerica MA 01821.

The next meeting is planned for Idaho Falls, Idaho, during the week of July 12.

The following meeting is tentatively scheduled for Boulder, Colorado, during the week of September 19.

Another meeting has tentatively been scheduled for Philadelphia, during the week of November 14.

It is expected that the principal business at these meetings will be deciding how to respond to comments received from the Public during the Public Review and Comment period which ends on July 4, 1976. The committee is required to give serious consideration to all comments received.

Fortran Tutorial at ACM 76

20-22 October 1976: ACM 1976 Annual Conference, Houston. This meeting will include a tutorial on "Highlights of the New Fortran Standard." Participants: Frank Engel, Walt Brainerd, Tom Gibson, Loren Meissner.

Module protection already working in SIMULA.

The Simula Development Group (with representatives for the existing 10 Simula implementations) has accepted in September 1975 a construct for module protection similar to the proposal by C.H. Lindsey in AB 39 p 20. Attributes of a module can be made inaccessible by means of the hidden and the protected specification. The use of two specifications makes it possible to let certain attributes be accessible only inside a module, other attributes accessible within a system of cooperating modules, other attributes accessible wherever the module itself is accessible.

See further Simula Newsletter, No. 1, 1976, or (in more detail) DECsystem-10 Simula Gazette, No. 3, Vol. 2, 1976 (Can be ordered from Section 142, Swedish National Defense Research Institute, S-104 50 Stockholm 80, Sweden).

The Module protection scheme in SIMULA is already implemented and available in Release 3 of the DECsystem-10 Simula system. The system is available for a price of about $ 100 from the Swedish National Defense Research Institute, Section 142, S-104 50 Stockholm 80, Sweden.

The DECsystem-10 Simula system implements the full Simula language (comparable in power to Algol 68 or PL/I, and based on the same ideas, but closer to Algol 60). The system is especially aimed at conversational applications, allowing one or more conversational terminals to be connected to an executing program, and the system has a conversational debugging system where the user at the conversational terminal can set breakpoints and query about internal data values.

A Codasyl type DBMS system entirely written in Simula, for use by Simula programmers or at a user terminal, is distributed with the DECsystem-10 Simula system.

A system for using separately compiled modules in a way which in no way endangers the security of the language and which does not incur any extra run time overhead is also available with the DECsystem-10 Simula system.

Jacob Palme

# news of past events

## SIGPLAN NOTICES

Recent Meetings of the San Francisco Peninsula SIGPLAN Chapter

March 4th SIGPLAN Meeting

REASSESSING THE VALIDITY OF CREATING A PROGRAM TO PERFORM A TASK
John Peers, President, Logical Computer Corp.

Time & Date: 7:30 pm (on time), March 4, 1976 (Thu)
Location: R-om 5M, Hewlett-Packard facility, 1501 Page Mill Road, Palo Alto, CA

Note: All meetings are open to all interested individuals.

It is clear from the progress in semiconductors that the prices of computers is going to come tumbling almost as rapidly as it has for hand-held calculators. The remaining problem, therefore, of using these inexpensive machines will be software. The key element in the software problem is the degree of human involvement; the more human involvement, the more expensive the end-product. The problem will remain difficult to solve until we readdress the problem of how to cause a computer, which is a single subset of a generic group of machines called logical machines, to perform a task in ways which ordinary users find simple. It is proposed, therefore, that the concept of compilation and assembly should be dispensed with and a way found of direct translation from the typed input into usable meaning as far as the system is concerned. It is further suggested that this is an intermediate step to a full-dialogue (that is, audio) response system.
One small step using this approach will be discussed.

Mr. Peers, at 34, is President of Logical Machine Corp., 887-A Mitten Rd., Burlingame, CA 94010, (415)692-4970. Prior to this, he was Chairman of Allied Business Systems in London, England. Preceeding that, he was with Unidata (now Ventek), and Interscan. He was orginally trained as a chemist, and has worked on semiconductor developments.

Editor's note: All you compiler and programming language sophisticates, out there....before you poo-poo Mr. Peer's talk as being too Buck Roger-ish or heretical, (1) think back over the brief history of our discipline, and (2) note that Mr. Peers already has a product in the field that implements his views... and it's selling like hotcakes. Furthermore, you should know that our own Dennis Allison spent an entire afternoon talking with Mr. Peers and examining his product, and has pronounced both to be Most Interesting. And, we all know that Dennis does not award such judgements casually or without good cause.
So, come on over and see what he has to say.

A HIGH LEVEL LANGUAGE EXTENSION FOR DYNAMIC COMPUTER GRAPHICS
Gregory F. Pfister, University of California, Berkeley

Time & Date: 7:30 pm (on time), February 19, 1976 (Thu)
Location: Room 5M, Hewlett-Packard facility, 1501 Page Mill Road, Palo Alto

Note: All meetings are open to all interested individuals.

DALI (Display Algorithm Language Interpreter) is a special purpose high level programming language extension for the creation and control of dynamic pictures which exhibit complex static and dynamic interactions among their elements. DALI allows complex organizations of interpolated ("smooth") change, discrete ("sudden") change, and change to the structure of a picture to be generated in a modular way, in the sense that picture elements determine their own behavior and hence manner of change in a local manner. DALI pictures are composed of elements called picture modules. These are analogous to processes or procedural activations, and contain arbitrary event-driven procedures called daemons. Daemons, which are user-written, run under the control of scheduling rules based on the functional dependence of daemons on one another. These rules result in smooth inter-daemon (process) communication and cooperation with no implicit or explicit reference to semaphores or other synchronication primitives.

Gregory F. Pfister received the S.B. and M.S. degrees in Electrical Engineering and Ph.D. degree in Computer Science from Massachusetts Institute of Technology in 1967, 1969, and 1974, respectively. In 1969, he was employed by Evans and Sutherland Computer Corp., doing architectual and logic design. In 1974-75, he was employed by IBM Advanced Systems Development and System Communications Divisions, programming interactive graphics systems and applications. He is presently an Assistant Professor in the Department of Electrical Engineering and Computer Science at the University of California at Berkeley.

## A HIGH-LEVEL DATA MANIPULATION LANGUAGE FOR HIERARCHICAL DATA STRUCTURES

Barry Housel, IBM

Time & Date: 7:30 pm (on time)
April 1, 1976 (Thursday)
Location: Room 5M, Hewlett-Packard,
1501 Page Mill Road, Palo Alto

The hierarchical view of data is used by many applications and users. In particular, some of these applications require complex data manipulation which, heretofore, has been delt with procedurally. In this light, a non procedural language, CONVERT, originally developed for data translation, is proposed as a high-level Data Base Management System interface. CONVERT is meant to provide users with a tool for performing complex data manipulation and query of hierarchical data abstractions referred to as "Forms." This talk will describe the language and present an illustrative example.

Barry Housel has worked for IBM for the past 11 years. He has been involved in scientific and systems research and development in both technical and managerial capacities. During that time, he received an M.S. (Stanford, 1968) and a Ph.D. (Purdue, 1973) in Computer Science. Since 1973, Barry has been with the Computer Science Department of IBM Research in San Jose. He has been working in Data Base technology, particularly in Data Base conversion and restructuring. Other areas of technical interest include programming languages, graphics, and systems design.

# official sigplan messages

*A Note from Your Treasurer*

Financially, SIGPLAN is indeed very healthy. Over the past years, conservative budgeting, prudent SIGPLAN officers, and economies of scale have allowed SIGPLAN to develop a rather large surplus. As of July 30, 1975, there were 5,991 subscribers to SIGPLAN Notices and SIGPLAN had a surplus over $36,000.

Table 1 gives a summary of the actual SIGPLAN budget for the fiscal year 1974-75 and a projected budget for the current fiscal year 1975-76. A few comments on some of the line items are in order.

A major source of income is in membership dues. The major source of expenses is the publication, printing, and mailing of SIGPLAN Notices. SIGPLAN has a small income from special publication sales and from conferences. SIGPLAN generally has a number of small expenses for office supplies, data processing, and an allocation to ACM headquarters.

For the projected fiscal year 1975-76 budget, there are two unusual items. There is an amount for travel, which was originally intended for a proposed national professional development seminar series. There's also $3,000 listed under "Other" for use as honoraria for the intended speakers. It now appears unlikely that these amounts will be expended as no such seminar series has been set up.

Besides the unlikelihood of expenses for some of the line items for the 1975-76 budget, ACM headquarters expects rather conservative figures for income and expenses. For example, a conservative estimate of growth was used when the FY76 budget was projected, and this caused the projected dues to be lower than the FY75 actual accounting. This worst condition budget can sometimes hide the expected average surplus at the end of the year. It is not unreasonable to expect that at the end of this fiscal year, SIGPLAN will have a surplus of at least $36,000.

We the officers, as well as members of the executive board, are wondering what to do with the surplus. Launch a new periodical? Sponsor a National lecturer series? Sponsor professional development seminars? We are not sure of the best course of action and would welcome any ideas from the membership. If you have the time, please write any of us and let us know what you think.

Henry F. Ledgard

Table 1. Summary SIGPLAN Budget for Fiscal Years 1974-75 and 1975-76

*(Note: All figures are in dollars)*

| | FY 1975 *(Actual)* | FY 1976 *(Budgeted)* |
|---|---|---|
| *INCOME:* | | |
| Dues | $53,686 | $47,074 |
| Publication Sales | 7,546 | 6,000 |
| Conference Income | 4,523 | 2,000 |
| TOTAL INCOME: | 65,755 | 55,074 |
| *EXPENSES:* | | |
| Travel *(mainly speakers)* | 0 | 4,800 |
| Temporary Help/Office Supplies *(mainly typing)* | 2,393 | 2,440 |
| Telephone | 619 | 880 |
| Misc. Postage *(mainly election ballots)* | 574 | 0 |
| Misc. Printing | 1,118 | 1,083 |
| Publications Printing *(mainly newsletter)* | 34,681 | 33,455 |
| Handling and Mailing *(mainly newsletter)* | 4,251 | 2,995 |
| Shipping/Freight/Postage *(mainly bulk mailing of newsletter)* | 7,661 | 5,606 |
| Data Processing | 1,393 | 1,317 |
| Local SIG Chapter Subsidies | 0 | 400 |
| Meeting Expenses | 32 | 300 |
| Headquarters Allocation | 3,260 | 4,400 |
| Other | 0 | 3,000 |
| Contingency Funds | 0 | 6,000 |
| TOTAL EXPENSES: | 55,987 | 66,597 |
| *SUMMARY:* | | |
| Net Profit *(loss)* | 9,768 | (11,522) |
| Surplus *(beginning of FY)* | 27,181 | 36,949 |
| Surplus *(end of FY)* | 36,949 | 25,427 |
| Reserve Expected by ACM *(Expenses ÷ 4)* | 13,997 | 16,649 |

# publications

compiled by
Ronald L. Lancaster
Computer Science Department
Bowling Green State University
Bowling Green, Ohio

**An Example of Hierarchical Design and Proof.**
Jay M. Spitzen, Karl N. Levitt, and Lawrence Robinson.
Stanford Research Inst Menlo Park Calif Mar 76, 28p SRI-4079-TR-2
**AD-A021 574/9WC PC$4.00/MF$2.25**

Hierarchical programming is being increasingly recognized as helpful in the construction of large programs. Users of hierarchical techniques claim or predict substantial increases in productivity and in the reliability of the programs produced. In this paper the authors describe a formal method for hierarchical program specification, implementation, and proof. The authors apply this method to a significant list processing problem and also discuss a number of extensions to current programming languages that ease hierarchical program design and proof.

**Interactive Generation of Object Models with a Manipulator.**
David D. Grossman, and Russell H. Taylor.
Stanford Univ Calif Dept of Computer Science Dec 75, 32p STAN-CS-75-536, AIM-274
**AD-A020 942/9WC PC$4.00/MF$2.25**

Manipulator programs in a high level language consist of manipulation procedures and object model declarations. As higher level languages are developed, the procedures will shrink while the declarations will grow. This trend makes it desirable to develop means for automating the generation of these declarations. A system is proposed which would permit users to specify certain object models interactively, using the manipulator itself as a measuring tool in three dimensions. A preliminary version of the system has been tested.

**The AUGMENT Precompiler II. Technical Documentation.**
F. D. Crary.
Wisconsin Univ Madison Mathematics Research Center Oct 75, 142p MRC-TSR-1470
**AD-A020 198/8WC PC$6.00/MF$2.25**

AUGMENT allows the FORTRAN programmer to easily use packages implementing nonstandard data types and operations. This report describes the internal structure and logic of AUGMENT. User documentation is contained in a previous report.

**MPL Programmer's Manual.**
Martin I. Wolfe, and Mark B. Schwartz.
Army Electronics Command Fort Monmouth N J Nov 75, 42p ECOM-4371
**AD-A020 456/0WC PC$4.00/MF$2.25**

A Modified Programming Language (MPL) similar to XPL, has been implemented for an advanced Data Processor Module (DPM). The features of MPL are described and illustrated. Syntax of constructs in the language are shown in BNF. Declaration statements, data types, assignments operations, expressions, blocks, scope of variables, flow of control, procedures, I/O, and compiler options and statistics are explained.

**DPM/MPL (Burroughs Data Processor Module/Modified Programmers Language) Compiler Manual.**
Martin I. Wolfe, and Mark B. Schwartz.
Army Electronics Command Fort Monmouth N J Nov 75, 86p ECOM-4373
**AD-A020 538/5WC PC$5.00/MF$2.25**

Modified Programmers Language (MPL), a programming language similar to XPL, has been implemented for the Burroughs Data Processor Module (DPM). The DPM/MPL compiler is described and illustrated, including the parsing and reducing algorithms, syntactic and semantic stacks, code buffer, storage allocation scheme, and procedures. The semantic routines of the compiler that deal with the MPL language features are also explained. A listing of the compiler in source code is provided.

**Interdata 7/16 DDS ANS Subset COBOL Release 1.15.**
Federal Cobol Compiler Testing Service Washington D C 11 Feb 76, 53p CCVS74 - VSR115
**AD-A020 551/8WC PC$4.50/MF$2.25**

This Validation Summary Report (VSR) provides a consolidated summary of the results obtained from the validation of the subject compiler. The VSR is made up of several sections showing the discrepancies found. These include an overview of the validation which lists all categories of discrepancies by Federal level module; a section relating the categories of discrepancies to each of the Federal levels of the language; and a detailed listing of discrepancies together with the tests which were failed.

Note: Papers with an NTIS report number are available from National Technical Information Service, Springfield, Virginia 22151, at the prepaid prices indicated for paper copy (PC) and microfiche (MF).

**SLAM - Simulation Language for Analogue Modeling.**
G. F. Tomlins.
British Steel Corp., Sheffield (England). Information Services. Dec 75, 12p CEL/CE/TN-34/75
**PB-249 305/4WC** PC$3.50/MF$2.25

SLAM is a high level language, based on FORTRAN, which can provide a framework for the simulation of continuous systems on a digital computer. Continuous system simulation is a process, in which the representative mathematical, equations are solved simultaneously. The digital computer, which is basically a discrete machine is made to model such systems, by holding the independent variable, conventionally time, constant while the other variables in the equation are calculated and updated. This note describes the macros available and how SLAM programs can more easily be run from MOP terminals using these macros.

**Hal/SM Language Specification.**
G. P. W. Williams, Jr., and C. Ross.
M&S Computing, Inc., Huntsville, Ala. 21 Nov 75, 253p
NASA-CR-144100, REPT-75-0043
**N76-14845/1WC** PC$9.00/MF$2.25

A programming language is presented for the flight software of the NASA Space Shuttle program. It is intended to satisfy virtually all of the flight software requirements of the space shuttle. To achieve this, it incorporates a wide range of features, including applications-oriented data types and organizations, real time control mechanisms, and constructs for systems programming tasks. It is a higher order language designed to allow programmers, analysts, and engineers to communicate with the computer in a form approximating natural mathematical expression. Parts of the English language are combined with standard notation to provide a tool that readily encourages programming without demanding computer hardware expertise. Block diagrams and flow charts are included. The semantics of the language is discussed. (Author)

**A Study of High Level Language Features. Volume 1.**
John B. Goodenough, and Lawrence H. Shafer.
Softech Inc Waltham Mass Feb 76, 149p 1021-10-Vol-1, ECOM-75-0373-F-Vol-1
**AD-A021 206/8WC** PC$6.00/MF$2.25

The two volumes of this report present a comprehensive survey and analysis of high level programming language features, together with a selection of features needed to program Army tactical computer-based systems. The report is devoted mainly to detailed analyses of the strengths and weaknesses of language features supported by a wide variety of general purpose languages. This volume of the report describes the approach taken to this study, and in an Appendix, provides an index to the detailed feature analyses contained in Volume 2. Two other Appendices contain surveys of language requirements for process handling and input/output. A fourth Appendix documents the need for certain application-dependent language features, based on a study of actual tactical system specifications and programs.

**A Study of High Level Language Features. Volume 2. Detailed Language Features Analyses.**
John B. Goodenough, and Lawrence H. Shafer.
Softech Inc Waltham Mass Feb 76, 389p 1021-10-Vol-2, ECOM-75-0373-F-Vol-2
**AD-A021 207/6WC** PC$10.75/MF$2.25

The two volumes of this report present a comprehensive survey and analysis of high level programming language features, together with a selection of features needed to program Army tactical computer-based systems. This volume of the report contains detailed analyses of the strengths and weaknesses of language features found in a wide variety of programming languages. Although not all features are analyzed with equal thoroughness and although certain completeness and consistency checks remain to be done, the analyses begin to capture what has been learned about the merits and deficiencies of existing language features. This information can be used to help guide the selection or design of a language for programming tactical or other kinds of systems.

**Graphics Interpreter Language.**
J. B. Stynes.
Illinois Univ., Urbana Dept. of Computer Science. May 75, 140p UIUCDCS-R-75-710
**COO-2383-0019** PC$6.00/MF$2.25

A "high level" machine language for efficient implementation of a multiterminal graphics system support package is described. The type of support package for which this is intended is an interactive network design system using relatively static displays, but making much use of user-defined and system menus for command and control. The language is executed via an interpreter on a PDP-8 with disk storage. (21 figures).

**Structured Programming Series. Volume XIII. Final Report.**
John J. Naughton, and Ronald L. Smith.
IBM Federal Systems Center Gaithersburg Md 15 Jul 75, 73p RADC-TR-74-300-Vol-13
**AD-A020 858/7WC** PC$4.50/MF$2.25

This volume summarizes the tasks and products resulting from the structured programming studies performed under contract. It contains contract conclusions and recommendations. It also contains scope of work, task activities and results for each of the fourteen tasks. The report also relates structured programming technology to the phases of the software development cycle.

**Table Producing Language (TPL), Version 3.5.**
Peter B. Stevens.
Bureau of Labor Statistics, Washington, D.C. Office of Systems and Standards. Jan 76, 1 reel mag tape BLS/DF-76/001 Source tape is in BINARY character set. Character set restricts preparation to 9 track one-half inch tape only. Identify recording mode by specifying density only. Call NTIS Computer Products if you have questions. Price includes documentation, PB-249 834.
**PB-249 833/5WC** Mag Tape $200.00;Foreign $250.00

A national statistical agency such as the Bureau of Labor Statistics usually produces the results of its surveys in tabular form at the cost of a substantial portion of its data processing resources. The tables are often quite complex with common requirements for cross tabulation of irregular patterns of hierarchical data files with many intermediate levels of summarization, percentage distributions, measures of statistical precision, and intricate publication formats. The Table Producing Language (TPL) has been designed and implemented as a general tool for producing statistical tables. The syntax of the language allows the user to construct expressions representing the desired tables and auxiliary operations. The expressions are problem oriented, nonprocedural, compact, and written with English-like nomenclature. Together with a Codebook, they allow the specification of complex data reduction, cross tabulation, and table generation operations. Users may be research economists, statisticians and other social scientists who need not have experience with conventional computer programming languages, as well as computer oriented staff concerned with production tasks.

## A GUIDE TO TSO SPEAKEASY

by

S. Cohen, J. Fink, F. J. D. Serduke and H. Z. Kriloff

### ABSTRACT

Speakeasy is a computer programming system that is powerful yet easily learned. This book is intended as an introductory guide to this system. It assumes that the reader is a complete novice, both with regard to Speakeasy and to the use of TSO. This book is divided into several sections in which the processor is used in a desk-calculator mode to demonstrate its capabilities for matrix operations, for calculations in calculus and for operations involving logical and relational tests. The program mode and the use of available teaching aids is also described.

**Available from** N.T.I.S. PC $5.45, MF $2.25 as document ANL-75-44

A Reminder for Language Designers
Frederic Richard
Henry F. Ledgard
University of Massachusetts, Amherst. Dept. of Computer Science and Information. March 76, COINS Technical Report 76-3
$3.00

A readable programming language is easy to learn and facilitates the validation and the maintenance of programs. This paper proposes eleven design principles for the development of readable, high level languages. Each principle is backed up by a discussion and several examples. Among the issues discussed are the limitation of the overall complexity of a language, the design of function and procedure facilities, and the correspondence between syntax and semantics.

A User's Guide to The PASCAL Assistant
H.F.Ledgard
Andrew Singer
Jon Hueras
University of Massachusetts, Amherst. Dept. of Computer Science and Information. March 76
$2.00

"So that even now the machines will only serve on condition of being served, and that too upon their own terms; the moment their terms are not complied with, they jib, and either smash both themselves and all whom they can reach, or turn churlish and refuse to work at all. How many men at this hour are living in a state of bondage to the machines?"
--Samuel Butler
quoting a scholar from Erewhon, 1872.

Over the past year and a half, there has been considerable activity conducted to create a highly human-engineered interactive language for the writing and preparation of programs. Design objectives have included:

a. Smallness of scale
b. A pleasant user interface
c. and its support by its formal standard.

PROPOSED ANS X3.9 FORTRAN LANGUAGE REVISION

Fortran Development Newsletter

Volume 1, Number 6; January 1976

For additional copies of this report,
or for copies of other issues of
For-Word, please send inquiriesto

Loren P. Meissner
50-B 3239
Lawrence Berkeley Laboratory
Berkeley CA 94720

# Basic statement times for ALGOL 60

by

B.A. Wichmann

Abstract. This report contains working details of a method of evaluating the processing speed of an ALGOL system using the times of execution for basic statements. Notes on all the times obtained to date (10/10/73) are also included. A more detailed analysis has been added since the production of a similiar report(NAC15).

Available as NPL Report NAC-42 from the National Physical Laboratory, Teddington, Middlesex, England.