With this first issue of PAX, I am dissolving the SR-52 Users Club, and announcing a successor activity. Back issues of 52-NOTES and membership lists will continue to be made available upon request, but no new membership lists or address corrections will be issued. Prior members and newcomers alike should view their relationship to PAX as "participants", and are invited to contribute inputs and to maintain money balances to cover a few issues ahead at the continuing rate of $1 per issue (1.67 $us abroad).

PAX succeeds 52-NOTES as a non-copyrighted non-profit publication built upon carefully-edited inputs from able readers, but with a change in scope. PAX aims to fill a void still largely ignored by the many computer/calculator-oriented publications: The analysis and evaluation of widely-used personal-computing software currently in the public domain, for which good supporting documentation is not known to exist. I see two motivations for such an effort: 1) It is challenging to find out how non-trivial code structures work, and 2) It is rewarding to significantly enhance the value of important software to the user, bringing limitations to light, and facilitating the generation of bug-free custom modifications.

What I see filling the pages of PAX would properly have preceded the publication and use of the associated code. But there is a reluctance even (especially?) among seasoned programmers to take the time and to make the effort during the software development process to formalize requirements, approaches, and limitations. Often in the commercial arena, where supporting documentation is a contractual requirement, it is generated at the end of the software development process (after key developers may have vanished) as a necessary but detestable chore, so it is not surprising that much of the documentation that does exist is hardly worth the paper it is printed on.

In cases where good documentation does exist, but is withheld from the public, I see nothing wrong with independently generating analyses and evaluations from available code. I will endeavor to detect outright plagiarism, and to reject any inputs which appear to violate copyrights.

As a start, here are some guidelines for potential contributors to observe:

1. Choose programs widely used across many disciplines, sufficiently complicated that cursory inspection does not reveal workings and attributes.

PAX is a non-profit newsletter edited and published by Richard G Venderburgh in Dayton, Ohio. Participation in the analysis and evaluation of computer/calculator programs is open to any interested person. Back issues (including 34 issues of the predecessor 52-NOTES) are available at $1 each (1.67 abroad $us), and new participants are encouraged to contribute for a few issues ahead (at the same rates), to stay current.
2. Obtain (preferably from the source) all available documentation, and determine that additional material would significantly enhance for the user the value of a selected program.

3. Organize your analysis to cover a) What the program appears to do, b) Where and how it does it, and c) What limitations and pitfalls appear to exist.

4. Construct algorithms in typewritable English, choosing a level of detail which conveys the most information. This is not easy to do well, and is apt to require several tries. For example, one could easily convert the 52-NOTES v3n1lp5 Euclid's Algorithm code into English along the lines of: 1. Zero the T register, store the first integer in R1, store the second in R2; 2. Divide r1 by r2, go to step 3 if the fractional part of the result equals t, else put the contents of R2 into R1, and the remainder times r2 into R2 and go to step 2; 3. Display r2, stop. But this doesn't show how the algorithm works as efficiently or machine-independently as: 1. Divide m by n and let r be the remainder. 2. If r=0, the algorithm terminates; n is the answer. 3. Set m to n, n to r, and go back to step 1. (See Knuth's Vol I The Art of Computer Programming page 2). I should note here that "set m to n" means the value of m become value of n, with n unchanged.

5. Be comprehensive in citing references, and acknowledge all credits where due.

6. Although the primary reward for your effort is the satisfaction of having your creations recognized and used by your peers, I may decide to award free PAX issues or other material goodies in some cases.

7. Contributed material must be free from proprietary restriction, and contributors' names, addresses and/or phone numbers will be published unless withholding them is specifically requested.

Good candidate programs will come from a variety of categories: the larger and more complicated TI and HP PPC ROM and program exchange programs, microcomputer operating systems and utility routines (editors, assemblers, interpreters, ...), and the most popular non-trivial microcomputer applications programs, come to mind. In some cases, merely identifying the starting addresses of key entry points in complicated multifunction programs (the "where" in 3b above) will be a good beginning, enabling others to proceed on with detailed analyses of particular sections.

**TRS-80 ROM MAP**

I'll lead off with just such a mapping of the TRS-80 microcomputer BASIC Level II processing. Here is an example of a lot of ROM code used in a lot of machines, for which neither the manufacturer (Radio Shack) nor the program developer (Microsoft) will provide documentation. The user can peer into the ROM's more than 12000 8-bit cells (bytes) and attempt to manually translate the code, or he can use one of the currently available disassembler programs to automatically translate the code into assembly language mnemonics, but he is still faced with the formidable task of separating the true instructions from the data (which look like instructions when disassembled), and then determine what processing is done where, before he can even begin to analyze and evaluate what's there.

Fortunately, some of the processing partitioning information has already surfaced, and it appears from the disjointed structure that Microsoft may have intentionally scrambled things! Input leading to the
following observations came from "Software Technical Manual", a publication of Houston Microcomputer Technologies, 5313 Bissonnet, Bellaire TX 77401, augmented by some exploring on my own. All numbers are in hex, and I assume that the reader from this point on is familiar with BASIC Level II terms and other more-or-less universal computer terminology. Although the nonTRS-80 user may not find this memory-map information very interesting, the analyses which I hope will follow should be of general interest to serious programmers of any machine, and I will try to present such material in machine-independent terms.

An ASCII dump of locations 1650-18F reveals a continuous string of characters comprising the set of BASIC and Disk BASIC keywords. Keyword delimiting is accomplished by the hex addition of 80 to the ASCII code for the first letter of each word. The processing start address corresponding to each keyword is either in one of two address-tables, or is found by token comparison. The token for a keyword is a one-byte hex number formed by adding 80 to the integer count corresponding to its position in the keyword table. The keywords END, FOR, ...NEW have starting addresses in an address table at 1822-1899; TAB(, TO, ...left bracket (less than) start addresses are found by scattered comparisons, some of which are: TAB( 2137, FN 4155, USING 2CBD, VARPTR 24EB, USR 27F3, ERL 24DD, ERR 24CF, STRING$ 2A2F, INSTR 419D, POINT 0132, TIME 4176, MEM 27C9, inkey$ 019D, NOT 2504; SGN, INT, ... MID$ start at addresses in a table at 1608-164F. Locations 18C9-18F6 comprise a table of error codes. Note that processing for Disk BASIC keywords is vectored through RAM addresses in the 4000+ region of memory.

A brief tour through ROM shows Rerunlations at locations 0-38; most I/O processing and RAM initialization 49-707; math functions 708-1607; BASIC tables, text interpretation, and processing 1608-2FF8. Some key parameters (for a non-Disk BASIC configuration) are stored as follows: start address of USR routine is stored at 408E, 408f; pointer to the beginning of BASIC program text is at 40A4, 40A5; number type flag is at 40AF; pointer to the top of BASIC RAM is at 40B1, 40B2; and the I/O scratch area for math processing is at 411D-412E.

So here is a start at what's there in TRS-80 Level II ROM. Choose any part that looks interesting, and send me your analyses/evaluations.

SARGON: ONE OF THE BETTER MICROCOMPUTER_APPLICATIONS PROGRAMS

SARGON is the best micro-class chessplayer program I've yet come across, having won the 1978 West Coast Computer Faire chess match, and appearing to play at least as well as BORIS (probably the best of the special-purpose personal chessplayers).

Here is a complicated assembly-language program for which some documentation is available to the public: "SARGON A COMPUTER CHESS PROGRAM" by Dan and Kathe Spracklen, Hayden Book Co, Rochelle Park NJ, 114 pp, $14.95. The text begins with a general description of the interaction among the 50 or so subroutines, then proceeds with a complete symbolic listing of the program, along with a data map, all of which when assembled and loaded into a Z-80 based machine occupies almost 9000 bytes of memory. The program as published is written in 8080 mnemonics for TDL Macro assembly to Z-80 code, with machine-dependent sections targeted to the Jupiter III microcomputer.

Although just about every instruction is annotated with a comment, there is an information gap between the general description and the comments, which if filled would provide the user with fairly complete docu-
mentation. So here is another challenge, involving a variety of interesting processing requirements, from dynamic graphics control to move making and record keeping, and board configuration evaluation.

I'm reminded by sargon that being able to make bug-free modifications is an especially good reason for having good documentation when there is a bug in the original program. It turns out that as published, when stalemated, SARGON moves into check! After a fair amount of code-scanning, I tracked the problem to the PNDMOV (find move) routine, which although it tests for stalemate and even sets an identifier, doesn't act on it. A simple conditional branch address change fixes things up, but I expect the error would have been caught earlier by the authors had they compared the code with a working document which at the critical point might have been written: ...If the end of the move list has been reached, go to step 25 ... 25. If the king is not in check go to step 50 ... 50. Display "STALEMATE" and wait for player to request a menu.

paxlp4 (end)
SR-52 USERS CLUB CLOSE-OUT OPTIONS

The lack of input for PAX suggests to me that few former members of the SR-52 Users Club are interested in pursuing the objectives I outlined in PAX1 last August. Recent communication with Maurice Swinnen (779) provides a transition option, in addition to the refund offer I made in V4N2p4.

Maurice has begun writing a newsletter for the Washington DC area TI PPC group, and is now willing to expand it to reach all interested readers. Although there may not be much more strictly new material to cover (until such time as TI markets a new PPC), Maurice is attempting to more broadly, and informally reexamine some of the more important concepts which have already surfaced, with an eye toward helping the beginner, as well as the PPC pro. He is also building up a collection of applications programs, copies of which he hopes to make available at nominal cost, with benefit of at least some technical refereeing. Write Maurice at 9213 Lanham Severn Road Lanham, MD 20801 for more details.

In the meantime (or whenever you decide), you may request of me one of the following options: 1) Send all outstanding contributions to Maurice to be applied to his publications, 2) Send a specified amount to Maurice, and the balance to you, or 3) Refund the entire balance to you. If you choose 2) or 3), please include a SASE (or just a Named envelope if living abroad). If you choose 1) or 2) please provide your current name and address for me to forward to Maurice.

So without further ado, I'll bow out, and wish Maurice good luck. I had a stimulating and challenging experience producing 52-NOTES, but now for me, it's time to move on to other things.

RICHARD C VANDERBURGH

PS I will continue to make back issues of 52-NOTES available for the foreseeable future at $1 each ($1.67 abroad).

January 1980
9459 Taylorsville Rd
Dayton, OH 45424