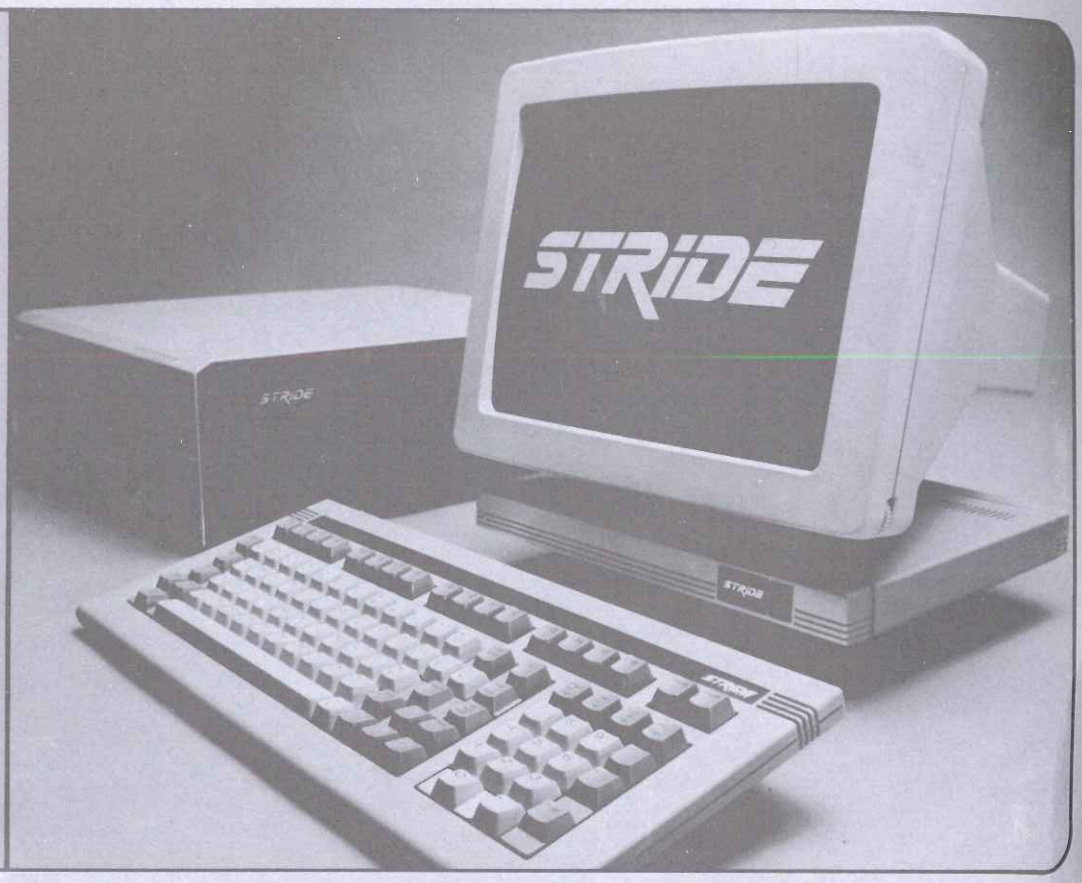
US NEWS AND REPORT

Issue Number
14
JULY, 1985

The Stride 400 Series



From $2900 to $60,000+

As you can see below, the Stride 400 Series has no shortage of features. We've recently added RAM expansion to 12M bytes, low-cost high-speed graphics and a revolutionary hands-free NOD™ cursor control device. But there's more to Stride Micro than hardware. We're an open company, sharing source programs and schematics with our users. That's the reason Stride and Sage (our former name) machines are preferred by many leading p-System developers. It's what we call "Performance by Design".™

Stride 400 Series Technical Specifications

- *10 MHz 68000 CPU (12 MHz optional)*
- *VMEbus interface*
- *256K bytes of parity RAM standard Up to 12M bytes with no wait-states*
- *5¼" 640K byte floppy disk drives*
- *10M to 448M bytes hard disk storage*
- *Battery backed-up real-time clock*
- *4K bytes of battery backed-up CMOS RAM*
- *4 to 22 RS-232C serial ports*
- *Omninet networking hardware*
- *p-System IV.2 w/LAN software*
- *High speed low-cost monochrome graphics (784 x 325 resolution)*
- *NOD cursor control device*
- *FPU hardware floating point*
- *Streaming ¼" tape drive backup*
- *MMU Memory management unit*
- *Stride configuration utilities*
- *TeleTalker communications software*
- *Centronics compatible parallel port*

The Stride Multiuser p-SYSTEM

Stride provides a complete multiuser system **standard** with every machine. This unique system allows multiple p-System (and CP/M-68K) operating systems to co-exist at the same time on a single machine. An easy-to-use beginner's program (MU.BUILD) will generate a multiuser system within minutes. Sophisticated users will appreciate the highly powerful and flexible MU.UTIL program which allows an incredible level of customization of the multiuser system. It is a multitasking, propriety BIOS developed in 1982 (the first commercial multiuser p-System), and has grown in strength and flexibility ever since.

STRIDE MICRO

Stride Micro, 4905 Energy Way, Reno, NV 89502 (702) 322-6868, TWX 910-395-6073.

USUS News And Report

July, 1985 Number 14

Eliakim Willner Editor

The USUS News And Report is published by USUS, P.O. Box 1148, La Jolla, California, 92038. USUS News And Report is a direct benefit of membership in USUS. This USUS News And Report was produced entirely with p-System software tools.

Inquiries regarding membership in USUS should be sent to the Secretary at the aforementioned P.O. Box. Newsletter correspondence and advertising should be directed to the Editor.

Institutional Members

Apple Computer
Attn: Lance Saleme
20525 Mariani Ave
Cupertino, CA, 95014

Donnelly Marketing Information
Attn: Dr. Gary Meyer
13511 Washington Blvd.
Stanford, CT, 06902
(203) 357-8773

Haverford College
Attn: Ed Meyers,
Dir. of Academic Computing
Haverford, PA, 19041

Hayes Microcomputer Products
Attn: Dan McCutcheon
5835-A Peachtree Corners East
Norcross, Ga 30092
(404) 449-8791

Japan Business Automation
Myojo Bldg. 3-50-11
Sendagaya, Shibuya-ku
Tokyo, 151, JAPAN
03/404-2221

NASA-Ames Research Center
Attn: W. Crawford
M.S. 233-15
Moffett Field, CA 94035

NCR Corporation
Attn: Ian Kaplan
11010 Torreyana Rd.
San Diego, CA, 92117
(714) 452-1020

Pinnacle Systems
10410 Markison Rd
Dallas TX 75238
(214)340-4941

SofTech Microsystems
Attn: Sharon Koehler
16885 W. Bernardo Drive
San Diego, CA, 92127
(619) 451-1230

Stride Micro Inc.
4905 Energy Way
Reno, NV, 89502
(702) 322-6868

Texas Instruments
Attn: B. Peterson
PO Box 226015
Dallas, TX, 75266

Xycom, Inc.
Attn: John Van Roekel
750 N. Maple Rd.
Saline, MI, 48176
(313) 429-4971

Officers, Board and Electronic Mail Contacts

Officers

President
Jim Harvison
4144 Red Bandana Way
Ellicott City, MD 21043
301/992-8831
Telemail: President
MUSUS: 70320,165

Executive Vice President
Kathryn Hjorleifson
TDI Computer Systems, Ltd.
66 23rd Street
Toronto, Ontario
M8V-3N2 Canada

Vice President, Member Services
Carl Van Dyke
SPM Support Services
10210 Iron Mill Road
Richmond, VA 23235
804/320-0144 (eve.)
804/320-2561 (days)
Telemail: CVanDyke
MUSUS: 70376,1435

Secretary
Michael Hartman
P.O. Box 3277
Silver Spring, MD 20901
301/445-1583
Telemail: Secretary
MUSUS: 73075,1171

Treasurer
Bart Thomas
First National Bank of Central Jersey
1 West Main Street
Somerville, NJ 08876
201/685-8395
Telemail: Treasurer
MUSUS: 76703,1031

Board Members

Eli Willner (Chairman)
Eliacomputer
1510 East 4th Street
Brooklyn, NY 11230
718/336-4109
718/336-4834
Telemail: EWillner
MUSUS: 76703,500

Dennis Cohen
215 North Kenwood Street; #102
Glendale, CA 91206
818/956-8559
Telemail: DCohen
MUSUS: 71076,1377

Gary Pritchett
SofTech Microsystems
16885 West Bernardo Drive
San Diego, CA 92127
619/451-1230
Telemail: GPritchett

Carl Van Dyke
SPM Support Services
10210 Iron Mill Road
Richmond, VA 23235
804/320-0144 (eve.)
804/320-2561 (days)
Telemail: CVanDyke
MUSUS: 70376,1435

USUS(UK) also covering Europe

Chairman
Mark Woodman
The Open University
Faculty of Mathematics
Walton Hall
MILTON KEYNES MK7 6AA
England
(0908) 653187
(0908) 653678
Telemail: UKChair

Membership Secretary
Angela Grey
Stark Associates
Claremont Buildings
Salbrook Road
SALFORDS
Surrey RH1 5DY
England
(02934) 76747

Electronic Mail Contacts

MUSUS SYSOP
Bob Peterson
P.O. Box 1686
Plano, TX 75054
214/995-0618 (days)
Telemail: BPeterson/TI
MUSUS: 76703,532

Telemail ADMIN
(Contact Secretary for billing information and new accounts)
Jai Gopal Singh Khalsa
MicroStrategies
Box 2278
Vineyard Haven, MA 02568
Telemail: ADMIN
MUSUS: 72355,1015

Local Groups

USUS Washington/
Baltimore Local User's Group
Carl Van Dyke
SPM Support Services
10210 Iron Mill Road
Richmond, VA 23235
804/320-0144 (eve.)
804/320-2561 (days)
Telemail: CVanDyke
MUSUS: 70376,1435

SIG and Committee Chairmen

Advanced System Editor SIG
Samuel Bassett
34 Oakland Avenue
San Anselmo, CA 94960
415/454-7282
Telemail: SBassett
MUSUS: 71735,1776

Apple SIG
(Co-chairs)
Dennis Cohen
215 North Kenwood Street; #102
Glendale, CA 91206
818/956-8559
Telemail: DCohen
MUSUS: 71076,1377

John Stokes
9223 Skokomish Way NE; #2
Olympia, WA 98506
206/459-7598
Telemail: JPStokes
MUSUS: 70345,1256

Bart Thomas
First National Bank of Central Jersey
1 West Main Street
Somerville, NJ 08876
201/685-8395
Telemail: Treasurer
MUSUS: 76703,1031

Application Developers' SIG
Harry Baya
565 Broadway; Apt. 2H
Hastings-on-Hudson, NY 10706
914/478-4241
Telemail: HBaya
MUSUS: 72135,1667

By-Laws Committee
(Acting chair)
Bob Peterson
P.O. Box 1686
Plano, TX 75054
214/995-0618 (days)
Telemail: BPeterson/TI
MUSUS: 76703,532

Communications Committee
Bob Peterson
P.O. Box 1686
Plano, TX 75054
214/995-0618 (days)
Telemail: BPeterson/TI
MUSUS: 76703,532

Communications SIG
Bob Peterson
P.O. Box 1686
Plano, TX 75054
214/995-0618 (days)
Telemail: BPeterson/TI
MUSUS: 76703,532

DEC SIG
Henry Baumgarten
3325 Hillside Street
Lincoln, NE 68506
402/489-6441

Education SIG
Connie Gruber
SofTech Microsystems
16885 West Bernardo Drive
San Diego, CA 92127
619/451-1230
Telemail: CGruber

Graphics SIG
Alan Freiden
Information Systems, Inc.
3865 Wilson Boulevard; Suite 202
Arlington, VA 22203
703/522-8898

IBM PC/Compatibles SIG
(Co-chairs)
Harry Baya
565 Broadway; Apt. 2H
Hastings-on-Hudson, NY 10706
914/478-4241
Telemail: HBaya
MUSUS: 72135,1667

John Blasquez
P.O. Box 305
Walnut Creek, CA 94596
415/935-9295

Meetings Committee
Dan Merkling
Mentor
P.O. Box 11881
Salt Lake City, UT 84147
801/969-7041
Telemail: DMerkling

Modula-2 SIG
Dennis Cohen
215 North Kenwood Street; #102
Glendale, CA 91206
818/956-8559
Telemail: DCohen
MUSUS: 71076,1377

Pinnacle SIG
If interested in chairing this,
contact Michael Hartman, Secretary

Publications Committee
Editor
Eli Willner
Eliacomputer
1510 East 4th Street
Brooklyn, NY 11230
718/336-4109
718/336-4834
Telemail: EWillner
MUSUS: 76703,500

Assistant Editor
Arley Dealey
Pacific Systems Group
3615 Security
Garland, TX 75042
214/349-1515
Telemail: ADealey
MUSUS: 70130,177

Stride/Sage SIG
Bob Peterson
P.O. Box 1686
Plano, TX 75054
214/995-0618 (days)
Telemail: BPeterson/TI
MUSUS: 76703,532

Software Exchange Library
Distribution Chairman
Jon Bondy
P.O. Box 148
Ardmore, PA 19003
215/642-1057
Telemail: JBondy
MUSUS: 71545,2023

Review Chairman
Dennis Cohen
215 North Kenwood Street; #102
Glendale, CA 91206
818/957-4411 (days)
Telemail: DCohen
MUSUS: 71076,1377

Technical Issues Committee
Temporarily Vacant

Texas Instruments SIG
Danny Cooper
1709 Fairfield
Plano, TX 75074
MUSUS: 70735,1122

USUS Archive
Archivist
David Ramsey
1510 S. Bascom Avenue; #8
Campbell, CA 95008
408/377-6297
Telemail: DRamsey
MUSUS: 70076,1161

Word Processing SIG
Samuel Bassett
34 Oakland Avenue
San Anselmo, CA 94960
415/454-7282
Telemail: SBassett
MUSUS: 71735,1776

Editorial

By: Eliakim Willner

Hi there. Lots has happened in the p-System world since the previous issue of the Newsletter. I'd like to talk about some of those things but I hardly know where to begin! The task is particularly difficult since I personally have been involved in these events.

As most of you are probably aware, SofTech Inc., of Waltham, MA, announced that they were leaving the microcomputer software business. Bids were entertained for the purchase of SofTech Microsystems. I and my associates submitted a bid and were fortunate enough to have our bid accepted.

So, as of this writing, it looks very much like I will be one of the new owners of SofTech Microsystems. We have great marketing and development plans for the p-System product line, but this is hardly the forum to discuss them. The tentative name of our new firm is p-Cube Microsystems, Inc. You'll be hearing from us.

A key element in our plans, of course, is to increase the p-System user base. This fact is directly relevant to the health of USUS. Active though our organization is, I hope and expect it to become still more active.

Many of the new USUS members will be UCSD Pascal novices. Many will be new to programming. It will be our job to integrate these folks into USUS and the p-System world in as smooth a manner as possible. A vibrant and imaginative USUS leadership is therefore very important.

Elsewhere in this issue is an announcement of the upcoming USUS elections, to be held at the Baltimore meeting in October. Proxies will be mailed to all members shortly (and may well be in your hands by the time you read this). It is very important for all members to become more active participants in USUS at this juncture. At the very least, fill out and mail in your proxies.

If at all possible, attend USUS meetings (the announcement of the Baltimore meeting also appears in this issue, and reminders will be sent to all members). Participate in USUS SIGs -- if you can't conveniently travel to USUS meetings, participate via our increasingly active Forum on the CompuServe Information Service. Simply subscribe to CompuServe by purchasing a CompuServe Starter Kit at most any computer store, log in and type GO PCS55. This will take you to MUSUS, USUS' very own CompuServe Forum. Follow the easy online directions and leave a message to SYSOP (Bob Peterson) indicating that you are a USUS members and requesting to be put on the online membership roster. That's it! Within a day, normally, you will have full access to an extremely rich resource of information and conversation.

As I mentioned in the previous issue, we are always looking for interesting and informative Newsletter material. Now is the time to put all those intriquing thoughts that flow through your head down on paper. Many a USUS member who *thought* they had no writing talent were delighted to discover otherwise. An article in the USUS Newsletter may lead to bigger and better things (and even if it doesn't, at least you've got *one* article under your belt!). If an article seems a bit ambitious for you, write us a letter. As New York City Mayor Ed Koch says, "How're we doin'?". Why not write to us and let us know!

The preferred means for getting Newsletter contributions to me is via CompuServe or Telemail. Please send plain text; I will insert the appropriate editing directives. I can also accept Newsletter contributions on Stride or IBM PC format diskettes, or "standard" 8" UCSD format diskettes. My electronic and physical addresses appear in the USUS officer list in this issue.

We also welcome display and classified ads. Our ad rates are quite reasonable and result in very high quality leads, if you sell UCSD Pascal or p-System related software or hardware. Contact me for details.

I'm excited about this issue of the Newsletter. Many members responded to my calls for articles in the previous issue, and this month's contributions are substantial in both quality and quantity.

Granville Kirkup's USUS Resource Directory makes its first appearance in these pages. I expect this to become a popular Newsletter feature; our next issue will doubtless contain a much larger Resource Directory.

We have an important article for p-System beginners; one that our TI-99/4 users have been

requesting for some time. This article, by Steve Jacobs, describes the features present (and those absent) on the TI-99/4 p-System implementation.

Experienced Pascal programmer Jai Khalsa (of EPA fame) empties his bag of Pascal programming tricks (well, some of them, anyway) for the benefit of his fellow USUS members.

Modula-2 aficionados will enjoy Dennis Cohen's article describing his implementation of the Kernighan and Plauger software tools in Modula-2, and Tom Cattrall's overview of MoSys, the Modula-2 Operating System from Robinson Systems in England.

And ASE freaks will delight over this issue's two ASE-related articles: Arley Dealey's notes describing differences between the various ASE releases and Richard "ASE Ace" Karpinski's column, which this issue presents his Pascal Assistant macros. Dick's macros enable a programmer to generate the body of a Pascal programming structure at the mere press of a function key. Though I enjoy working on all Newsletter articles, I confess to spending more than the usual amount of time playing with this one!

Harry Baya's article discusses his plan, mentioned in the previous Newsletter, to design a simple database applicaton that would assist us in using the USUS software library.

Finally, to give our membership a taste of what our electronic bulletin boards are like, we are printing a small selection of message "threads" drawn from MUSUS and Telemail. Some interesting technical tidbits here!

An embarrassment of riches, as you can see, but please -- keep 'em coming!

A number of individuals have asked me what my personal plans are regarding a future role in USUS. I will not remain chairman of the USUS board of directors after my term expires in October, though I do intend to run again for the board. I want to remain an active USUS member and would like to remain assistant Sysop of MUSUS and Editor of this Newsletter. That, of course, is up to the USUS leadership that emerges at the Baltimore meeting.

Eli

USUS Resource Directory, 5/31/85

The purpose of the Resource Directory is to help UCSD Pascal users, potential users or developers locate available 'resources' - experts, consultants and contractors with expertise in specific areas of the UCSD Pascal system. The resource directory is open to both companies and individuals, and inclusion is not a recommendation or endorsement by USUS.

This is an 'embryo' version of the Resource Directory, with some initial entries received mostly from the CompuServe MUSUS group. If you have services available to third parties in any area of UCSD Pascal, you are urged to add your name to the directory.

Samuel B. Bassett
Technical writing / Documentation
34 Oakland Ave. San
Anselmo, CA 94960
(415) 454-7282
* Five years experience in writing clear English about microcomputers and general electronics as a journalist, contract technical writer and manual creator.
* Familiar with UCSD Pascal and the p-System, disk formats, RS-232 wiring, Turbo Pascal, CP/M; reading knowledge of Modula 2 and exposure to most other languages.

Felix Bearden & Associates. Inc.
Felix Bearden
1310 Larkview Drive,
Lilburn, GA 30247.
(404) 923-4825
Telex: 9103803560 FELIX BEARDEN
* Measurement & control applications using micro & minicomputers. Mechanical Stock Storage & Retrieval Systems (MSRS). MSRS control & inventory accounting systems. WIP product tracking, color-graphics monitoring systems for copper rod mills, jet engine facilities, etc.
* 25 years experience in computer industry, from compilers to product development.

Dan R. Daniels
6212 Newberry Rd., #624,
Indianapolis, IN 46256.
(317) 841-0744
* Municipal Bond Portfolio Management Systems.
* Using DOS-hosted IV.21 p-System on IBM PC.

Eliacomputer, Inc.
Eliakim Willner,
1510 East 4th. Street,
Brooklyn, NY, 11230.
(718) 336-4109, (718) 336-4834
* Specifying and recommending Pascal and p-System products.
* Providing Pascal and p-System training.
* Custom Pascal and p-System systems and applications. General consultation.

Endicott Consulting Inc.
Steve Endicott
1408 James Way,
Anaheim, CA 92801.
(714) 535-2416
* Communications, real time applications, programming tools.

GK Microsystems Inc.
Granville Kirkup,
PO Box 440412,
Aurora, CO 80044.
(303) 841-1719 (303) 841-2471
CompuServe: 74166,2230 Telemail: GKIRKUP
* Applications development contractor. Completed projects include 'Trim' (Rental store management), 'Servicemanager' (Service shop), 'Able-1' (integrated business) and Real-Estate office applications.
* Complete database-based software applications design & implementation.

* IBM PC/XT/AT. DOS-hosted p-System. Substantial p-System to C conversions.

K3 Software Inc.
Kent K. Kobuchi
80-B Belvedere Street, Suite 2,
San Rafael, CA 94901
(415) 258-9166
* Analysis, design and programming services.
* Apple Pascal, UCSD IV.2 Pascal on Stride 440.
* Experience in engineering/scientific and commercial applications.

Kingdom Computer Concepts
Curtis Sjolander
POB 182
St. Johnsburg Center, VT 05863
* Computer consulting and contract programming
* Computer graphics, simulation, database programming, etc.
* Apple II, Macintosh. Pascal, Modula-2, assember.

James A. Mullens
5524 Outer Drive
Knoxville, TN 37921
(615) 691-7957 (eves) CompuServe 70255,1346
* Experienced with p-System versions II and IV, assembler
* PDP-11, Apple II, Sage II, IBM PC
* Communications, device handlers, graphics & system functions
* Extensive low-level p-System interface experience

Art Nelson
State Rd,
W. Tisbury, MA 02575
(617) 693-3007
* 6 years experience in UCSD Pascal
* Volition Modula 2

Peterson & Associates
Bob Peterson
P.O. Box 1686
Plano, Texas 75074
(214) 596-3720 (answering machine)
* UCSD Pascal, mostly version IV. Familiar with some of the system internals. Have worked with terminal emulation and database software quite extensively. 'Consulting is a part-time effort'.

Software Construction Inc.
Barry Demchak
3252 Holiday Ct, Suite 104,
La Jolla, CA 92037.
(619) 457-1595
* p-System experts & designers. Participated in Version IV p-machine & O.S.
* Native code generator designers. Adaptable p-System designers.
* NEC APC and APC III BIOS, NEC APC graphics subsystem designers.
* Operating system, networking, BIOS & graphics implementations.

SPM Support Services
Carl Van Dyke
10210 Iron Mill Rd,
Richmond, VA 23235
(804) 320-2561
CompuServe: 70376,1435 TeleMail: CVanDyke
* Transportation, Simulation, Operations Planning and Modelling, Railroads.
* Apple Pascal, IV.x Pascal, IBM Pascal/VS.

Cye H. Waldman, Ph.D.
Consultant.
515 Hermes Avenue,
Leucadia, CA 92024.
(619) 942-3016
* Modeling of physical systems: combusion, fluid mechanics, chemical kinetics, multiphase flow, transport phenomena, systems analysis; deterministic and stochastic modeling.
* Ph.D. in engineering; over 15 years experience.

USUS Resource Directory Form

Please add the following entry to the directory:

Name, Title _____

Company (if work address) _____

Number and street _____

City, State, Zip, Country _____

Phone _____

Areas of specialization _____

Additional remarks _____

Mail to: Granville Kirkup
GK Microsystems Inc.
PO Box 440412,
Aurora, CO 80044.

Your entry will also appear in future editions of the Resource Directory. Please amend your entry when any future changes are required.

Classified Ads

For Sale: Sage II with dual 640K floppies, Freedom 100 terminal, and complete cabling. p-System, CP/M; assorted development software. Contact Jim Camp or Brooks Parker at Vega Trading (212)777-0444.

Baltimore Meeting Announcement

The USUS Fall National Meeting will be held at the Omni International Hotel in Baltimore's Inner Harbor. The meeting will start the morning of Friday, October 25, 1985, and run through mid- afternoon on Sunday, October 27, 1985. The theme of this meeting will be "A Workshop on the UCSD p-System Family." We plan to have experts in the areas of applications development, Pascal, Modula-2, communications, data bases, word processing/text formatting, and using UCSD Pascal and/or Modula-2 on specific hardware such as the Apple II, MacIntosh, IBM PC, Sage/Stride, and TI 99/4A. Planned sessions include:

- Writing communications software in the p-System
- Design of a data base management system
- Writing software for the Mac environment
- Writing software for the MS-DOS environment
- Text processing and the p-System
- How to use ASE macros
- The TI 99/4A and the p-System
- Workshop on Apple Pascal 1.2

In addition to these special "theme" sessions we will also hold meetings of a large number of "Special Interest Groups." Meetings centering on Apple, IBM PC, and Sage/Stride computers are expected along with such topical areas as communications, applications development, the Advanced System Editor (ASE), and the software library. Attendees often find that the opportunity to exchange information at these meetings (and in the hallways) can be extremely rewarding, and can alone justify the time and expense of attending the meeting.

As always, the USUS software library will be available for copying by USUS members. The library will be available at a cost of $1.00 per volume plus the cost of media. You can either supply your own disks, or purchase disks at the registration desk (at cost). We plan to have equipment available to copy the library in Apple II, Sage/Stride, and IBM PC formats. As of this writting, it appears that we will also have the library and machines to reproduce both TI 99/4A and Macintosh disks as well.

The meeting will also be an opportunity to conduct the business of USUS. A new board of directors will be elected; both the old and new boards will hold meetings, as well as the Executive Committee. All board and Executive Committee meetings are open to the general membership, and represent an excellent way for you to get actively involved in USUS.

The national meeting will be hosted by the now one and one-half year old USUS Washington/Maryland/Virginia local users' group, or LUG as it is affectionately known. Members of LUG will be supplying the computers needed to copy the USUS library, and manning the registration desk. In addition, we are teaming up with members of the local Washington area TI users' group to present at least one session on using the p-System on the TI 99/4A.

With the uncertainty surrounding SofTech Microsystems and the future of the p-System, the role of USUS as a promoter and educator for the p-System and UCSD Pascal has never been more important. Our fall meeting should be an important opportunity to catch up on the latest news about the p-System, and provide a forum for serious discussions of the p-System's future.

THE LOCATION

The meeting will be held at the Omni International Hotel in Baltimore's Inner Harbor. The Omni is a first class hotel and will provide us with an extremely pleasant environment to hold our meeting.

Baltimore's Inner Harbor is the cornerstone of a revitalized inner city, with the magnificant glass enclosed Harbor Place containing over 140 shops and restaurants as its centerpiece. There are numerous sites and attractions convenient to the hotel including the National Aquarium, the US Frigate Constellation, the submarine USS Torsk, the Baltimore Clipper reproduction Pride of Baltimore, the Babe Ruth Birthplace, the B&O Railroad Museum, the Baltimore Museum of Art, the Baltimore Zoo, Streetcar Museum, Edgar Allan Poe House and grave, Fort McHenry National Monument, and numerous eateries. Needless to say, opportunities to sample Chesapeake crab and oysters, two of the area's most famous delicacies, abound. In addition, there are harbor cruises leaving from the inner harbor around the clock.

CALL FOR SPEAKERS

As always, we are looking for speakers to add to the meeting roster. We are planning to have a number of panel and individual presentations on practical p-System topics. These include developing specific types of programs such as communications or data base software, and writting software for specific machines such as the Apple II, IBM PC, Sage/Stride, or Macintosh. If you would like to speak on a particular topic, or would like to suggest someone as a speaker, please contact Carl Van Dyke.

SPONSORS WANTED

Meetings cost money. To keep meeting costs, and registration fees low, USUS has avoided providing a number of traditional amenities such as coffee breaks, luncheons, etc. at past meetings. One amenity sorely missed by the many caffeine addicts in our membership is the coffee break. We are trying to find a way to provide approximately five coffee breaks for the fall meeting. To this end we are looking for companies (or even individuals) willing to sponsor morning and afternoon coffee breaks. Companies or individuals can also choose to jointly sponsor coffee breaks as well. Sponsorship of a break costs $150.00. If you are interested please let us know!

MEETING REGISTRATION DETAILS

Registration for the entire three day conference is $25.00 in advance, $35.00 at the door. There are also a number of special registration rates as follows:

- 3-day registration
- $20.00 students (advance or at the door)
- 1-day registration
- $18.00 regular (advance or at the door)
- $10.00 student (advance or at the door)

Students must present a current college or high school ID to obtain the reduced registration fees. Table space is also available in the demo room at $50.00 per 3' x 6' table.

An advance registration form follows this announcement. This form along with a check made out to USUS, Inc. must be received no later than October 15 to obtain the reduced registration fee. Please note that you must make hotel room reservations directly with the Omni International. Registering for the meeting does not reserve a room for you at the hotel, nor does it guarantee availability of a room.

I hope all of you will give serious consideration to attending. If you have any questions please contact Carl Van Dyke, a USUS board member and officer. He can be reached through the USUS mailing address or at 804/320-2561 days, and 804/320-0144 evenings. Please do not call after 9:30 PM eastern time.

HOTEL RESERVATION DETAILS

The meeting will be held at the Omni International Hotel in Baltimore's Inner Harbor area. Special room rates have been arranged of $64.00/night for a single, and $80.00/night for a double plus applicable local and city taxes. A charge of $16.00 will be added for a third person sharing a double room. Children 17 years and under may share the same room with parents at no additional charge.

These reduced rates are available for the nights of October 24, 25, and 26, 1985. You may reserve a room either by calling the Omni, or by sending the form that follows this announcement directly to the

Omni. Do not send the hotel reservation form to USUS. You must make all hotel arrangements directly with the Omni.

Should you decide to make a telephone reservation, you must call the Omni International Baltimore directly. The Omni can be reached as follows:

Omni International Hotel
101 West Fayette Street
Baltimore, MD 21201
301/752-1100

You must identify yourself as attending the USUS/UCSD Pascal Users Society meeting to obtain the reduced rates. Preferred rates and space in our block of rooms cannot be confirmed through any "800" toll-free service. Rates cannot be changed at check-in or check-out for guests who fail to identify their affiliation at the time the reservation is requested. Make sure you tell the reservation agent you are with USUS!

All reservations, whether by telephone or mail, must be made no later than September 24, 1985 to obtain the preferred rates. Reservation requests made after September 24, 1985 will be accepted on a space available and a rate available basis. If you make a reservation by mail please use the form provided here; this will guarantee you will obtain the correct room rate. Please make your reservations early as space is limited.

TRANSPORTATION

Baltimore can be reached in by numerous modes including car, train, and air. If you are flying you should use the Baltimore- Washington International Airport (BWI). Inexpensive scheduled limousine service is available to the Omni. Baltimore is also a major stop on Amtrak's Northeast Corridor, with hourly service throughout the day from points south and north. The train station is a short cab ride from the Omni. If you are arriving by car you should check with the hotel about parking arrangements and fees. Numerous attactions are within walking distance or accessible by public transportation from the Omni, as a result you should not need a car during your stay in Baltimore.

MEETING UPDATES

We plan to do a first class mailing to all USUS members with additional details on the meeting in late August. This mailing will be combined with the proxy ballots for the annual board election. The August mailing should include a specific meeting schedule and list of presentations and speakers.

USUS Baltimore Meeting Registration Form

Name, title _____

Company (if work address) _____

Number and street _____

City, state, Zip, country _____

Day telephone _____

Please register me for the USUS Fall Meeting as follows:

_____ Regular Advanced Registrant ($25.00)

_____ 3-day Student Registrant ($20.00)

_____ 1-day Regular Registrant ($18.00)

_____ 1-day Student Registrant ($10.00)

I am a member of USUS: Yes No

I have made a room reservation at the Omni International Hotel: Yes No

I am interested in sponsoring a coffee break or in demo room space, please contact me: Yes No

---

Please enclose a check made out for the appropriate amount payable to USUS, Inc. Mail this form to:

USUS
Fall Meeting Registration
P.O. Box 3277
Silver Spring, MD 20901

USUS MUST RECEIVE THIS FORM NO LATER THAN OCTOBER 15, 1985 TO OBTAIN THE ADVANCED REGISTRATION RATE.

Note: All student registrants will be required to present a current school ID at the door.

USUS Hotel Registration Form

Please reserve hotel accomodations as circled.

Name, title _____

Company (if work address) _____

Number and street _____

City, state, Zip, country _____

Day telephone _____

Arrival date _____

Departure date _____

Length of stay (nights) _____

Please submit only one registration form per room request. Reservations are held until 6 pm unless guaranteed by American Express, Diners Club, Carte Blanche, or one night's deposit. All guaranteed rooms must be cancelled 72 hours prior to arrival date.

Type of credit card _____
Credit card number _____
Expiration date _____
Signature _____

Reservations received after September 24, 1985 will be accepted on a space available basis.

Please circle requested rate:

$64.00 Single $80.00 Double

NOTE: If a room at the requested rate is unavailable, one at the nearest available rate will be reserved. Room charges subject to applicable local and city taxes.

Mail this form to: Omni International Hotel
Reservations Office
101 West Fayette Street
Baltimore, MD 21201

Questions concerning your reservation? Call our Reservations Office at 301/752-1100.

Salt Lake Meeting Minutes

Advanced System Editor SIG

The Advanced System Editor SIG met on April 26, 1985 at 11:00 AM. The meeting, chaired by Carl Van Dyke, was attended by approximately 10 individuals and lasted approximately one hour. Attendees included Arley Dealey, author of the ASE editor, and several ASE distributors. Volition Systems, the primary ASE vendor, was not represented.

Agenda items included:

- o ASE Version 1.0
- o Bugs/Requested Changes
- o Future of Volition Systems
- o ASE Upgrade Policy
- o Edvance/SofTech
- o Availability of Bug List
- o Lineage of ASE

A new version of the ASE editor is now available from Volition Systems. This version is designated 1.0 and contains many new features and fixes several bugs. Arley described those changes he could recall. He also indicated that a more detailed description could be supplied to the Newsletter with a little "gentle persuasion." Versions of ASE are commercially available for Apple Pascal 1.1 and 1.2, IV.0 through IV.21, and Volition's II.2. It was noted that IV.21 required a different version than IV.0 through IV.14 (IV.14 is NCI's extension of IV.13). While ASE 1.0 works on all versions I.5 and above, it is not commercially available for all versions.

Changes include:

- o 12-20% Reduction in Code Size
- o Increased Terminal Requirements
- o Does Not Clear Bottom of Screen if Terminal Supports "Insert Line" (i.e. your text no longer disappears when you insert stuff in the middle)
- o Allows $ Wildcard in Destination File Name
- o Name of Work File Changed from .ASE! to .NEST
- o Under IV.21 ASE knows how to use increased compiler info, will automatically obtain correct file name

The increased terminal requirements include commands to clear to the end of line, and clear to the end of the screen. In addition, an 80 column by 24 line display is required. Note that the unenhanced Apple II/II Plus/IIe has a virtual 80 x 24 screen and can thus use ASE 1.0. A number of other changes were mentioned in general terms including changes to blank line handling and bug fixes.

Other bugs, and a number of suggested changes were discussed. Jon Bondy indicated that Arley's "fixes" removed features he found useful, once again proving some old adage about pleasing people. It was noted that ASE will not work with MS DOS volumes in the hosted system, but should work with p-System virtual volumes. The common request of a version that does not create a back-up file was mentioned. Arley said this was unlikely ever to be implemented, but he was not adverse to thinking about it for a price... Arley indicated that a bug list existed, but was not releasable under his agreement with Volition.

Some time was spend discussing a problem one user was having using ASE with a RAM disk. It seemed that ASE worked fine on all volumes except the RAM disk. Use of the RAM disk hung the system. Arley stated that ASE would write information in the last block of a volume when it opened the back-up file. Often, RAM disks are slightly mis-sized in the directory, but nothing ever writes to the last block so the problem is not detected. ASE is very good at finding this bug in the OS implementation because it does write to the last block.

Volition is currently selling ASE, and is offering an upgrade for $20.00. This is the case as of the end of April. Volition is in bankruptcy, and the status of ASE may change shortly. The final resolution of who will be supporting and distributing ASE in the future (if anyone) is unknown. Please standby for announcements... In the meantime, Volition is shipping promptly.

At one time SofTech distributed a version of ASE called Edvance. It was the group's understanding that SofTech no longer was selling this product. Verlene Bonham thought that an article on the history and lineage of ASE would be interesting. She indicated that she might be persuaded to write such an article for the Newsletter.

Upon reaching noon it was generally agreed there was not much else to say without bringing on a law suit from somebody, so the meeting was adjourned.

Communications SIG

Arley Dealey reviewed some of the new changes in the MUSUS software.

Most of the session was spent discussing differences among the various communications packages on the p-System.

The following are based on the use of REMUNIT:

TERMINAL	Bob Peterson
TELETALKER	Randy Bush
MAIL	F. Whittington
EDDOWES	Arley Dealey, for Apple III
REMCOM	TICOM
LOGON	only for 300 baud; not recommended
EPA	MicroStrategies, the REMUNIT has been very much changed in this program.

Others:

CROSS-TALK	Not real
P-TERM	Novations (?) for the Apple and 64K systems only.

INTERLINK	concurrent support
DATALINK	Link Systems. Also for Apple III
Datalex	
N.C.I	
Kermit	Library, Bob Peterson, V. Bonham very minimal implementation, no file server yet. A good article on Kermit is available in back issues of BYTE magazine.
CONVERS	A. Dealey. Has script language, available on vol 29 or 30 from USUS library.
XMODEM	Bob Peterson is working with XMODEM protocol and may contribute something to library soon.

A need for a message taking program was discussed. A Bulletin board system is available on MOG for the Apple III. Craig Vauhn offers SOFTWARE SORCERY.

A program that could take messages in the background was discussed. An idea to put the task in GOTOXY described using illegal position values such as GOTOXY (-1000,-1000) to call the message procedures.

Another idea from Bill Bonham was to put a new unit into SYSTEM.PASCAL. One problem with this is that initialization must be done by a program run by the user since there is no way to guarantee the order of initialization of units in the p-System.

Modula-2 SIG

The Modula-2 SIG meeting was called to order by Chairman Dennis Cohen. Attending were Dennis Cohen, Mark Luker, Tom Haag, Gary Pritchett, Jon Bondy, William Smith, Bill Bonham, Verlene Bonham, Arley Dealey, and Jim Harvison.

The agenda was:

1) Scenic/Volition Native M2 compiler for the Sage/Stride.
2) Macintosh Modula-2 implementations.
3) Modula-R
4) Interface Technologies Modula-2 for IBM-PC
5) Library Standardization Effort
6) Modula-2 volume in USUS library

Bill Bonham gave a brief presentation on the first topic. It runs "on top of the p-System" -- the only interface is a File/IO access module which handles requests from executing Modula-2 programs. The compile speed is about 500 lpm. The compiler is approximately 650 blocks and is unsegmented. Slightly more that 512K of memory is required for even the smallest module to be compiled; however, 750K should be sufficient for almost anything. Some pieces of the library need work. The bugs are not fatal, but they do exist. Bill did all of the neat graphics demos on the Stride using this compiler.

Jon Bondy made negative comments concerning support by Volition/Scenic of the beta-test Macintosh product. This was echoed by Dennis. Dennis also told of good experiences with the Modula Corporation system for the Mac. He stated that it compiles to m-code, but the Toolbox interface was very good and that true Macintosh programs spend most of their time in the ROM and therefore most of his programs were running at 60-70% the speed of native coded Pascal. Besides that, he purchased the package when it was still $90 and felt that their documentation was so good that it was almost worth the price by itself.

Modula-R is a topic of interest to Bill Bonham -- he would like more information on it. So far, all that is known by anyone in attendance is that it is M2 extended to support database primitives and that it was published in a paper from ETH. Anyone with more information is encouraged to contact both Bill and Dennis.

The Interface Technologies Modula-2 was fairly roundly condemned by those few in attendance who had used it. Almost all of the complaints were based on the forced use of the syntax-directed editor, which does things its own way (and nobody liked its style). However, the compiler is very fast and generates fairly tight native code. The $80.88 offering was described as inadequate to do anything with.

BSI is reported to be working on a draft standard as published in the MODUS newsletter. MODUS will be having their first meeting on this side of the Atlantic in the San Francisco/Palo Alto area this September.

Major issues are function styles, how errors are passed back, lack of some notation for scientific and math applications, lack of matrix operators, how to handle "side effects" or avoid same in implementations, and static vs. dynamic linking.

Dennis reported that Volume 33 is now in the Library and is an all Modula-2 volume. It consists of a Volition submission -- the source to the p-Shell and related utilities and documentation that they shipped with their Modula-2 systems -- and a submission from Joseph Folse that was a rework into Modula-2 of the B-tree items on Volume 16 (15?). He reported that he was still trying to track down another submission, the LIBMAP utility shipped with the Volition package and that he would welcome any other M2-related submissions.

There is a project management system written in M2 called TimeLine.

Many oil refinery simulations run under the p-System.

SofTech expressed their interest in Modula-2.

Discussion of Borland's Modula-2 (not yet released) provoked major concern from the attendees since it is rumored that the M2 will not concern itself with many of the standards and that its ability to do modules will be limited.

Other comments can be summarized: "One problem in the M2 community is the lack of a leader who would take a firm stand on the M2 standard's issues. Wirth has stepped back from many of the hot topics and no one else has any clout."

IBM SIG

Attendees: Richard Guenzi, Carl Van Dyke, Paul Buhr, J.Dennis Green, Norman Cannon, William Burns,

Charles Schwarz, John R. Cluff, Joseph A. Ladyka, Jr., Jon Nevins, Harry Baya,

The following notes reflect my sense of what was said during the SIG meeting rather than an accurate summary. I (Harry Baya) have added comments of my own. I hope this is of some use in framing what questions you should be asking in dealing with these areas.

Most of the discussion dealt with the UCSD Pascal p-systems put out for the IBM-PC by NCI (Network Consulting Inc.) and SMS (SofTech Microsystems). Jon Nevins and Carl Van Dyke have been working with DOS-hosted 4.2.1 p-System from SMS and are very pleased. Both had some prior experience with NCI. Harry Baya has been working with NCI's version C1.f and has just begun experimenting with NCI's version C2.0.

Any comments about SMS version 4.2 is also true of 4.2.1.

Several participants expressed the view that NCI's version of the p-System had been clearly preferable to that put out by SMS up until SMS version 4.2. The p-System put out by IBM (an early SMS version) has received bad evaluations by several USUS members who have tried it. Jon and Carl feel that version 4.2.1 is as good or better than the NCI version they had been using.

The complete 4.2.1 development system can be purchased as an upgrade for about $200. Owners of NCI systems qualify for this upgrade price.

There was a lot of discussion of the new features available in 4.2.1 and how these differ from earlier versions. Some of the comments were:

4.2.1 : - permits larger code pool segments - Improved access to data space outside the 64k data space (though this is somewhat clumsy, it is an improvement. One user estimated that access time was as much as 10 times faster than with a file in ram disk) - Improved Native code generator o supports 8087 o native code now runs much faster than p-code in most situations. This is an improvement. Native code seems to take advantage of 8087 board. o Quickstart is impressive, it even quickstarts the operating system.

- Due to changes (eg to the FIB / file information block) some programs that ran with NCI's p-System will not run with 4.2.1, most notably ASE (Advanced Systems Editor).

- Volition Systems has a version of ASE that will run under 4.2 and they are still selling it now. As they are in the process of dissolving the firm this may not continue to be so.

- 4.2.1 has instructions that will permit reading in the 10-sector floppy disks created on the NCI system, but those formats are not supported and SMS p-System users work in 9-sector format. The 9-sector format disks can be copied with standard DOS disk copiers, the 10-sector cannot. The 10-sector format holds 800 blocks, the 9-sector holds 720.

- The 4.2.1 system runs under the DOS operating system and does not require that a p-System partition be created. The system uses a virtual p-System volume which appears as a single file in the DOS file system. The volumes in the p-System partition can still be accessed from the DOS hosted system.

- 4.2.1 can create, write to and read from files in the DOS partion. (i.e. non-contiguous files)

- 4.2.1 permits input and output to go through the same drivers used by DOS (eg to printer or to hard-disk)

- programs compiled under 4.13 (eg NCI) will run under 4.2.1 without recompiling but programs compiled under 4.2 will not always run under 4.13.

My overview of the two systems (NCI 2.0, SMS 4.2.1) as it relates to the standard IBM PC DOS enviroment are:

- Both systems permit setting up applications that can be run from a DOS environment and which use DOS files for data or text. The user need not use any p-System commands.

- The SMS version will often be easier to set up because it can run on a hard-disk without a p-System partition and because it can use the DOS drivers directly.

- There are other strengths and weaknesses of the two systems and my guess is that NCI will release the next upgrade. This could put them ahead again if, indeed, they are now behind.

NCI Version 2.0

- The transition from C1.f to C2.0 is relatively easy. Harry Baya had to call NCI twice in the process of learning how to create and use files and programs stored in the DOS partition.

- The DOSOPS unit in SYSTEM.PASCAL uses up at least 3k of data space. It can be removed if the DOS interface stuff is not used. This loss of data space can be a serious problem if an application is already very tight on data space.

Using DOS files rather than files in a p-System Partition (both NCI and SMS):

- The good news: The non-contiguous file structure on DOS permits a single file to be written in more than one contiguous space on a disk. This means that the full DOS partition is available for files at all times and that there is no need to "k)runch" files as we do in p-System partition.

- The bad news: If a hard disk has a large number of such non-contiguous elements it is possible for the system to permanently loose all or part of a file. To reduce the chance of this occurring you must periodically reduce the number of non-contiguous file sections on the disk. One way to do this is to off-load all files, reformat the disk and the load all the files back in file by file.

- The Pascal filer will not deal correctly with files in the DOS partition. The program DOSFILER is used for this purpose and permits transferring files back and forth between the p-System partition and the Dos partition.

IBM PC-AT

- SMS version 4.2.1 will run on the IBM PC AT

- NCI has a special release of the p-System for the AT but this is not considered an upgrade for pricing purposes, i.e. you have to pay full price for this system. NCI is in the process of developing an upgrade from version 2.0 that will run on both the PC and the AT.

Clocks in the IBM PC.

- SMS and NCI provide ways of accessing hardware clocks in the PC - The standard machine does not have a hardware clock - Several add-on cards for the PC include a hardware clock. - The DOS environment has a software clock but this is not available from within the p-System as far as I know.

Application Development SIG

Attendees: Richard Guenzi, Carl Van Dyke, Paul Buhr, Joseph A. Ladyka, Jr., Jon Nevins, Kelly Ward, Jim Harvison, Dennis Cohen, Ed Kyte, Harry Baya, J.Dennis Green

These notes should not be construed as the verified truth. These communicate what I think I heard and are intended to help others get a flavor of the discussion.

In particular, any statements concerning particular vendors or products should be verified before using them as a basis for decisions.

Joe Ladyka was elected as the new chairperson of this SIG.

Part of the discussion dealt with the versions of the p-System available for the IBM PC. These are covered in the notes of the IBM SIG meeting and are not included here.

Creating applications that can use a variety of printers:

For simple setups most applications have a setup program that asks the user to enter the printer control characters for their printer.

This is normally done only once when the application is initially configured by the end user, eg. "Enter character pattern to start condensed print " "Enter character pattern to end condensed print"

Harry Baya stated that he would like to see a high quality package that could be used to interface any application with the most common printer/printer interface. (eg. an IBM PC Printer interface unit). No one in the group was aware of such a package being available to application developers for inclusion in their package.

UCSD p-System TEXT files:

There was some discussion of the file format of a UCSD Text file. This is covered somewhere in the USUS library and is also covered in disk #33, the latest addition to the libary. Disk 33 is a Modula-2 disk.

Among the items mentioned were:

Two leading blocks that may be used by the p-System (i.e they do not contain the text information written to the file and do contain such information as the length of the file)

"dle esc" sequences to more efficienty store sequential blanks in a text file

Text files are composed of contiguous two-block pages. A line cannot cross a page boudary.

The unused portion of a page is filled with null characters.

Future of the p-System:

The future of the p-System was discussed. Some attendees felt the system was dying while others saw it as still healthy and growing.

One attendee felt the 64k data space limitation was particularly limiting now that many machines (eg all new IBM PC's) come with 256k or more of ram).

Another felt that the Unit structure and the dynamic segment swapping more than compensated for other shortcomings of the p-System.

Several attendess commented that the p-System's availability on several machines has permitted them to support their applications on several machines with a minimum of fuss.

There was some discussion about the ease of use of the system for naive end-users.

There was no clear consensus. Several people felt that the system was not suitable for unsophisticated user. The filer was specifically mentioned as something some developers would like to hide from their unsophisticated users.

Other Information:

Though this may be covered elsewhere I am going to include some notes from the major vendor's pannel that may be of interest to application developers:

SofTech currently supports the p-System on the following machines:

- IBM-PC and AT
- Apple MacIntosh
- TI Professional
- Wang PC (DOS-Hosted only)

SofTech also has relationships with other vendors that permit those vendors to provide SofTech based P-systems. These include, among others:

- Stride Microsystems (formerly Sage)

- Network Consulting Inc.
- Eliacomputer (for DEC machines)
- Radio Shack
- PCD
- Tenneco

Texas Instrument has dropped its support of the p-System and is thought to have sold its rights to NCI.

The Mac XL can use SofTech's Pascal to access up to 2-meg of ram (via ram disk) and can deal with files on the hard disk.

Electronic Meanderings

By: Various USUS Members

The Programmers' SIG
Sb: parthogenetic Pascal
Fm: Steve Brecher 70365,164
To: All

```
program self;                                          {}
var                                                    {}
  i: integer;                                          {}
  s: array[0..9] of packed array[1..62] of char;       {}
begin                                                  {}
  s[0] := 'program self;                                       {}';
  s[1] := 'var                                                 {}';
  s[2] := '  i: integer;                                       {}';
  s[3] := '  s: array[0..9] of packed array[1..62] of char;    {}';
  s[4] := 'begin                                               {}';
  s[5] := '  for i := 0 to 4 do writeln(s[i]);                 {}';
  s[6] := '  for i := 0 to 9 do writeln(Q  s[Q,i:1,Q] := QQQ,s[i],QQQ;Q);';
  s[7] := '  for i := 1 to 62 do if ord(s[6,i])=81 then s[6,i]:=chr(39) ;';
  s[8] := '  for i := 5 to 9 do writeln(s[i]);                 {}';
  s[9] := 'end.                                                {}';
  for i := 0 to 4 do writeln(s[i]);                    {}
  for i := 0 to 9 do writeln('  s[',i:1,'] := ''',s[i],''';');
  for i := 1 to 62 do if ord(s[6,i])=81 then s[6,i]:=chr(39) ;
  for i := 5 to 9 do writeln(s[i]);                    {}
end.                                                   {}
```

From: EWILLNER
To: EDITOR (REC)
CC: us.us
Subj: Patch for Buggy DEC RX02 Bootstrap

The pre-Eliacomputer release of the DEC p-System has a bug in the bootstrap for the RX02 (double density disk) device. The test for density does not work properly. The result is that the user can only boot a single density disk; attempts to boot from a double-density disk result in a system crash.

This is unfortunate because most users with double density drives will of course prefer to boot of a double density, rather than a single density disk! (Note: The bug is in the bootstrap, not the driver -- once the system boots properly disks of either density may be used.)

Currently shipped systems do not have this problem. The patch which follows will not cure the bug, but will reverse it -- it will produce a bootstrap that will boot ONLY double-density, NOT single-density disks. If the user keeps a copy of both a patched and an unpatched RX02 bootstrap he will have something

that will work on both kinds of disks.

The patch is implemented as follows: Run the PATCH utility. Read in block 0 of the RX02 bootstrap, which is distributed as DYBOOT. MAKE SURE YOU HAVE BACKED UP THIS FILE FIRST!!! Examine location 36 (decimal). It should contain the value 0303 in byte-flipped hex. If it does not contain this value, DO NOT ATTEMPT THE PATCH. You have an older version of the bootstrap, and offsets are no longer the same. Otherwise, Change the 0303 to A000, which is a NOP instruction. Save your change and exit PATCH.

Use the ABOOTER utility to copy your patched bootstrap to a double- density disk with system files, and boot it. Good Luck!

Eli Willner

From: UKCHAIR
To: USUS
Subj: Compiler Listing Pagelength

This fix [to change the lines/page of a compiler listing] is courtesy of the USUS(UK) mail system...

From: Hans Kasche <USUS.AVERSAL>
Subject: telemail: compiler listing
To: usus.mwoodman

Pascal compiler IV.13, segment BACKEND, procedure 5 contains a LDCB 60 instruction at offset 63 decimal. Change this to suit your wishes. Also if you want the header of each page to be at top-of-form (and not two lines down) delete, i.e. substitute p-code nop (9C hex), instructions at offset 24 dec to 36 dec inclusive in segment BACKEND, procedure 28. These begin with LCO 99 and end with SCIP1 5. regards

Jeremy Morton

From: TPOWELL
To: UKChair
CC: USUS
Subj: Compiler Listing Pagelength

Jeremy,

Many thanks for the tip. I'll be using it! Since I'm still using IV.1, my compiler is: Pascal Compiler IV.1 c5s-2 The LDCB 60 is at offset 63 in this version also, but it is in procedure 4 of BACKEND, rather than procedure 5. For those who have this version, the 60 (= 3C hex) may be found in block 86, at (decimal) offset 82. (To be sure you have the right spot, the context is: AD82 FE80 3CB3.)

Ted

From: EWILLNER
To: UKChair
CC: USUS
Subj: Compiler Listing Pagelength

Jeremy -- Thanks much; I'll be using it regularly too! For Version IV.13 users, the patch location is block 10, offset 270, same context as Ted mentioned.

Eli

Three Little Tricks in UCSD Pascal

By: **Jai Gopal Singh Khalsa**

Two of the greatest benefits to programming in a high level language are readability and portability. Pascal, in particular, is often used to illustrate algorithms as it is readily understood even by those who commonly use a different programming language. Still, vast numbers of C programers who populate the pages of professional journals insist that their nearly incoherent language is superior to Pascal because it is less restrictive and allows (forces?) greater intimacy with the code at the bit/byte level. And the experienced and/or creative Pascal programmer will admit that the strong type- checking that normally saves us from careless mistakes can be extremely frustrating at times.

Fortunately, the original implementors of UCSD Pascal were practical people who added a few 'low-level' intrinsics to the language that the venturesome programmer eventually discovers and employs to alleviate some of the restrictions of standard Pascal. This article will explore three of these intrinsics, SCAN, MOVELEFT, and PMACHINE, and in the process, demonstrate three techniques that may apply to 'common' programming problems. It will be seen that, at the expense of somewhat reduced readability, significant gains in speed and flexibility are possible using these intrinsics.

Trick #1 - SCANning a Boolean Index:

The SCAN function is intended for quickly scanning a packed array of characters to locate a particular character. The formal declaration from page 216 of my User's Manual is:

```
FUNCTION SCAN(length, partial expression, array): INTEGER;
```

'length' is an integer, 'array' is usually a PACKED ARRAY OF CHAR, and 'partial expression' is a '<>' or '=' followed by a single character in quotes.

SCAN scans 'array' for 'length' characters, *or* until it finds a character that satisfies the 'partial expression' (whichever comes first). It returns the offset from the position in 'array' to the point at which it stopped.

The manual goes on to give examples and details including the fact that:

1) 'array' may be subscripted to indicate a starting position for the scan and 2) 'length' may be negative in which case the scan will be from 'right-to-left' instead of 'left-to-right'.

For our purposes here, though, the interesting key word in the above definition of the 'array' parameter is 'usually'. It points to one of the most delightful and liberating discoveries regarding these low-level UCSD Pascal intrinsics; that is, *no type-checking is performed.*While 'array' is *usually* a PACKED ARRAY OF CHAR, it can actually be a variable of any type!

Consider, for example, the following data type:

```
VAR boolIndex: packed array[0..4095] of boolean;
```

It might be used as an index to keep track of 'used' records within a file (or blocks on a volume in a non-contiguous file system); deleting a record would be as simple as marking its corresponding boolean element to FALSE. Then, when space for a new record is needed, the array of booleans is searched for a FALSE element indicating an unused record position. A single block (512 bytes) can manage 4096 records. (NOTE: There is nothing sacred, of course, about 512 bytes... index 'pages' can be any convenient size and there can be as many as necessary; a mere 25 blocks will maintain a free space index for 102,400 records.) The fact that a 'packed array of boolean' in UCSD Pascal packs eight elements to a byte is a crucial bit (no pun intended) of practical knowledge, not only for saving space, but also for utilizing the SCAN function in

the unconventional way that follows.

The straightforward (and most readable) approach to searching boolIndex is:

```
k := -1;
repeat k := succ(k);
until (k = 4095) OR not boolIndex[k];
if not boolIndex[k] then recNum := k
  else RecNotFound;
```

Unfortunately, this method can be rather slow when 'k' is large; it helps only a little to disable range checking.

Another way to view the boolIndex variable is the 'raw memory' approach more commonly practiced in assembly language. In this view, it is simply an array of bytes (or words as we shall see). Recalling that there are eight bits to a byte and that all eight bits set to TRUE (binary 11111111) is equal to decimal 255, it would seem that finding the first byte *not equal* to 255 would put us very close to the first FALSE (i.e., zero bit) element. In practice, word boundaries and byte-sex enter the picture and the code to search the boolean index using the SCAN function is:

```
k := SCAN(sizeOf(boolIndex),<>chr(255),boolIndex);
if k < sizeOf(boolIndex) then
  begin
    k := (k DIV 2) * 16;
    while boolIndex[k] do k := succ(k);
    recNum := k;
  end
  else RecNotFound;
```

The SCAN function returns the byte offset of the first byte in boolIndex that is not equal to 255, i.e., that contains at least one zero or FALSE bit. Since microprocessors generally address memory in words rather than bytes and the order of bytes within a word differs depending on the CPU (i.e., the 8086 and 68000 have different 'byte-sex'), it is necessary to convert the byte offset returned by the SCAN function to a word offset and then search the entire word for the zero bit. Thus, the first possibly FALSE element of boolIndex is the byte offset (k) DIV the number of bytes per word (2) times the number of bits per word (16). There is no need for a bounds check in the WHILE statement as the SCAN function has guaranteed that a FALSE element will be found within the word (i.e., a maximum of 16 iterations of the WHILE loop in this case).

The difference in speed between these two approaches can be dramatic. To find the 4000th element of boolIndex, the SCAN function is 571 times faster than the straightforward approach (yep, 57100 percent)! Times are as follows:

```
Straight REPEAT loop.....953 ms
Range checking disabled.....827 ms
Using the SCAN function.....1.67 ms
```

For an excellent discussion of the use of bitmaps for list comparison and file searches (including a fascinating bitmap compression technique), see Eric Sohr's article *Bitmaps Speed Data-Handling Tasks*, October 1983 Byte, pg. 480.

The NonZero Function and A WARNING!!!:

A technique similar to the above may be used to discover non-zero elements in an ARRAY OF INTEGER but, since type-checking is not performed on the 'array' parameter, you must take special care not to reverse the 'length' and 'array' parameters. In a particular application, I was using a large, one-dimensional array (swBuffer) to maintain variable sized two-dimensional arrays and needed a 'NonZero' function to quickly tell me if a specified subrange in 'swBuffer' contained any non-zero elements. I spent an embarrassingly long time trying to figure out why the following expression always (and erroneously) returned TRUE:

```
NonZero := (SCAN(swBuffer[FirstEntry],<>chr(0),numBytes) < numBytes);
```

The answer was that I had switched the 'length' and 'array' parameters and the SCAN function was happily scanning my 'numBytes' variable for the length indicated by my starting element in the array! This happened to be zero which, of course, was less than 'numBytes' so 'NonZero' always returned TRUE. If I hadn't indicated a starting element in the array *or* had used an expression or the 'sizeOf' function instead of an integer variable for the 'length' parameter, the compiler would have caught the mistake. There are undoubtedly many other clever uses of the SCAN function but *be careful!*

Trick #2 - Fun and Games with MOVELEFT/RIGHT:

The MOVELEFT/RIGHT intrinsics are also byte-oriented functions that accept parameters of any type. They are defined on page 210 of the User's Manual as:

```
PROCEDURE MOVELEFT(source, destination, length);
```

'source' and 'destination' are any sort of arrays. Or (as with BLOCKREAD/WRITE and FILLCHAR) they may be of any other type, as well. If either is an array, it may be subscripted, and if either is a record, it may have a field specification.

'length' is an integer. Moves 'length' bytes from 'source' to 'destination', starting at the left. (MOVERIGHT does the same but starts at the right!)

What are these things good for? At first, I used them only on PACKED ARRAY OF CHAR, in particular for fooling with text buffers. Given a large buffer, one can quickly 'remove' a specified character at position 'k' by doing:

```
moveLeft(buffer[succ(k)],buffer[k],numChars - k);   numChars := pred(numChars);
```

When writing the buffer to a text file, it must be broken up into 'pages' of less than 1023 characters each and padded with NULs. SCAN (with a negative 'length' parameter) is used to find the first carriage return before buffer[1023] and then MOVELEFT is used to move those characters into a 'OnePage' buffer that is written to disk (using BLOCKWRITE). (NOTE: A more complete discussion of the peculiarities of UCSD text files is left for another article).

Another fairly tame application of MOVELEFT is the case where you want to place a substring into another string at a particular position. For example, if you want to right-justify a number (131) within a fixed length string, the following will do:

```
st := 'Number of Students.......';   {initialize string}
str(numStudents,st2);                {convert 'numStudents' to string}
moveLeft(st2[1],st[succ(length(st) - length(st2))],length(st2));
```

results in 'st' having the value 'Number of Students....131'.

The above examples are quite obvious because the data being manipulated are explicitly declared as bytes (PACKED ARRAY OF CHAR and STRING). What took a little longer for me to realize was the flexibility implied in the definition of 'source' and 'destination' by the words 'they may be of any other type, as well'. For example, suppose you want to store a Pascal record of arbitrary size starting at byte zero in block eight (8) of a file (there are other things in blocks 0-7). One way to do this is as follows:

```
VAR paramFile: file;
    OneBlock: packed array[0..511] of char;
    OneParam: ParamCo;
begin
  {$I-}   reset(paramFile,SkusIndex);
  if (blockRead(paramFile,OneBlock,1,8) = 1) then
    begin
      moveLeft(OneBlock,OneParam,sizeof(OneParam));
      ...
```

The opposite is done when writing the record:

```
moveLeft(OneParam,OneBlock,sizeof(OneParam));
if (blockWrite(paramFile,OneBlock,1,8) = 1) then...
```

The Insertion Sort Revisited:

Now for some real fun... well, programming fun anyway. The straight insertion sort is perhaps the most basic method of sorting. It is the method a card player normally uses to put his cards in order if he/she picks them up one at a time. It is also the method often used to maintain an ordered list. A programmer, however, can not quite so easily place an item between two other items, especially if the items are stored in an array. (NOTE: There are many variations of linked lists that will do this job nicely but that is beside the point of this example.) A space must first be made by moving all the items above the insertion point (ptr) 'up' in the array, starting with the last item and traversing 'down'. For example:

```
numItems := succ(numItems);
for k := numItems DownTo succ(ptr) do item[k] := item[pred(k)];
item[ptr] := newItem;
```

Similarly, if item[ptr] is to be removed:

```
numItems := pred(numItems);
for k := ptr to numItems do item[k] := item[succ(k)];
```

The same thing can be accomplished using MOVELEFT/RIGHT. In fact, this is the first opportunity to demonstrate the need for and use of MOVERIGHT. Notice that when the insertion is made, array elements are moved up starting at the 'top' (i.e., item[numItems]); when a deletion is made, elements are moved down starting at the 'bottom' (i.e., item[ptr]). Thus, the insertion method requires MOVERIGHT:

```
numItems := succ(numItems);
moveRight(item[ptr],item[succ(ptr)],(numItems - ptr) * sizeOf(ItemRec));
item[ptr] := newItem;
```

While the deletion method requires MOVELEFT:

```
moveLeft(item[succ(pNum)],item[pNum],(numItems - ptr) * sizeOf(ItemRec));
numItems := pred(numItems);
```

Note here that the 'source' and 'destination' parameters are not explicitly byte-oriented; in this case, they are arrays of type 'ItemRec' which can be any type including a Pascal record definition. The UCSD Pascal intrinsic 'sizeOf' returns the number of bytes allocated for any given data type and multiplying this value by the number of elements to be moved supplys the 'length' parameter to MOVELEFT/RIGHT.

The principal factors that influence the speed of insertion are the size of the array elements and the number being moved. Other factors affecting only the conventional technique include whether or not range-checking is disabled and whether or not the iteration variable of the 'for' loop is declared in the first 16 words of the procedure block (see David Gelfand's article in USUS News #13, page 32). As the following table indicates, a significant gain in speed may be achieved in many cases using MOVELEFT/RIGHT:

Speed Ratio of MOVELEFT vs. FOR loop

Size/ bytes	Number of Elements Being Moved				
	10	50	200	1000	10000
2	6.7	19.2	29.1	35.4	36.2
10	4.5	6.7	7.6	7.8	*
30	2.5	2.8	2.9	3.0	*
100	1.2	1.2	1.2	*	*

Trick #3 - Goodbye Type-Checking:

The fact that SCAN and MOVELEFT/RIGHT accept parameters of any type allow for some powerful and flexible uses of these functions. Occaisionally, you may wish to write procedures of your own that allow data of any type to be passed as a parameter. Unfortunately, UCSD Pascal does not go quite so far as to provide a standard type called 'AnyType' for use in procedure parameter declarations... There is a way, however, to implement the equivalent of the following:

```
PROCEDURE MyProc(VAR data: AnyType);
```

Note that a VAR (or variable) parameter simply passes the address of the data... so all that is needed is the ability to obtain the address of the data being passed. The UCSD intrinsic PMACHINE provides this capability (and many more!). Since the p-System's address space is limited to 16-bits, an integer will serve to hold the address. The code required is as follows:

```
CONST STO = 196;   {the PMACHINE 'Store Indirect' opcode}
VAR OneRec: RecType;
    orAddr: integer;   {address of OneRec}
begin
  pMachine(^orAddr,^OneRec,STO);
```

Now the integer 'orAddr' may be passed as a parameter in place of the actual data variable. A general purpose disk IO procedure might look like this:

```
FUNCTION diRead(VAR fileID: diFIB;   {type FILE}
                 address,            {pMachine address of data}
                numWords,            {# of words to read}
                startBlk,            {0..pred(fileLength)}
                startWord: integer   {0..255}
                ): boolean;          {success or failure}
```

Note that 'address' doesn't even have to be a VAR parameter; the procedure is accepting a memory address of data of any type. To use this address within the procedure, a variant record is declared with one variant being an integer and the other being a pointer to any convenient data type. In this case, it may be useful to treat it as an array of word (i.e., INTEGER):

```
TYPE uBuf = array[0..32765] of integer; {i.e., word}

     ptrTrick = record
       case boolean of
         TRUE : (intAdr: integer);
         FALSE: (bufPtr: ^uBuf);
     end;
VAR userData: ptrTrick;
      buffer: packed array[0..511] of integer;   {two blocks}
begin
  diRead := FALSE;   userData.intAdr := address;
  if blockRead(fileID,buffer,2,startBlock) = 2 then
    begin
      diRead := TRUE;
      moveLeft(buffer[startWord],userData.bufPtr^,numWords * 2 {bytes!});
    end;
end; {diRead}
```

(NOTE: For simplicity, the above example will work only if 'numWords' <= 256. Also, if the address is wrong or the variable pointed to by 'address' is too small for the number of words read, disaster will result.)

A similar construct can be used for handling variable sized multi-dimensional arrays. The following function assumes the user data is a one dimensional ARRAY OF INTEGER to be treated as a two-dimensional array with dimensions [1..colMax,1..rowMax]. It returns the value of the cell who's coordinate is 'col,row':

```
FUNCTION cellGet(address,colMax,rowMax,col,row: integer): integer;
{TYPE and VAR declarations as above}
```

```
begin
  userData.intAdr := address;
  cellGet := userData.bufPtr^[pred(col) * colMax + row];
end; {cellGet}
```

For convenience, functions can be be written to return the address of commonly used data types:

```
FUNCTION IntAdr(VAR intVar: integer): integer;
VAR temp: integer;
begin              {returns address of 'intVar'}
  pMachine(^temp,^intVar,STO);   IntAdr := temp;
end; {IntAdr}

FUNCTION StrAdr(VAR strVar: string ): integer;
VAR temp: integer;
begin              {returns address of 'strVar'}
  pMachine(^temp,^strVar,STO);   StrAdr := temp;
end; {StrAdr}

FUNCTION myAddress(VAR myRecord: myType): integer;
VAR temp: integer;
begin               {returns address of 'myRecord'}
  pMachine(^temp,^myType,STO);   myAddress := temp;
end; {myAddress}
```

These functions are then used in place of the 'address' parameter. 'intVar' or 'strVar' may be the first field in a record in which case the address of the record will be returned. NOTE: This only works if the first field of the record definition is *not* in a variable list. For example:

```
TYPE GoodRec = record
       field1: integer;   {This will work fine}
       field2: integer;
       field3: string;
       etc.
     end;

     BadRec = record
       field1,              {This will NOT work!}
       field2 : integer;
       field3 : string;
       etc.
     end;

if diGet(fileID,IntAdr(GoodRec.field1),recNum) then...   {OK}
if diGet(fileID,IntAdr(BadRec.field1),recNum) then...    {Bad news!!!!}
```

Conclusion:

For those who dare and, at the same time, are very, very careful, there exists a degree of freedom beyond 'given' limitations, even in UCSD Pascal.

A Guide to the TI 99/4a p-System

By: Steve Jacobs

Introduction

When TI discontinued the machine in October, 1893, few of the over two million 99/4 owners were using the p-System; if all 99/4 owners felt like orphans, we p-System users were doubly so. But however isolated we seem there are advantages: the p-System on the 99/4a is a full implementation of a "serious" operating system and as such we are able to do much more with it than TI-BASIC. Also, we can get help from any p-System user, regardless of their hardware. Finally, when the day comes when our machines either die or are outgrown, our programs will be portable to virtually any other machine on the market.

The purpose of this guide is to orient the TI 99/4a user to the p-System in general, and point out the special considerations about its implementation on our machine. It is hoped that this guide will be of value to all who are interested, from the serious programmer to those who are considering purchasing the p-System

Overview of the p-System

The p-System is an operating system -- a set of programs to run the computer and allow us to write and run programs. It is not a language; any language might be available under the p-System. While only Pascal is currently available to us, other users of the p-System can use BASIC, FORTRAN, and Modula-2.

This concept is confusing to many TI 99/4 users because TI Console and Extended BASIC have the operating system built in -- including text editors, disk management programs, etc. -- all in the ROM containing BASIC. In the p-System one addresses all these functions separately, and most are on disk.

The essential part of the p-System for the TI99/4a is the p-Code card (or the old p-Code peripheral). The p-Code card contains some of the system programs in ROM. You must also have the 32K memory expansion. There is currently no way to use the extra 96K RAM available in the 128K card manufactured by Foundation. Another manufacturer has recently announced a 128K card for the TI 99/4a. This card supposedly is sold with a RAM disk simulation program. This might be usable under the p-System and would be very useful for storing programs which heavily use the disk -- especially the Pascal Compiler.

A disk is not absolutely necessary for using the p- System, as programs can be loaded and run from cassette. However, I do not know of any cassette programs, nor do I know of anyone who has used a cassette with the p-System.

The p-Code card includes, in ROM, the p-Code interpreter. This program accepts the user's and system programs in p-Code and executes them by interpreting them into 9900 machine instructions.

Anyone serious about the p-System will have to have at least one and preferably two disk drives. The p- System will support double sided disk drives. If you is going to make use of the p-System for writing programs, then you will find that in a single drive system that your drive had better be double sided; two drives are better and two double sided drives are best. I see little use for a third drive. Does anyone know if the Corcomp double density controller allows double density disks on the p-System (360 single sided, 720 blocks double sided)?

A printer is certainly useful for the serious p-System user and can be fully accessed through the RS232 (and PIO) card. Has anybody had experiences with other RS232 and PIO cards, peripherals, etc. -- by 3rd party manufacturers?

There are several programs that are important parts of the p-System. One is the Pascal compiler, the program that takes Pascal programs and translates them to p-Code, so the p-Code card can run them. This

program comes on disk and is necessary if you want to write your own programs. You do not need the compiler to run someone else' programs.

Another disk contains two important programs: the Editor and Filer. The Editor is used to edit documents and programs; you only need this to write your own programs although it can even, to some extent, be used for word processing. The Filer manages the disks; virtually any user of the p-System needs this program. Also on this disk are a number of utility programs which are of much value to most users. Any user of the p-System needs this software.

The last disk contains the Assembler and Linker, both of which are necessary to do assembly language programming.

Learning to use the p-System

The TI manuals that come with the system components are pretty good as manuals go, but are not designed to teach the basic use of the system. There are many books about the UCSD p-System and Pascal. Two very useful books are: "Introduction to the UCSD p-System" by Grant and Butah (Sybex), and "The UCSD Pascal Handbook" by Clark and Koehler (Prentice-Hall). The former is an excellent introduction to the use of the p-System. This guide will describe the unique aspects of our implementation by comparing the TI 99/4a system to that described in this book. The latter book is a relatively technical guide to UCSD Pascal with very good examples. There are a number of good books that teach Pascal; a discussion of them is beyond the scope of this guide.

USUS (the UCSD p-System Users' Society) is a good source of help in the use of the system. Many of its members can easily be reached via the MUSUS SIG on Compuserve (PCS-55). The beauty of the p-System is that it, at least in theory, acts exactly the same on all systems; thus, you can get help from any knowledgeable p- System user; MUSUS is a good place to find such people.

The Filer

The Filer works pretty much as described by Grant and Butah. The major exceptions are that some of the prompts are shorter and the TI 99/4 only displays 40 columns at once. To see the whole 80 column display one uses the FCTN-7 and FCTN-8 keys (screen left and right).

It should be noted that disks may be physically in the drives but cannot be addressed by their volume names unless they were present at startup of the p-System, or reinitialization (I command) or by invoking the V (Volumes) command of the Filer. This command doesn't just list the on-line diskettes, but formally recognizes them so that they can be accessed by volume name in user programs. If you want to use the FILER but didn't have it in a disk drive at start up, then you have a problem. You cannot invoke the FILER V command to get the system to recognize the disk - because you can't run the FILER. The solution to this catch-22 is to use the system I command and restart the system. On restart, the p-System looks anew at the disk drives, and will then recognize the diskettes loaded in them, and know where to find the FILER, EDITOR, etc. If the p-System is started with FILER disk in the root drive (DSK1) -- and no diskette in other drives (DSK2, DSK3), then you cannot access these drives, even by number (#5:, etc). The system can be made to recognize that these drives exist by running the FILER V command. It directs the system to look at all possible disk drives. Once the system has recognized that the drive is there you can use it normally - even changing diskettes, etc.

The Z (zero) command sets up a new (or zeros an old disk) after it has been formatted by the DFORMAT utility program (or the Disk Manager Cartridge). Single sided disks have 180 (512 byte) blocks, double sided disks are twice that size. Using duplicate directories consumes 4 extra blocks but will often avert disaster when disk failure occurs.

The Editor

The Editor for the 99/4 is that which Grant and Butah call the "Screen Oriented Editor" -- the Editor that is discussed at length in their book. In general the Editor works just like that of Grant and Butah with a few differences that could probably be best described as bugs. The "S" (Same) option of the "F" (Find) and "R" (Replace) command does not work. The copy buffer seems to be emptied at times for reasons that are not obvious. After repeated insertions and deletions of a given page on screen, often the screen image is not correct. The actual text is different. As a result, it is best to use the "V" (Verify) command when in doubt -

it re-displays the current screen page. The fact that the Verify command exists indicates that this problem is known to the designers of the p-System.

The 99/4a's small RAM size allows only a little more than 12k characters in the text buffer. If you need to break down a larger file the AUTOPSY program in the USUS library can be of help. Programs can be written in several parts and strung together with the Pascal compiler {$I} (include file) command.

Compiler

The TI 99/4a Pascal compiler is a full Version IV.0 implementation. It seems to be identical to the compiler described by Clark and Koehler. In its TI version, the Compiler appears to be in four parts: 1) the start-up code; 2) the code to process the Declarations, 3) the code to process the body of procedures, and 4) the code to produce p-Codes. Parts 2 and 3 are swapped in and out for each procedure in the program, making compilation a good time for a coffee break. The swap parameters described by Grant and Butah seem to be inoperative on the TI; I suspect that the compiler is set for maximum swapping to accommodate the TI's limited RAM. Making a listing causes the already slow compiler to run much slower still , so one should be sparing in using this function. Also, if errors are found by the compiler it may not make a listing, even if the user keeps continuing compilation to the end. It may even hang with the error STACK OVERFLOW*REBOOT. At this point, one must turn the console off and on again and to restart.

The compiler would be the ideal candidate for the use of a RAM disk program -- if one could be made to work on the system. The excessive swapping would be made painless if done at RAM speed.

The degree of swapping experienced by the Compiler could be reduced by decreasing the number of Procedures and Functions. One could replace them, to some extent, by frequent use of GOTO statements. In other words, by avoiding the use of "structured programming" one could speed the compilation of programs. This is certainly contrary to the philosophy of Pascal and would make the programs much harder to maintain. So it is much better to put up with slow compilation of good programs then fast compilation of bad programs!

The TI 99/4 version of Pascal includes a number of extensions that allow the use of the special capabilities of the machine. These extensions are in the form of procedures included in the "TEXAS INSTRUMENTS UNITS", which are in the system LIBRARY. These include the routines to change screen color, use graphic modes (bit map, multi-color, etc), sprites, sound, joysticks and speech. The use of these routines will allow the sophisticated user to do in a high level language what otherwise was previously only possible in Assembly. TI FORTH posesses these capabilities as well. Please note, however, that the use of these procedures will make the program machine dependent, and thus not portable. One cannot, for example, expect a program using sprites to run properly on an APPLE, because that machine lacks the hardware to produce them. To keep a program portable, one should avoid using the facilities in the TEXAS INSTRUMENTS UNITS.

Among the routines included in these UNITS, I have only used the procedures for random number generation and screen color definition. These procedures work as specified. I welcome input from others who have used the other facilities.

For program development, it is desirable to be able to quickly go from compiler to editor, so it is good to have both programs on one disk or both on different disks on different drives (in a multi-drive system). Also, put the file SYSTEM.PASCAL on the ROOT volume so that meaningful error messages are displayed.

Utilities

There are a number of valuable programs included with the Editor and Filer disk. This guide will discuss the most useful of these programs.

DFORMAT formats new disks (or reformats old ones). It serves the same function as the Disk Manager module in this function. It can format double sided disks but usually fails (I was told by TI that they knew of this bug) but works fine on single sided disks. For double sided disks, one can always use the Disk Manager 2 module if available. The Z)ero command in the FILER must still be used in either case before a disk is usable.

MARKDUPDIR and COPYDUPDIR are useful in dealing with duplicate directories on disk, whose use is highly recommended.

MODRS232 alters the device definition associated with REMIN: and REMOUT: (these use the same definition) and PRINTER:. TI has kindly included the source code for these programs. If you use a printer whose definition is different from the default (RS232/2.BA=9600.DA=7.PA=O.EC -- this probably isn't what you're using), you can alter the source code to make the printer definition without user prompting and run it as SYSTEM.STARTUP. The rules for describing the RS232 or PIO communication setup are the same as used by BASIC.

RECOVER is useful if an both directories are blown (or you foolishly used only single directories). It can help restore lost files. PATCH will allow repair of individual files. You have to be a very expert user to use PATCH, however.

SETLTYPE allows one to force a program when loaded from disk in RAM to be run to reside in RAM rather than VDPRAM. Its stated purpose is to declare whether a program is in p-Codes or native code (machine language). Machine language programs must reside in RAM; p-Code programs may be in RAM or VDPRAM as the p-System selects (usually in VDPRAM).

To understand the significance of the use of SETLTYPE you need to understand how the TI p-System manages RAM. The system views both VDPRAM and RAM as one large space to be filled with programs and data. It will store data down from the high end of RAM and programs up from the low end of VDPRAM until RAM or VDPRAM is filled; then it will overflow into the other space (i.e. programs will overflow in RAM or data into VDPRAM). Apparently, the p-Code interpreter must operate on p-Codes in RAM (this is certainly the only way such a program could work within the architecture of the 99/4 system). In order to run a p-Code program the interpreter must first direct that each code be copied from VDPRAM to RAM before it can be operated upon. Thus if a p- Code program is placed in RAM it will run faster by avoiding the copying from VDPRAM. One can use SETLTYPE to force a program or segment to be in RAM. If this is done it will run somewhat faster, but may impinge on (and thereby limit) the area for data. As mentioned above, assembly language programs must be put in RAM with SETLTYPE.

Using the p-System

Several notes can be made about using the p-System on the TI99/4a. First, you cannot use the TI LOGO cartridge on a system with the p-Code card in place. The p-System will work fine; LOGO will, in certain circumstances, loose control to the p-Code card, destroying what you were doing with LOGO. If your p-Code card has an on/off switch, all will be well if you turn the card off before using LOGO.

When a 99/4 is powered up with the p-Code card is in place (and turned on) the p-System initializes and then displays the p-System greeting. The p-System remains active until the user stops it; thereafter it will not restart without using the technique below (or turning off the system). To get from the the p-System to the BASIC environment is simple - use the p-System 'H' command. To go from BASIC to the p-System, you need to have one of the following cartridges in place on the console:

1. Extended BASIC
2. Editor/Assembler
3. Mini Memory

From console or Extended basic, execute the following command: CALL LOAD(14586,0,0) and then type fctn-= (Quit) or the BASIC command 'BYE'. The p-System will then operate as if you had turned the system off and then on again. Some programs and cartridges seem to have the same effect as the CALL LOAD listed above. Exiting TI-WRITER, MULTIPLAN, or the COMPANION word processor will lead to restart of the p-System. Likewise, some of the commands in the DISK MANAGER will reset the p-System so that it will become active again upon leaving this module.

Speed is an issue of interest to all potential users of the p-System. If computing speed is the reason for the TI BASIC programmer to consider the p-System, the following facts apply: Because the p-System is interpreted, it is relatively slow compared to compiled Pascals (such as Microsoft and TURBO Pascal on MSDOS and CP/M systems). Compared to other machines, the p-System on the 99/4 is quite slow - I

believe it is the slowest implementation of the p-System around. TI BASIC, however, can be benchmarked with a sundial, so it is easy for the p-System to beat. Number crunching programs Pascal can run up to 60 times faster than their BASIC counterparts, especially if put in RAM (see SETLTYPE). Programs doing mostly screen I/O can be slower than BASIC, however. Disk I/O seems a little faster with the p- System -- perhaps because of the simpler file system, leading to less searches for the next block, etc. I have no experience with FORTH, but know that it too is faster than BASIC. The programmer desiring speed above all else should compare the p-System to FORTH.

On running the V command in the FILER program, you will notice that there is a device #14 called 'OS:'. If you do not have a disk in drive one (the usual root device), one will find that the 'PREFIX' and 'ROOT' devices are defined as 'OS'. This device includes files that are in ROM and are used for booting the system. You can read and examine these files but not, of course, modify them. To change the PREFIX definition, use the FILER P command; to change the ROOT definition you must use the system I (warm start) command.

Compatibility with other Machines

The p-System on the 99/4 is very compatible with other p-System implementations. Source code developed on the 99/4 can and does run without modification on most other machines if you avoid using the TEXAS INSTRUMENTS UNITS included with the Pascal Compiler. The 99/4 is less able to run programs from other machines due to it's relatively small RAM size. If the program is properly segmented, however, there will be no problem. The Pascal compiler, for instance, is 99 blocks long -- 49.5K. It runs on the 99/4 (with a lot of disk swapping). The only other problem in running program from other machines on the TI is the presence of a 40 (rather than 80) column display. The TI will display all 80 columns with the SCREEN RIGHT and LEFT keys, but you will probably want to modify the display (if possible) to fit the screen. In the file SCREENOPS.CODE on device #14 (see above) is a description of the TI screen size. This file can be read by a program. This file tells the program in the case of the TI that the screen is 40 columns. Clever programs will adapt themselves to the size of the screen.

A larger problem is physically transferring programs to and from the 99/4a. The TI p-System lacks any functional communications program at this time. Files can be exported from the 99/4a easily by using the FILER to T)ransfer to REMOUT:.

There is good but not complete compatibility with other (non UCSD) Pascals. I have programs which were written on the TI p-System which I ported to an MSDOS (IBM PC or compatible) machine and converted to Turbo Pascal. Little modification was necessary. Has anybody had experience with other Pascals?

Software for the TI p-System

I know of no commercial software for the TI p-System. I suspect that if enough interest could be shown, then the whole spectrum of existing p-System software would be made available.

USUS maintains a large library of programs, most of which are not machine dependent. These volumes are available to USUS members for a small fee - in TI format. Most programs have not been modified to run on the TI. If you modify a USUS program to run on the 99/4, you are requested to send a copy back to the USUS TI distributor.

Wish List

The p-System on the TI 99/4a is powerful but remains largely undeveloped. What follows are my suggestions for software development which would greatly aid the use of the system.

FORTRAN: could someone port this compiler to the TI? There would surely be great interest in this language on the 99/4. Likewise, many would welcome Modula-2.

Terminal Emulation: There have been programmers who have had TE programs "almost working". This is probably the highest priority for the p-System. A TE program would allow porting of other programs to the TI without re- keying the source code.

A good Data Base: there are no high quality data base programs available for the 99/4 at all - not in any operating system. Several excellent programs are available on the p-System. One could one be ported easily

to the 99.

HELP!

The information in this guide is essentially all from my own meager experiences with some help from MUSUS and Compuserve TI Forum members.. Like most p-System users on the 99/4a, I doubt that I have ever seen another user. This guide has been written, therefore, in a vacuum. Its purpose is to catalyze discussion, debate, and the creation of a better guide to the TI p-System.

To improve on this guide, I need the experiences of other users, both the sophisticated and the novice.

This guide could be improved with the following information: Is there anybody out there who can help with machine language programming on the system? How about how to change character sets? Anybody have any suggestions or other tricks for use of the p-System? I solicit your comments, suggestions, and criticisms. If you feel the urge, trash this guide and write your own. We all will benefit by the interchange of information and ideas.

An Overview of MOSYS

By: **Tom Cattrall**

Introduction

MOSYS was put together from pieces developed by Robinson Systems Engineering Ltd, TDI Ltd, PRG Oxford University (all in England) and ETH & ETZ Institutes Zurich. It is marketed by 32DOS Ltd in England and by Pinnacle Systems and Maritime Infosystems in the U.S.

I have had MOSYS for about 4 months but my experience with it is not extensive because of other projects. This article therefore concentrates more on what is available rather than a thorough evaluation of every feature in the MOSYS system.

What You Get

My system came from Pinnacle and consists of the following:

Four floppy disks: MOSYS boot disk
MOSYS release files (2)
MOSYS BIOS and utilities

Four Manuals: Users manual
Programmers manual
Editor manual Document
processor manual

Pinnacle specific release notes.

Installation consists of configuring the MOSYS BIOS using the Pinnacle p- system UTIL program and then transferring the boot disk to a partition reserved for MOSYS. You then boot up from that hard disk partition and type in a series of commands to initialize the file system and to configure for the correct terminal. The system comes set up for a minimal terminal and they supply modules for four other terminals (Qume QVT 102, Wyse WY50, Butel KDS7362, and DEC VT100). The last step in generating a system for another terminal is to do 9 file renames. The instructions only gave 7 names. MI100A and DP100A must be added to the top of the list. You then copy the two floppy disks of files to the winchester disk as the last

step.

I have a Wyse WY-75 and use the VT100 option. My only experience with any of the other terminal options is some brief use of the WY-50 system running at Stride Faire. Source is supplied for the minimal terminal keyboard and screen modules so it is possible to create modules to support other terminals using these modules as a basis.

The major pieces of the MOSYS system are:

Command line user interface
Menu mode user interface using overlapping windows
File System
Screen Editor
Modula-2 native code compiler based on ETH Zurich compiler
Linker and Debugger
Document Processor that formats text (ie. for manuals or reports)
EBNF processor that validates Extended Backus Naur-Formalism files
Macro processor used to define and manipulate keyboard macros
PATCH: display and/or change files
PSYSTOMOSYS: copy files from p-system disks into MOSYS
XREF: produces cross referenced listings of program source files
Definition Module source for much of the system
26 utility programs (commands)

The File System

The file system is a giant step up from the p-system. Files are labelled with a device name (3 characters:floppy drive number, winchester partition, or ram disk), a qualifier (6 characters), description (11 characters), and extension (3 characters).

Example: wi0:AL100A/WindowsDemo.mod

This file is on winchester partition 0 with a qualifier of AL100A and name of WindowsDemo. The mod extension is the standard extension for Modula-2 source files. There are 23 extension conventions but you can use any 3 characters. Notice that filenames are case sensitive (as is Modula-2). My little fingers got used to it a lot faster than I expected.

You specify the number of files on a device with the init command (I have 2000 on wi0:) and then just create them as needed using qualifiers to group them. A qualifier can have as many files as you need: 1 or 1000. MOSYS doesn't care.

MOSYS has a very useful search device feature that is a superset of the p- system Prefix. You can define pseudo devices that contain a list of real devices and qualifiers. Then whenever a filename is given to the system without a qualifier the file system tries device/qualifier pairs from the assign list until it gets a match.

Wildcard characters are supplied for any sequence of characters (*) or an individual character (%). I sometimes "lose" files on the p-system. That is, I know the name but can't find which subvolume holds it. As near as I can tell the only way to find it is search each subvolume. In MOSYS I just do:

DIR */name.*

and it checks all qualifiers and shows a listing entry for each one it finds.

The file directory list shows date/time of last modification, backup and protected flags, size, and location on disk. No backup utility is supplied (I use copy) and as far as I can tell there is no command to set or clear the file protected flag. It would be a simple thing to write such a command using the supplied procedures in the module Files.

Commands are supplied to do the normal things such as create, delete, rename, copy, directory list, initialize, consolidate (Krunch). I couldn't get the utilities that configure the disk drive or format floppies

to work. I use the p-system to format floppies and then do an init in MOSYS to write a MOSYS directory. I used the p-system UTIL to configure the disk drive to 1600 block format and have not tried any other format.

The User Interface

MOSYS has a normal command line mode where it issues a prompt "=>" and then waits for you to type a command. To delete a file you would type:

delete Prog/transp.mod

Commands are not case sensitive.

The other mode is the menu user interface. This consists of a series of menu windows in a tree structure. Commands that are used together are grouped in the same menu. Some functions such as file system commands that are useful in many environments appear in several different menus. The main menu and the program development menu are shown below to give an idea of the structure. You select a menu option by going to the desired line with up and down arrow keys and then press right arrow to select that option. You can also type the number of the desired menu item and it is automatically selected. The > symbol on some items says that that item is another menu rather than a command. When you select another menu it overlays the existing one in such a manner that you can see where you came from. The maximum stacking level is 16 windows. To return to the previous menu press the left arrow key. At any time you can access a universal menu by pressing the space bar, or call up a help screen by pressing the help key.

```
   MOSYS     1.00a          >          PROGRAM DEVELOPMENT
1. System Restart                   1. Directories and Files     >
2. User Environments                2. Edit a File
3. System Closedown                 3. Compile a Module
4. Program Development         >    4. Link a Program
5. Document Processing         >    5. Execute a Program
6. System Configuration        >    6. Debug a Program           >
7. Develop User Interface      >    7. Cross Reference a Module
8. Command Line Environment
9. Execute Command File
```

The main menu on the left is what you see when first entering menu mode. Selecting item 4 would give you the program development menu shown on the right. It would overlay the main menu on the screen. Notice in the program development menu that items 1 and 6 would select another menu level while the remaining items are all commands that execute directly.

A dialog window appears when the menu item selected is a command that needs parameters. Prompts show what is needed and limited editting of the parameters is allowed. Parameters are remembered and often times all you do is hit return to accept the existing parameters. An example: after entering a file name to edit it is not necessary to reenter it for subsequent edit, compile, link, or execute steps.

The menu interface has the curious "feature" that it tends to ignore alternate commands. Edit, compile, link, execute can be run in order without a problem, but trying to execute several times in a row only works every other time.

This user interface allows a newcomer to MOSYS to do things almost immediately and yet it is also very useful to the experienced user. I prefer the command mode for a few things but use the menu interpreter most of the time. All of this is done using an interpreted language that you type into a text file and then run through a preprocessor. Similar systems could be set up for user application environments. The source for the MOSYS interface is supplied and you could add to it or do a new one. I like the concept so well that I plan to implement a similar scheme on another computer system where computer illiterate people have to manage an extensive set of commands and parameters.

The Editor

I am not very fond of the editor. If MOSYS is to prosper I feel that it needs a better editor. It seems to be patterned after DEC's EDT. Commands are entered by operating control keys on the keyboard. For

VT100 terminals this means the PF1-PF4 and numeric keypad keys. The Wyse WY-50 uses the function keys on the top of the keyboard. With a minimal terminal you hit escape and then another key. My first complaint is having to remove my hand from the keyboard to hit a pair of keys on the keypad. The second complaint is that there aren't very many commands once you get there. In VT100 mode several of the commands don't work but the whole thing is so primitive I don't even miss them. Maybe I shouldn't be so negative about it. Back in the good old days I edited Fortran code on 026 keypunches and got by. This is a definite improvement over that.

Anyway, the commands are:

```
movement:       arrows for up, down, left, right
                word left/right, line left/right, line next/prev
                page next/prev, tab
                home                    (doesn't work on VT100)

delete:         char left/right,  word left/right, line left/right
                line,  undelete

cut/paste:      mark, cut, paste        (don't work on VT100)

find/replace:   define search string    (doesn't work on VT100)
                Find next/prev          (might work if I could define string)
                find error next/prev    (nicest feature in the editor)
                replace                 (might work if I could define string)

escape:         escape to menu window that gives commands below:

          abort, end-edit, write file,
          Top of file, bottom of file, centralize
          read/write text from/to file
          5 misc and utility commands
```

The editor is always in insert mode, there is no overwrite mode. It pushes existing text along without making a gap first as the p-system editor does. This makes it less than clear what you are doing while entering new code. I often create an extra blank line or two and then go back and do my insert before the blank line. When done I delete the blank line. If the cursor is in column 70 and you press the down arrow the cursor moves down to the next line with something in column 70. This could be 2 lines later or 200. The manual says this "may be slightly surprising to the novice user". After 4 months of putting up with this "feature" I must say surprise isn't the emotion I feel when this happens.

Easily the best feature of the editor is the way it interacts with the compiler for compile time errors. If the compiler encounters errors it writes an error file that contains a list of error numbers and the line and column where each one occurs. When you go back to the editor it reads the error file and puts a mark (default is @) at the location of each error that the compiler found and positions itself at the first one. The text for that error message is displayed on the bottom line of the screen. The find error keys will position the cursor at the next or previous error in the file. At each error the corresponding error message appears at the bottom. The same thing happens if you move to an error location using the arrow keys. When you leave the editor the @ characters are removed and the error file is deleted. With this feature you can fix a lot of stuff with each compile.

The Compiler, Linker, and Debugger

Many compilers for small systems come without a full manual. They only have a description of the non standard or missing features in the compiler. The MOSYS compiler doesn't even have that. You get a description of what parameters you need to invoke the compiler, a list of the various files that go in and out during compilation, a list of the compiler options that may be imbedded in the source, and a short blurb on module keys and what happens when they don't match. I haven't really felt a need for any further documentation since everything seems to be there. The only trial and error I did was to discover that the compiler only allows 64k of data. I wasn't ambitious enough to see how much code you can have. The books by Wirth and Gleaves have done the job. The only compiler error I have run across is minor (it says TYPE

short= -32767..32767; is too big for it). My experience with it hasn't been very extensive and I have seen others find more serious bugs.

The code produced is relocatable native code that goes into the linker. It does an uninspired but decent job of generating code. Integers and cardinals are only 16 bits now but they plan to support 32 bit data types as well as 32 bit address space. Code not using reals seems to run around 10 times faster than the equivalent p-code. Reals are 32 bits supported by software and are faster than the p-system reals using 64 bit software by a factor of 4.

Two utilities are supplied that disassemble compiler output files. DECLNK converts link files into assembly code and DECSYM converts symbol files (output from a DEF mod compilation) back into Modula-2.

The compiler is multipass and isn't especially fast but the ability to get all errors reported at once helps a lot.

The linker combines compiler output files into an executable object file. It does module key checking along the way. There are options for getting a map, including information for the debugger, and creating a file that is to run in a standalone mode at a specified address.

Dumps may be taken when a program aborts by turning on a dump mode. After you get a dump, the debugger is used to analyze it. The debugger lets you look at the aftermath through 4 different windows:

P window: shows chain of activated procedures at moment of crash
D window: view actual values of variables belonging to a procedure
or a module.
T window: view source file of a seperate module.
C window: view dump file core image

I haven't done enough with the debugger to know if it all works as advertised. A full blown run time debugger that allows breakpoints and source level interaction would be a nice next step for MOSYS.

Application Programs

MOSYS supplies 7 programs that the manuals call application programs. They are:

TEXTPROC takes files prepared using the editor, and using imbedded commands produces a formatted output file. I haven't used it but as all of the manuals were done with it I assume it works. Any line beginning with a decimal point is a command line. The command groups are: Page format, Pagination, Line layout, Line spacing, Text heading, special characters, and macro commands.

DOCPRINT Used to print files produced by TEXTPROC. Bold and underlining attributes are supported by the printer modules. The printer module source is provided so that special needs may be provided for.

EBNF This will read an EBNF definition file and verify it. A cross referenced listing is also produced. See Appendix 1 of Programming in Modula-2 by Wirth for an example.

MACROSOPS The system supports keyboard macros of the form \x where x is an upper case letter. These 26 macros are defined and manipulated using MACROSOPS. They supply a default set that includes many Modula-2 words and phrases. You can of course define sets of your own.

PATCH Enables you to examine and change a file in hex, ascii, or characters using a screen editor type arrangement. I haven't had occasion to use it but there have been many times I wished I had something like it on other systems.

PSYSTOMOSYS Allows p-system files to be copied to MOSYS. There is no equivalent to go the other way.

XREF Takes as input a Modula-2 source file and produces a line numbered listing and an identifier cross reference listing.

Supplied Modules

MOSYS comes with source for 50 or so DEFINITION modules. Most of them are commented well enough that you could make use of them in your own programs. I was going to produce some kind of comparison chart between MOSYS, Wirth's book, Volition, and the proposed MODUS standard. After some study I gave it up as a lost cause. A few things like SIN and COS were the same across all 4 but usually I couldn't find even two that would agree. Let's hope that the proposed MODUS standard sweeps the Modula-2 world like wild fire and ends this nonsense. Brian Kirk of Robinson Systems is participating in MODUS so I assume that MOSYS will reflect the standard some day.

InOut is close enough so that simple terminal I/O can be handled with no conversion. Unfortunately, reading (ReadString, ReadInt, ReadCard) doesn't work. They produce varying results such as bus errors or instantly returning with a zero value. Switching to TextIO gave similar results. I finally found that by setting the proper terminal mode I could read a character at a time. Using this and some string to number conversion modules in M2Conversions I wrote my own ReadString, ReadInt, and ReadCard. Writes work OK except that WriteReal gets an out of range trap when writing a value of zero in scientific notation.

Modules are supplied to handle file names and MOSYS style command string prompts and decoding as well as windows and three types of schedulers for concurrent programming.

They supply IMPLEMENTATION module source for the keyboard, screen, and printer drivers. The keyboard and screen sources are for the minimal terminal but with your terminal manual and a little work almost any terminal could be supported. One of these days I'll get tired enough of the VT100 bugs and keyboard layout to make a version more to my liking. The initialization section of the minimal terminal driver prompts for the key to do a home cursor and the key to do non destructive right space. So if you use a terminal with at least those two features you should be able to run MOSYS. After its up and running you could switch to one of the standard terminal types or roll your own.

Support

Nobody at Pinnacle seems to know much about MOSYS so bug reports go from me in Oregon to Pinnacle in Texas and then on to the MOSYS people in Bristol England. So far this has been a one way path. I take it on faith that the bug reports arrive at the other end and are acted upon. I did get to talk to Brian Kirk at Reno and he was already aware of some of the same things that I had found. Maybe they are working frantically on a new version of MOSYS that fixes all of the problems.

Documentation

The documentation consists of four small manuals. They are written in a very minimal style and tell you just enough (or sometimes almost enough) to allow you to do something. That may sound like I don't like them but that's not the case. When I want to look something up I can find what I'm looking for right away and the explanation is short and to the point with no fluff. In some cases a little more explanation is needed but not much more. I much prefer these manuals to the p-system manuals where you have to read whole pages (or sometimes chapters) to find out the most trivial thing. I'm a newcomer to the p-system as well as MOSYS but have used lots of other operating systems before. A dyed in the wool p-system user may have a different opinion.

The only documentation supplied with the modules is in the form of comments in the DEF mod source. Some modules have sufficient comments while others have none. If they assign someone to fill in the gaps that would be enough to keep me happy.

Future Versions

Nothing concrete to report here. They seemed quite emphatic about wanting to have 32 bit data and addressing in a future version. I also heard talk of better code generation such as better use of the 68000's registers. Just using registers for local variables usually gives about a factor of two speed improvement on the 68000. I heard mention of an assembler and multitasking. Of course I'm hoping for fewer bugs.

Summary

MOSYS has quite a few bugs right now. In addition to those mentioned elsewhere, running programs have a tendency to abort with a bus or address exception. One time it became solid and I finally had to reboot. I haven't been able to come up with a reproducible case that fails. Actually there probably aren't as many bugs as it seems. I have just become somewhat gun shy. In any case I wouldn't try to do production work with the present system.

MOSYS needs a better editor. To change 25 occurences of writeln to WriteLn requires that you enter 28 commands rather than one as in most editors. I can't see MOSYS getting very far in the world with this editor.

Except for the editor, the MOSYS development environment is very nice. The menu and file systems are the bright spots in MOSYS and are definite improvements over the p-system.

With MOSYS you can't get spreadsheets, spelling checkers, data base packages, accounts receivable, Basic, C or adventure. MOSYS is for people that produce software not for people that use software. Given Modula-2 and the menu system you could develop some nice turnkey systems without a lot of hassles. I work with real time standalone process control systems and Modula-2 along with the MOSYS ability to produce standalone code would make MOSYS a good tool for developing systems which are then transported to a target computer.

The current Pinnacle BIOS (4.0) doesn't support MOSYS. If you get MOSYS for a Pinnacle make sure you get the 3.0 BIOS or the 5.0 when it becomes available.

MOSYS was bundled in with my Pinnacle system and didn't really cost any extra. I hadn't expected to use Modula-2 for 6 months or so and now it looks like it'll be farther off yet. So I have been able to play with MOSYS without the pressure of needing it to get a job done. The current version has some problems but I still like the system.

A future MOSYS system with fewer bugs, a decent editor, and the ability to use the 68000's memory addressing capabilities will be a very nice tool for Modula-2 developers.

Implementing the (Enhanced) Kernighan & Plauger Tools in Modula-2

By: **Dennis Cohen**

Most of us that are interested in programming have come across at least one of the books "Software Tools" and "Software Tools in Pascal" which were written by Brian Kernighan and P.J. Plauger and published by Addison-Wesley. They are generally considered to be excellent illustrations of modular software construction with useful resultant programs. Because of this, I decided that Modula-2 would be even superior to Pascal as an implementation language for these tools and proceeded to test my hypothesis. As a quick preview, I succeeded beyond my wildest expectations.

Because the K&P tools are primarily versions of U**x utilities and because I had the source to the pShell (from doing the Apple /// {SOS} port), which was a mini-unix, I decided that I had a good starting

point.

The first thing that I wanted to change from the K&P implementation was the treatment of characters. They added an extra layer of translation at all points by treating characters as their ASCII values and manipulating them as integers. This is probably due to their C and FORTRAN backgrounds. I wanted characters to be manipulated as characters and to eliminate the extra layer of translation on input and output. Besides all this, Volition had done a good job of implementing their Library and I felt that it would be foolish not to take advantage of this work.

After I had gone through the procedures and functions which K&P classed as "primitives" and "utilities," I determined that most of them were going to need rewriting almost from scratch. This was not surprising considering the amount of diversity between the Pascal implementations; however, I did not expect them to differ as much from the UCSD Pascal implementation as they did. At any rate, I paced my way through "Software Tools in Pascal" redesigning the low-level interfaces as I went. Fortunately, the interface was not where the majority of the differences lay. Because of this and Modula-2's support of piece-wise development, I was able to rewrite some of the simple tools using the newly-created definition modules and finding out how to improve the interface.

From there I went for the throat -- the text-formatter. At this time I started implementing the primitives and utilities. Within one week, I had the defmods and impmods for the primitives and utilities and a couple of simple tools operational. By the end of the second week, I had completed the task that I had set for myself -- I had the tools that interested me (format, ar, more, translit, change, sort, and uniq) implemented, including most of the recommended enhancements. When bound into one code file with the pShell this all came to only 42K (84 blocks). This compares with over 100 blocks under UCSD Pascal and there I did not have the piping and redirection provided by the pShell nor the other utilities. I had also reimplemented grep to provide real pattern- matching as well as a few output options.

If you are interested in this material, contact Volition Systems (or whoever acquires the rights from them) as they have purchased the distribution rights. I still maintain publication rights and am continuing enhancement as time permits. Because of this, you might find an article or series appearing someday with the source listings (if there are any publishers interested).

Pascal Assistant, a Fancy ASE Macro Module

By: **Richard Karpinski**

Volition's Advanced System Editor (ASE) is a descendent of the popular screen editor native to UCSD Pascal. In addition to handling files of unlimited size, ASE exhibits many other improvements. Perhaps the most interesting new feature is the provision for user defined function keys. When a function key is loaded with a sequence of ASE commands and data to be executed by a single command, it is said to hold a macro-instruction. These macros are the subject of this series of columns.

Abstract: Model statements and a skeleton programs are provided which,
together with the ASE macros, assist the programmer in creating
readable, indented Pascal programs.

From time to time one or another programmer builds a text editor for programs which assists in the construction of proper (syntactically correct) programs. Usually, the result is not seen as very handy for programmers to use. There may be many reasons, but often it is that improper programs are never permitted. Thus many modifications are made more difficult than with a conventional editor.

This column offers some convenience features, but not the constraints usually associated with language knowledgable editors. It implements a set of model statements in the Pascal language by using "macro" or "function keys" provided by the Advanced System Editor under the UCSD Pascal system.

Sample result

Figure 1 shows an example with each assisted statement. The indenting style and all shown comments are built into the model statements (see below). You will have the opportunity to change them for your style of indenting, or to assist with some other language, or for some other function entirely.

The created program has no particular meaning. In fact, since functions without value are not permitted, the program is not even correct. It is however, fairly tidy. It was very easy to do. Each addition was made by pressing function key 2, typing a single letter to select the statement, and pressing function key 3.

Each such addition was immediately followed by tailoring. A R(eplace command was automatically begun to supply the name or expression needed in this instance. (You also type the closing " \".)

```
------------------ Figure 1 - Sample ---------------------

procedure MyOwnProcedure;

   function MyOwnFunction(  ) : ;
      begin { MyOwnFunction }
         if Condition-X
            then begin
               while Cond-Y do begin
                  end { while Cond-Y };
            end {True Condition-X}
            else begin
               repeat { until Cond-Z }
                  until Cond-Z;
            end {False Condition-X};
         end { MyOwnFunction };

   begin { MyOwnProcedure }
      case Quantity-Q of
         Val-A :
            begin
               end { Val-A };
         Val-B :
            begin
               end { Val-B };
         end {case Quantity-Q };
      end { MyOwnProcedure };

------------------ Figure 1 - Sample ---------------------
```

Notice that each structured statement reminds the reader at the end about how it began. This is intended to make the reader feel secure about understanding the structure of the program as the details are investigated. It is this repetition which aids the reader that can be effortlessly added by means of an automated program builder. Perhaps it will be easier to write programs which are easy to read.

Menu selection mechanism

This particular program builder uses two function keys to perform a menu selection. The menu page itself which is visible while the selection is made can be quite arbitrarily arranged. It can even mislead, on purpose as well as by the usual accidental route. Most importantly, however, the page may easily be changed by the user, using only the familiar tools of ASE itself. In fact, the entire mechanism is (necessarily) visible and accessable for change to anyone who studies it appropriately.

The two key menu selection could be used for other purposes. Perhaps, some time in the future, when hard disks are a matter of course, on line help in using or learning ASE could be supplied by such menus. For now, it is enough to see that separate menus, sub-menus and other extensions of that sort are easy to accomplish.

Since the macros have to be saved anyway, it may be appropriate to put a little work into a standard skeleton for a program. The point of this exercise is to build a framework which contains all the standard (boiler- plate) stuff to preserve copyright and so forth. It should serve as a checklist for your style of building programs. Change it at will. The macros under discussion are at the end.

Skeleton Pascal Program

```
program Prog; { Example skeleton program for starting from. }

   const

   Title   = 'Skeleton sample UCSD Pascal program for ASE.';  { These lines  }
   Author  = 'Richard H. Karpinski';                          { are printed    }
   Version = '23 Jul 85 for exemplary use';                 { at the beginning  }
   Notice  = 'Copyright 1985 R. H. Karpinski';            { of each output report }
   Rights  = 'All rights reserved.';                    { made by this program.   }

(* Table of Contents:

   - Title, Author, etc. (above)
   - Table of Contents (this part)
   - Log of changes
   - $PROFILE and housekeeping
   - Table of markers
   - Marker setting macros
   - The program itself
      - Global constants, types & variables
      - Global functions & procedures
      - The outer program block
   - Model statements for ammending program


                     (see marker setting macros below)
Log of changes made:  (the marker $LOG should point here)

01 Jun 85 Improved UPCASE and locase to use comma/blank lists
01 Jun 85 Tailored for column in USUS News
16 Apr 83 $profile gets NF, U(pcase & L(owcase simplified
17 Nov 82 Many minor changes, menu paragraphs added, tail extended
06 Oct 82 Moved selection screen below $forms to avoid reverse scroll.
03 Oct 82 Noted problem with U(pcase and L(owcase long function key chain.
03 Oct 82 Got "p9c@r"  off user selection screen.
03 Oct 82 Upcase and Lowcase added: chains of function key definitions
03 Oct 82 First draft complete.  Markers and function keys defined.
          These remarks are requested by ASE at the end of a session.


                          (see marker setting macros below)
$PROFILE and housekeeping:   (the marker $PROFILE should point to first "|".)

This defines function key 1 whenever this document is edited.

|s~             |{ Real spaces shown as ~, others are for readability }
|"profile"      |{ Name the function key for the prompt line }
|x              |{ Execute this definition whenever it is "taken up"  }
NF              |{ Ensure definitions in page buffer by Next Forward }
|* |f2 |* |f3   |{ Take up function keys 2 and 3 }
JM $version |n   |{ Find version constant for program }
X|f8 |e         |{ eXchange todays date (in function key 8) on top of old one }
```

```
????            |{ Show function key names on user's prompt line }
|.

|"use form"|s~  |{ <f2> : ignore spaces, title the function key }
SM $where |n    |{ set marker at point where statement belongs}
JM $forms |n    |{ go to "picker" }
4 |n U          |{ go up to show prompts for direct user editing }
g:~~ I          |{ go to after the ":" and leave user inserting }
|.

|"select"|s~   |{ <f3> : escape the insertion of users selection }
|!             |{ escape the insertion of users selection }
JM $forms |n    |{ go to where the "picker" is }
g%~ CB          |{ go to after "%", and insert users choice }
b |* |f6        |{ go to beginning of line and take up as <f6> }
b g%~ Dg/ |e    |{ go to after "%", delete users choice and accept }
|f6             |{ execute key 6 }
|.
```

(see marker setting macros below)

Table of markers: ($ marks predefined markers so you can use plain names)

$CURSOR	marks the cursor location when editing was last suspended
$EQUAL	marks the other end of what Z(ap would delete
$FORMS	marks the "picker" function key definition for model statements
$LAST	(not in use) marks the cursor location at last Q(uit so that editor automatically prepares to resume there next session
$LOG	marks the top of the log of changes made
$PROFILE	marks the initialization function key definition
$TAG	marks the last place marked by S(et T(ag, used by J(ump T(ag
$TAIL	marks the end of the last macro selected with select & picker
$WHERE	marks the place where a new statement form is to be installed

Marker setting macros (for those who download this)

```
|s~ |x ST JF     |{ auto-execute me, set tag (marker $TAG), go to top }
F/$PROFILE~should/    |{ find a string }
F/||s/ =         |{ find the beginning of the profile macro }
SM $PROFILE |n   |{ set marker $PROFILE there }
JT |* |f1        |{ return to marker setters and takeup the next }
|.

|s~ |x ST JF     |{ auto-execute me, set tag (marker $TAG), go to top }
F/$LOG~should/   |{ find a string }
2 |n SM $LOG |n |{ down 2 lines and set marker $LOG there }
JT |* |f1        |{ return to marker setters and takeup the next }
|.

|s~ |x ST JF     |{ auto-execute me, set tag (marker $TAG), go to top }
F/"picker"~FCT/  |{ find a string }
B SM $FORMS |n   |{ go to beginning of line and set marker $FORMS there }
JT |* |f1        |{ return to marker setters and takeup the next }
|.

|s~ |x ST JF     |{ auto-execute me, set tag (marker $TAG), go to top }
F/Version~=~'/   |{ find a string }
SM $VERSION |n   |{ set marker $VERSION there }
JT |* |f1        |{ return to marker setters and takeup the next }
|.

|.
```

```
end of housekeeping comment *)

   { Global constants (continued) }
               { comment about the significance of each const you add }

   ProgramName  = 'Prog'; { For identifying this program as responsible }

   { Global types }
   type        { comment about signifcance to program of each new type }

   Rec         = record      { used for disk data file }
      ID          : integer; { matches number on form Q }
      Bal         : real;    { brownie points accumulated to date }
                    end;

   { Global variables }
   var         { comment about signifcance of each variable definition }

   Buf         : Rec; { holds record from old master file }
   NewRec      : Rec; { record being built for new master file }


   procedure Initialize; { show Title, Author, etc. }
      begin { Initialize }
         writeln;
         writeln( Title );
         writeln;
         writeln( Author );
         writeln;
         writeln( Version );
         writeln;
         writeln( Notice );
         writeln;
         writeln( Rights );

         { add initialization code here }

         end { Initialize };

   begin { Prog }
      Initialize;

      { put main line, or top level, code here }

      end { Prog }.

Material after the "." (like this) is ignored by the UCSD Pascal compiler.
```

Pascal Assistant:

Finally, after a few comments and the picker macro, we see the selection menu as it will appear on the screen. The "use form" and "select" macros were shown above in the housekeeping section of the skeleton program. The remaining (model statement) macros follow the display.

The macro which follows is used to pick one of the following forms by typing a single letter (or word) and pressing <f3>.

```
|s~ |"picker" FCT/%/ |n |* |f7 SM $tail |n JM $where |n |f7 |.

User Selection screen:

                                        You want a screen title, maybe?
```

```
Your choice: -

INSTRUCTIONS:  1. Choose one of the following list.
               2. Type the first letter of your choice.
               3. Press function key 3.  (Many things happen here.)
               4. Complete the R(eplace (if any) with the intended value.
               5. Type a backslash " \" to let the replacement happen.

N(othing    resume editing, I didn't mean it
P(rocedure  install a new procedure where I showed you
F(unction   install a new function there
I(f         insert an if then else statement
W(hile      insert a while loop
R(epeat     insert a repeat loop
C(ase       insert a case selection
V(alue      insert a case element
U(pcase     revise keywords to be UPPER case (it could take an hour)
L(owcase    revise keywords to be lower case

                                          You can change this yourself!


%N %n                              Nothing
|s~ |"nothing" |.

%P %p                              Procedure definition
|s~ |"procedure"
I |n                          |{ Insert a bunch of text }
procedure~Proc; |n            |{ |n is a return, ~ is a space }
~~~begin~{~Proc~} |n          |{ more text }
   ~~~end~{~Proc~};           |{ more text }
|n |n |e                      |{ two returns and accept the insertion }
=                             |{ go back to the beginning of the procedure }
3R\Proc \\                    |{ replace the 3 "Proc" instances with a name }
|.

%F %f                              Function definition
|s~ |"function"
I |n                          |{ Insert a bunch of text }
function~Func(~~)~:~; |n      |{ more text }
~~~begin~{~Func~} |n          |{ more text }
   ~~~end~{~Func~};           |{ more text }
|n |n |e                      |{ two returns and accept the insertion }
=                             |{ go back to the beginning of the procedure }
3R\Func \\                    |{ replace the 3 "Func" instances with a name }
|.

%I %i                              If then else statement
|s~ |"if then else"
I                             |{ Insert a bunch of text }
if~C |n                       |{ more text }
~~~then~begin |n              |{ more text }
   ~~~end~{True~C} |n         |{ more text }
|b|b|belse~begin |n           |{ more text }
   ~~~end~{False~C}; |n       |{ more text }
|e =                           |{ accept it and return to the beginning }
3R\C \\                       |{ replace the 3 "C" instances with an expression }
|.


%W %w                              While statement
|s~ |"while"
I                             |{ Insert a bunch of text }
while~Condition~do~begin |n       |{ more text }
```

```
~~~end~{~while~Condition~}; |n  |{ more text }
|e =                        |{ accept the text and return to its start }
2R\Condition \\             |{ replace the 2 "Condition"s with an expression }
|.

%R %r                              Repeat statement
|s~ |"repeat"
I                           |{ Insert a bunch of text }
repeat~{~until~Condition~} |n   |{ more text }
~~~until~Condition; |n           |{ more text }
|e =                        |{ accept the text and return to its start }
2R\Condition \\             |{ replace the 2 "Condition"s with an expression }
|.

%C %c                              Case statement
|s~ |"case"
I                           |{ Insert a bunch of text }
case~Expr~of |n             |{ more text }
~~~end~{case~Expr~}; |n     |{ more text }
|e =                        |{ accept the text and return to its start }
2R\Expr \\                  |{ replace the 2 "Expr"s with an expression }
|.

%V %v                              Value for case statement
|s~ |"value of case"
I                           |{ Insert a bunch of text }
Value~: |n                  |{ text }
~~~begin |n                 |{ more text }
   ~~~end~{~Value~}; |n     |{ more text }
|e =                        |{ accept the text and return to its start }
2R\Value \\                 |{ replace the 2 "Value"s with an expression }
|.

%L %l                              Lowcase keywords
|"locaser"|s~
JM $tail |n      |{ come back here to get the list & macro }
|* |f5 |n |n    |{ takeup <f5> below and skip down to its list }
99 |f5           |{ do it 99 times or until the <f7> value is empty }
|.

|"locase"|s~     |{ locase makes a list of words lower case }
SM $list |n      |{ set a marker there }
D |b g, |!       |{ delete to a comma and escape (copy it to buffer) }
                 |{  the initial backspace allows for zero length element }
|* c |f7         |{ takeup copy-buffer into <f7> }
JM $forms |n <   |{ start from bottom of program and go backwards }
/RCT77           |{ forever replace case-insensitive token-mode <f7> <f7> }
                 |{   i.e. find a <f7> and put <f7> there instead }
                 |{   e.g. find a CASE and put case there instead }
> JM $list |n    |{ return to the list to get the next element }
g, ~ ~           |{ skip the one we did and its comma and blank (or return) }
|.

procedure, function, while, repeat, if, case, begin, end, do,
until, then, else, of, for, to, ,(ends the list)

%U %u                              UPCASE keywords
|"UPCASER"|s~
JM $tail |n      |{ come back here to get the list & macro }
|* |f5 |n |n    |{ takeup <f5> below and skip down to its list }
99 |f5           |{ do it 99 times or until the <f7> value is empty }
|.

|"upcase"|s~     |{ upcase makes a list of words UPPER CASE }
```

```
SM $list |n      |{ set a marker there }
D |b g, |!       |{ delete to a comma and escape (copy it to buffer) }
                 |{  the initial backspace allows for zero length element }
|* c |f7         |{ takeup copy-buffer into <f7> }
JM $forms |n <   |{ start from bottom of program and go backwards }
/RCT77           |{ forever replace case-insensitive token-mode <f7> <f7> }
                 |{   i.e. find a <f7> and put <f7> there instead }
                 |{   e.g. find a case and put CASE there instead }
> JM $list |n    |{ return to the list to get the next element }
g, ~ ~           |{ skip the one we did and its comma and blank (or return) }
|.

PROCEDURE, FUNCTION, WHILE, REPEAT, IF, CASE, BEGIN, END, DO,
UNTIL, THEN, ELSE, OF, FOR, TO, ,(ends the list)
```

Known problems.

There are already known problems with using macros in this way:

1. They are slow running without ramdisk.
   (Write smaller pieces, using UNITS and $Includes.)

2. The models use an indenting style that you may not prefer.
   (Change the models.)

3. A function key definition which lies across the main memory buffer boundary cannot be taken up.
   (Try NF before |*.)

4. Upcase and lowcase stop whenever a replacement fails due to finding no instances to replace.
   (Arrange to put a copy of the full list at, say, the beginning.)

Please send any new problems, workarounds, solutions, and suggestions to:

BITNET : dick@ucsfcca — USPS: R. Karpinski
TeleMail: RKarpinski — Modula Assured Quality Software
Work : (415) 666-4529 (12-7) — 6521 Raymond St.
Home : (415) 658-6803 (9am - 11pm) — Oakland, CA 94609
MUSUS : Dick Karpinski [70215,1277] — (415) 658-3797 (ans. mach.)
USENET : ...ucbvax!ucsfcgl!cca.ucsf!dick

Use any method that is convenient for you, copy the page or write on it and mail it or mention the section or however you like, but if you don't tell me then I won't ever fix it.

ASE Version Differences

By: N. Arley Dealey

Notable differences between 0.9 and 1.0 versions of ASE 25 Feb 85

* Terminal requirements
 - Display must have at least 24 lines and at least 80 columns
  (should actually work on smaller screens but the prompts
  and menus make no concessions to tubes less than 24x80)
  (Note: A virtual screen is ok, therefore standard Apple //s without any
  80 column hardware are ok since Apple implemented a virtual 80 column
  display in the BIOS)
 - Terminal must support both ClearToEndOfLine and ClearToEndOfScreen
 - Your GOTOXY when passed an X parameter beyond the right hand edge of
  the screen must place the cursor at the right hand edge (as recommended
  in the gotoxy specs). This is not actually a new requirement but is,
  I believe, the first time it has been documented.

* File specification in Init
 - Now supports wildcards in the destination file spec. A '$' acts as a
  'right hand' wildcard in either the volume or filename fields. This
  means that the right hand portion of the corresponding field in the source
  file spec will be appended to any chars preceding the wilcard in the
  destination field. Examples:
   AVOL:THISFILE,*$ --> *THISFILE
   AVOL:THISFILE,THAT$ --> THATFILE
   AVOL:THISFILE,$:$ --> AVOL:THISFILE
   AVOL:THISFILE,B$:NEXT$ --> BVOL:NEXTFILE
  Note also that forms like:
   ?,#5:$
  and
   A?,#5:B$
  and even
   A?=Gadzooks,#5:B$
  are legal and handled appropriately (or you can append the wildcarded
  destination after the selected filename is inserted into the response
  field if you prefer)
 - On IV.2= versions, if editor is invoked from the compiler error prompt and
  the error is in an include file, the include file name is passed to the
  editor as the default file to edit.
 - On all versions, if the editor is invoked from the compiler error prompt
  and you select the wrong file you may issue a Q(uit E(xit A(nother command
  sequence and the $Syntax marker info will be carried over into the new file.

* Insert
 - On terminals with InsertLine capability displays full screen context. This
  probably still has some kinks in it so please scream early and often when
  you run across them. Upon abandoning a complete line of the insertion

(either via <DEL> or by backing up to the previous line) it gives up and reverts to clearing the bottom portion of the screen. Someday it'll handle this too, but not for now...

* Replace
 - Verified replaces repaint the screen only when necessary

* Kolumn & Adjust
 - Are no longer direction sensitive
 (If direction was '<' then <space> used to adjust/kolumn left and <backsp> went right)

* Zap
 - No longer issues the message "You are about to Zap a lot", etc.
 The rationale is that the zapped text either will fit into the copy buffer or not. If it will, it can easily be recovered by C(opy B(uffer and so the warning is not really necessary. If it won't fit you will get a warning to that effect and be asked for verification anyway so this one was redundant.

* WordMove
 - Now behaves the same forward & backward.
 - No longer stops at illegal chars when moving forward.

* Delete
 - Supports P(age and O(ppPage movement commands

* Environment
 - Main menu has been rearranged
 - DeleteMarkers has a new look

* Nested edit
 - The nested edit file name suffix has been changed from 'ASE!' to 'NEST'.
 ('!' is not actually a legal file name character)

* Quit
 - C(hange Name "sticks" if you R(eturn or <esc> from Q(uit
 - C(hange Name discourages attempts to change the volume name and complains if you persist

* general display
 - Screen updates are smoother, flashing has been lessened

* Initialization
 - Emits about a zillion dots during initialization. Don't panic, this is a concession to make polled type-ahead (particularly on the Apple) more reliable.

* Byte sex issues
 - Text file headers are now flipped transparently. This means that you will no longer lose all of your markers and other environment information when you transfer a text file to a machine of the other gender (for example, a Stride and an Apple)
 - Opposite gender directories are now handled as appropriate when a file menu is requested. This means that under IV.= the directory is now

displayed properly and under other versions an error message is displayed (the os would not be able to open any file in the directory at any rate).

* Size
 - Code file size is 10 to 15% smaller
 - Edit buffer is much larger on combined code & data systems

* Bug fixes
 - Over 100 known bugs have been fixed. Some of these are highlighted below.

---

some notable differences between 0.8a and 0.9r versions of ASE 25 Feb 85
(included for the edification of those running 0.8a and 0.9q)
(note: this list is (probably) not comprehensive)

* P(age
 - No longer displays a partial page at the end of the buffer if there is more text in the file.

* Fn Keys
 - CopyDown now places the key name in the text form.

===

BUGS CONSIDERED FIXED IN ASE 1.0a (& 0.9r):

This is a list of some of the bugs fixed in the 1.0 release of ASE. This is an abbreviated list so if your favorite bug is not here, ask. Chances are quite good that it has been fixed.

[ASE]
Illegal commands should ring the bell. - rb 26 Jun 84
 Ok, ok. It was silenced because of user complaints (& besides it saved a few bytes). So now its back by 'popular'(?) demand in A.0f 30 Jul 84

Sometimes clears & rewrites the same prompt. - RGoldthwaite 01 Oct 83
 fixed in 0.9r 05 Nov 83

The error messages STINK. Stuff like 'Error: Bad file name or no room in directory' is ridiculous. It should give the proper msg for each error, not some multiple choice test. Messages like "can't read page zero" and "cbuf in buf" not only don't but shouldn't mean anything to the end user and should be replaced with something that does tell them something useful. - everyone
 much improved in A.0a

Far too many errors are considered fatal. - everyone
 improved in A.0a

Can we clean up the files from a crash? - rhk 30 May 82 #201
 This is in reference to the fact that most programs if they crash with files open do not leave those file entries in the directory. This is an OS function and the editor MUST make sure the file in edit is locked into the directory to accomodate shared disk environments where any user booting will _remove_ all temporary files from the root volume with ensuing disasterous effects.

Therefore, no, we cannot clean them up as requested. Sorry.

[entry]
Escape from the file name prompt leaves a promptline on the screen.
- RGoldthwaite 15 Sep 83
fixed in 0.9r 07 Nov 83

[A(djust]
UpArrow changes the default direction (within Adjust) to <-, causing subsequent <space>s to adjust in an unexpected direction. - Ro Lutz-Nagey 05 Nov 83
'Fixed' in A.0f because Adjust is no longer direction sensitive

When direction is set to '<', tab still goes '>'. -acd 06 Aug 82
'fixed' in A.0e. Direction ignored in Adjust now.

[C(opy B(uffer]

may leave cursor in wrong place. - dt 25 May 82 #173
Believed FIXED! in A.0f 30 Jul 84 This one was tough to track and tougher to test so it may not be completely clean yet. PLEASE let me know if it resurfaces.

If copy buffer with a repeat count aborts prematurely with a 'buffer almost full' message it does not update the screen properly. - acd 16 Feb 83
believed to be fixed in A.0f

[C(opy F(ile]
indentation: the first line of the text copied in is at the indent level of the line at which copying was invoked, and vice versa. -rb#11
Believed fixed in A.0f

Copy File into middle of a line. Congratulations! You've just corrupted your file with a mid-line <dle> sequence. -acd 04 Sep 84
Fixed in A.0f

Croaks the buffer in 0.9q if marker is not present or file is opposite byte sex.
FATAL! fixed in 0.9r 04 Nov 83

C(opy F(ile <esc> double flashes the screen. - rb 26 Jun 84
Fixed in A.0f 30 Jul 84

If choose too big a file, you get the proper error message, but afterward, the screen not refreshed. -atate 12 Mar 82 #167
fixed in 0.9

[C(opy fnkey]
Why does CopyDown sometimes put null comments into the textform? -Karp
to protect significant blanks which might end up as the first char of a line and therefore be lost to indentation
fixed in 0.9r which lays down a |s~ as it could have all along

C(opy <fkey> generates a spurious <cr> in front of the function key text form.
- rb 6 Oct 82
fixed in 0.9r 08 Nov 83

[D(elete]
Page and OppPage, please. -cdc
Added in A.0f, please report any problems. -acd 22 Sep 84

D/<cr><etx>, then <cr> to get to next buffer load; congrats, you have just won a free blank line! - es 06-Jan-82#139
another off-by-one aggravated by Richard's indecision over which line the <CR> belonged to (the one preceding or following it)
FIXED!!! in A.0a

When deleting chars off the right edge of the screen (yes, thats right) left-arrow back onto the screen will screw up the right hand column or two. -acd 18 Sep 84
Fixed in A.0f and (as far as I can tell) the fix has not screwed-up anything else... -acd 19 Sep 84

D(elete L(ineEnd on a line which is flush right on the screen leaves the last char still on screen until <acc>. -acd 18 Sep 84
Fixed in A.0f -acd 18 Sep 84

[DelChar]
On a line which extends beyond the right edge of screen, G(et a char off the right edge of the screen then D(eleteChar... Bang! - TSiep 16 Oct 83
FATAL!! Fixed in 0.9r, IV.x unitwrite bug... count is treated as unsigned!

[F(ind & R(eplace]
default fkey during parameter entry is fatal. - ??? 10 Oct 83
happens on NCI where <takeup> is <esc>C, the <esc> gets dumped, the C accepted as CaseInsensitive & it loops in a fit of infinite recursion! FATAL!
Fixed (albeit crudely) in 0.9 07 Nov 83

[F(ind]
F(ind // finds the next occurrance of the delimiter (although Set Environ shows the search string properly as null) and a subsequent F(ind S(rch also finds the delimiter. - rb 26 Jun 84
Fixed in A.0f 30 Jul 84 Specifying a null search string is now an error.

When direction is "<", and the <srch_string> is multi-token, and it finds a token_anagram; F(ind S(ame sticks to the first one found. Try F(S(after taking this up: |xb<f/"</|. - rb 19-Dec-81 #119
Actually only happens if the left-most token in the search string is a word-delimiter
Fixed in A.0f 21 Aug 84

In Token mode, only the last occurance of a non-word token will be found in any sequence of that token. Example:
|xf/./|.
should find this one-->....<--but actually finds this one
fixed in A.0f 07 Aug 84

The message 'Pattern not in the file' may be misleading. The pattern is not in the portion of the file that was searched.
fixed in 0.9r, now says 'Pattern not found'

[function keys]

Column1 in a fnkey does not CopyDown correctly. - BPeterson 20 Nov 83
fixed in A.0a

Record misplaces the cursor in Xchange. - RBush 12 Oct 83
fixed in 0.9r 04 Nov 83

When a function key overflows during takeup the function key name is retained although the definition is aborted. - BPeterson 11 Jul 83
fixed in 0.9r 07 Nov 83

[editor initialization]
If File.NEST exists then errors out with file already in edit.
- rb 26 Jun 84
Something in the back of my mind says that there is an obscure situation that makes this behaviour desirable.
Ah, yes! Shared disk volumes, LANs, Liaison, etc. Gotta stay this way. - acd 18 Sep 84

Filenames can spill over onto directory listing on line two. Tacky!
fixed in A.0a

Quick flash of copyright notice is found to be annoying. - Eric Smith
fixed in A.0a

If I start to pick a wrong file, when I get the right one, the $syntax info is lost. -rhk 06-Oct-81 #99
fixed in A.0a

[I(nsert]
Going into I(may flash screen unnecessarily when it thinks it has done a "..." and it hasn't. dt 25 May 82 #174
Believed to be fixed in A.0f -acd 18 Sep 84

when you type a printable in col 80 (rmargin) then a <cr>, the char in col 80 is left blank - jm 31-Dec-81 #127
Believed fixed in A.0f -acd 18 Sep 84

With Filling & AutoIndent true, if the last character of a word falls exactly on the right margin that character gets blanked when the cursor is moved to the next line. - acd 25 Aug 82
Should be better (maybe fixed?) in A.0f -acd 18 Sep 84

With filling on, on a line with a long last 'word', insert a stream of characters near the end of the last word then escape. Characters are lost! - JKhalsa 24 Oct 83
2 characters are lost at the point of the aborted insertion. The line is not stitched back together, which Jai would like, but is _not_ going to be implemented.
FIXED in A.0f & it even stitches the blasted line back together for ya. -acd 18 Sep 84

carriage return wipes too much screen. It's scary to user. -acd
Fixed for InsertLine-capable terminals in A.0f. Alas, 'tis not practical for non-InsertLine tubes. -acd 18 Sep 84

'Buffer almost full' msg doesn't beep and <space> is a common char so error is often cleared before user is aware of it. - KBalke 10 Oct 83
'fixed' in 0.9r, the err msg is really obnoxious about it now 05 Nov 83

[InsertLine]
InsertLine fails on some VT52 emulators - BNR 9 Oct 83
probably overrunning the tube
believed to be fixed in 0.9r. Delay after InsertLine is now controlled by the value of FillCount from miscinfo. 03 Nov 83

[K(olumn]
When moving down and buffer boundary coincides with indentation change, the indentation is altered. - RBush 01 Oct 83
fixed in 0.9r

[M(unch]
Changes "word -12 word" to "word - 12 word". That is, it inserts a <space> after '-' when it perhaps should not. -acd 30 Aug 84
fixed in A.0f

If the command Char is defined as " " (yep, that's a blank), munch will munch one and only one line whether it should or not. -- rb 02-Dec-81 #114
I believe the bug is better stated as, when RunOffCh is " ", munch will munch lines with leading blanks Otherwise, the behaviour seems correct to me
Believed to be fixed in A.0f 30 Aug 84

[nested edits]
filename:= "foon=", tries to jump to " and does <TakeUp> from front of file. - rb 04-Feb-82 #149
fixed in A.0a

In nested edit, selecting 'xyzzy:foon,fubar:dontcare' ALWAYS fails and reports 'Ran out of disk room.' regardless of how much space is available on the destination volume. - rb 6 Oct 82
FIXED in A.0a

[P(age]
if it reaches the end of buffer displays a partial page. - 06-Jan-82 #139
fixed in 0.9r, it cost me 64 bytes of buffer space that I'd rather not have lost but I gather many folks will be much happier now - acd 09 Nov 83

if it causes buffer movement, it seems to do a V(rather than putting the cursor and screen at the proper place - 06-Jan-82 #138
fixed in 0.9r

[Q(uit]
Q(uit C(hange should warn user that volname can't be changed.
it does in A.0a. In addition the new screen formatting discourages the entry of a volume name

C(hange doesn't stick if you R(eturn or <escape> from log entry. -jbondy #29
fixed in A.0a

Q(uit E(xit Y(es leaves the quit menu on the screen. - RGoldthwaite 01 Oct 83
fixed in 0.9r 05 Nov 83

[R(eplace]
If a fnkey contains unprintable chars then R/barf/n where 'n' is the fnkey number will put bad characters into the text. - acd 07 Nov 83
fixed in 0.9r, but I've put it on a compiler conditional and currently disabled the fix. Many folks have screamed, begged and pleaded for a way to corrupt their files with illegal chars. This is baroque enough, and they
-- BUFFER FULL - COPY OFF --
will
have to work hard enough to do it, that I am of the opinion that we should leave it as is and tell people how to use it. - acd 07 Nov 83

"r...." removes a dot. Then "rrs" does too. - rhk 07 Jul 82 #181
yet another off-by-one, obviously...
Fixed in A.0f 31 Jul 84 Null search string is no longer legal & complains

Flashes the screen needlessly during verifies. - acd
Fixed in A.0f 30 Jul 84

Replace string with embedded <ret>s can really foul up a file.
<ret> dis-allowed in A.0a

[S(et M(arker]
S(et M(arker <ret> creates null marker. - rb 02 Sep 83
fixed in 0.9r 05 Nov 83

[unit]
The ASE unit should release the heap on exit. -ADI
fixed in 0.9r

[V(erify]
In a function key still does a Verify, not a prompt for continued execution. Probably should be a DIFFERENT command anyway. - acd 04 Nov 82
fixed in A.0f though it still should be a different command

[W(ordMove]
/WordMove still leaves you at the beginning (rather than end) of the last line. - rb 26 Jun 84
Fixed in A.0f - moves to last token, would eof be preferable?

[X(chg]
I notice that the right screen border is more severe than for Insert. It feels like they should treat both the borders the same, but in fact neither side is treated the same by the two modes. -rhk 27-Sep-81 #72
Believed fixed in A.0f -acd 18 Sep 84

<ret> onto blank line places cursor at left side of screen instead of at indent level of line. - acd 08 Mar 83
fixed in 0.9r 04 Nov 83

A Proposed Library Database

By: Harry Baya

I'd like your help in designing a simple database applicaton that would assist us in using the USUS disk library. If successful it should have a number of other uses.

The system would permit users to use keywords to search for programs available in the USUS library.

The concept is quite similar to the Data Libraries we now use on MUSUS on Compuserve. The main differences are that this system would:
(a) be designed for use on Micro-computers using the p-System
(b) would permit users to add comments to the stored data
(c) would be a public domain program (or freeware)

My assumption is that the large number of programs now in the USUS Library makes it difficult for new users of the library (and some older ones) to find the ones that best meet their needs.

I think of the USUS library as a collection of applications. Each application consists of one or more files. Each application would be one "ITEM" in this data base. Each ITEM would consist of a title, a longer description, and a number of comments by users.

Let's assume that you are looking in the USUS library for a program to do telecomunications on an Apple][. You would put the program disk in one drive and the data disk in a second drive and ask the program to for information about ITEMS associated with the keywords "communications" and "apple". The system would then tell you which programs in the USUS library were associated with both those keywords.

This seems simple enough and I intend to have a version of the system that does this much, together with a data disk containing USUS library titles, available at the USUS meeting in Baltimore this fall. The abilitiy to add comments may take a little longer. Actually the whole thing may take a lot longer, but I thought I would be optimistic.

This article is intended to accomplish the following goals:

- encourage the development of some sort of USUS Library reference system available on disk.

- Engender some specific suggestions as to what the system should do and/or how it should be implemented. Ideally, someone will simply identify a public domain program that could be easily modified to do all that is needed.

- communicate a rather simple concept that interests me and may be of interest to others.

- help me to decide how much effort to put into this and to determine how much help I can get from others.

EASE OF USE AND PORTABILITY

These two goals will dominate the design. A number of other goals, such as faster execution speed and being able to handle as many ITEMS as possible, can be compromised to make the system easier to use and more portable. I am also very interested in minimizing the number of man-hours of my time needed to develop this system. I will use the K.I.S.S. approach whenever possilble.

One of the design goals would be to permit adding or deleting new ITEMS, and keywords associated with ITEMS, as easily as possible. I would like the system to be such that it could used in applications where non-technical first time users would find it of value.

I would like to make the source code publicly available and I would like to have debugged versions that will run on:

- the Apple][,
- the Apple /// (since I have one)
- the IBM PC using both Softech Microsystem's and NCI's version of the p-System.

We will make every effort to make the source code easily portable to most other p-Systems implementations.

HOW WOULD IT BE USED?

The program disk and the data disk would be added to the USUS library. The data disk would be updated as needed to reflect additions to the USUS libraray and made available to USUS members. The data disk might also be updated to comments from users of the library.

ADDING COMMENTS TO THE STORED DATA

The system would permit users to enter comments associated with any ITEM. Users could also enter comments associated with other comments.

Users of the library acquire knowledge about the programs they use. This system would collect that knowledge and make it accessible, just as the library does with programs.

I assume that those library programs that prove useful will collect useful comments. At present there is no way to determine which programs have indeed been sucessfully used by USUS members after they were put in the library. Identifying good, but seldom used, programs will continue to be difficult, though the keyword search will make things a little easier.

To see how this might work, assume that you discover that a particular telecommunication program (in the USUS disk library) is designed for an IBM PC, but will run nicely on an Apple][after only minor modifications. You would then add the keyword "apple" to those associated with that program. You would also enter a comment explaining what changes were needed.

In order to make this ability to add keywords and comments valuable to other USUS members we would need some way of collecting them. Someone would have to be responsible for deciding what would be added to the USUS data disk. Over time some comments would either be dropped out or moved to a lower priority data disk.

My hope is that this system, or something like it, will also prove useful to clubs with their own base computer which they make available to users, as might be the case in a local user's group. A public domain telephone BBS implementation would also be attractive.

I would appreciate your letting me know of public domain packages similar to this, either as a stand alone database system or as a BBS.

ONE CUT AT THE DATA STRUCTURE

My first pass at the design is given below. My best guess is that this is a relatively primitive and inefficient approach. It's main appeal to me is that I think it would be easier to program than the other approaches I have come up with. My hope is that readers will suggest better approaches.

Each ITEM in the database would have:

- a title of up to 40 characters
- a description consisting of up to about two type-written pages (normally less than 6 lines)
- up to 100 associated keywords of up to 16 characters each (usually less than 10 keywords per ITEM)
- Up to 25 comments (normally less than 5) with the same size limitations as the description.

Each comment would be treated like an ITEM. (i.e it would have a title, text/description, keywords etc.)

Data Structure:

The data would be stored in three files of fixed length records:

- keywords
- 40 character records
- association records

Each ITEM in the database would consist, at a minimum, of a 40 character title and one association record.

Each association record would associate an ITEM with one of the three kinds of records:

- a keyword record
- the first record in a linked chain of decription records
- the first record in a linked chain of comment records

To see how this might work consider that in order to associate a new keyword with an ITEM,

(a) the keyword would be added to the keyword file

and

(b) an association record would be added to the file of association records.

SEARCHING:

Any search using keywords would search through the association records to find those ITEMS which met the selection criteria.

Each record in each of these three files (keywords, 40-character records, association records) would be identified by the record # of the record it occupies in the file. Eg. if the keyword "game" were stored in record # 18 of the keywords file, then we would say that it's "file address" was 18. An association record indicating that a particular ITEM was associated with "game" would contain the number "18" rather than the word "game".

Lets look at the way this "file address" would be used in a typical search of the database. Assume that user wants a list of the programs associated with the keywords "communications" and "apple".

(1) The system would first search through the keywords to find the record addresses of these two keywords.

(2) the system would then search through the association records to identify those ITEMS associated with both keywords.. It would do this by identifying those ITEM record addresses associated with both of the keyword record addresses.

The following should make this search process a little clearer.

Consider that the system could collect together all those records that associate

(a) the record address of the keyword "apple"

with

(b) the record address of an ITEM title.

The system could then collect a second group using the keyword "communications".

It would then find whether there were any ITEM record addresses that were in both groups. If an ITEM title record address occured in both groups the system would read the record from that address and display the title to the user.

I think it would simplify the system if the 40 character title were stored as the first record in the linked chain of records that make up the longer description. An ITEM with a title and no description would be a

chain that is only one link long.

Comments would be handled similarly.

This is the conception of this application. I wonder whether it will survive gestation and if so, how long it will take. Perhaps we should plan a birthday party and leave the date open.

Please send comments or suggestions to :

Harry Baya
565 Broadway, 2H
Hastings-on-Hudson, NY, 10706
Compuserve # 72135,1667

USUS Software Library

From Jon Bondy

Note our two new Macintosh distributors, and that Marc Wigan's address has changed.

Prices have been standardized for all of those formats which are distributed by more than one person. IBM, NCI, and Sage prices have been set at $6 (1 volume/disk), $7 (2 volumes/disk), or $8 (3 volumes/disk 10-sector) (all 80 track). 8" disk prices have been set at $5 each. Apple][prices are set at $8.00 for a two disk volume.

If you have problems with a distributor, first contact them directly to see if the problem is a misunderstanding or some "reasonable" delay (death in the family, computer hit by lightning, etc). It is reasonable to expect a distributor to send you disks within 4 weeks of an order: delays longer than that should cause you to investigate further. If you don't get satisfaction, please WRITE to both them and myself, describing the circumstances.

- Jon Bondy

USUS Software Library Distributors

Distributor	Disk Format(s)/Prices
Mark Bailey 1942 Linden Ave, #D Highland Park, IL 60035 312-433-7697 (h) 312-937-4206 (w)	Macintosh at $7.00/volume. IBM PC and Apple][at standard prices.
Henry Baumgarten 3325 Hillside Street Lincoln, NE 68506 402-489-6441 (h) 402-472-3301 (w)	Standard Sage and 8" prices; see above

Jon Bondy Box 148 Ardmore, PA 19003 215-642-1057 (h)	Standard Sage and 8" prices; see above Subtract $4/disk if you send him properly formatted and zeroed Sage disks. Subtract $3/disk if you send him formatted SSSD 8" disks
Clark Gestring 4643 W Oberlin Place Denver, CO, 80236 303-797-6739 (h) 303-694-8797 (w)	TI 99/4A SS SD 35 TK @ $13/4 disk volume. SS SD 40 TK @ $11/3 disk volume. DS SD 35 TK @ $10/2 disk volume. DS SD 40 TK @ $10/2 disk volume.
Kenneth K. Kam P.O. Box 3112 Torrance, Ca. 90510	Heath H-89 5-1/4 inch disks for $17/tri-disk vol.

Distributor	Disk Format(s)/Prices
Dick Karpinski 6521 Raymond Oakland, CA, 94609 415-666-4529	Northstar disks for $15/dual single-sided disk volume.
Jim Harvison 4144 Red Bandana Way Ellicott City, MD 21043 301-992-8831	Standard Sage and Apple prices; see above
David McFarling 3815 Adams Street Lincoln, NE 68504 Rockville Md, 20850 402-467-3591	Standard IBM and NCI disk prices (specify format and number of blocks); Apple // disks at standard prices; SofTech Universal Medium disks at $8/vol. (two disks); Macintosh at $7.00/volume
George Schreyer Box 1645 Redondo Beach, CA 90278 213-371-0198	8-inch disks for $10/volume.
Marc Wigan Wigan Associates Box 233 Heidelberg Victoria 3084 AUSTRALIA 03-459-9671	8-inch disks and Sage disks. Pricing unknown.

Who Supplies What:

"Std" 8"	Henry Baumgarten, Jon Bondy, George Schreyer, Marc Wigan
Apple II	Mark Bailey, Jim Harvison, David McFarling
Heath-89	Ken Kam
TI	Clark Gestring
IBM	Mark Bailey, David McFarling
NCI	David McFarling

Sage	Henry Baumgarten, Jim Harvison, Jon Bondy, Marc Wigan
NorthStar DD	Dick Karpinski
SMS Univ Med	David McFarling
Macintosh	Mark Bailey, David McFarling
Victor 9000	Nil
CSI 6809	Nil
OSI	Nil

USUS Software Library Order Form

Use this form to order the USUS Library volumes of your choice. Send the order DIRECTLY to the distributor, and make your check out to him/her, NOT to USUS.

Thirty-four volumes are now available. All 8080-specific code and CRT terminal data on Volumes 1, 2A, and 2B have been removed to form a single Volume 1 for the Apple. Volume 2B is normally shipped in CP/M format and has little utility for anyone without CP/M. Volume 21 is Apple-specific. A special Western Digital (WD) format disk contains a Mapper program, allowing owners of WD machines to read DEC-format disks, the USUS standard 8-inch format. WD owners CANNOT obtain double-density disks from USUS; ask WD for help converting single to double density on your machine. DEC format begins on track 1/sector 1, with interleave 2 and skew 6.

NAME, TITLE _____

COMPANY (if work address) _____

NUMBER AND STREET _____

CITY, STATE, ZIP, COUNTRY _____

SYSTEM (PLEASE fill in) _____

DISKETTE FORMAT REQUIRED (circle one):

Standard 8-inch SSSD NorthStar Apple OSI

Heath H-89 IBM PC IBM PC TI SAGE

VOLUMES REQUESTED (circle one or more):

1 2A 2B 4 5 6 7 8 9 10 11 12 13 14 15 16 17

18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 UK3 UK4

Price per volume (see distributor list) $_____

Total for disks $_____

Sales tax (if intra-state sale) $_____

TOTAL AMOUNT ENCLOSED ($U.S.)$_____

(NOTE: ALL orders MUST be prepaid!)

IMPORTANT: PLEASE READ AND SIGN THE AGREEMENT ON THE FOLLOWING PAGE, OR YOUR ORDER CANNOT BE PROCESSED!

Software Order Agreement

I will not permit the Programs I receive pursuant to the foregoing order to be published for profit in whole or in part or to be transferred to any person who is not a member of USUS, without the express written consent of the author identified in the files of the Library. Further, lacking such author's consent, I will ensure that the following items have been submitted to the designated individual (for now the USUS Treasurer, USUS Library Chairman, or a Library Distribution Subcommittee Chairman) before transferring any Program(s) to another USUS member: a) the current Software Order Agreement signed by the software recipient; and b) $1 per volume or fraction of a volume received.

I acknowledge that neither USUS nor any of its representatives nor the author makes any warranty with respect to the Program, particularly NO WARRANTY OF FITNESS FOR ANY PURPOSE, and that the Program may require extensive alteration by an expert in programming before it will suit my needs.

(Signed) _____ Date _____

USUS ™
P.O.Box 1148
La Jolla, CA 92038

Bulk Rate
U.S. Postage
PAID
Brooklyn, NY
Permit No. 316