

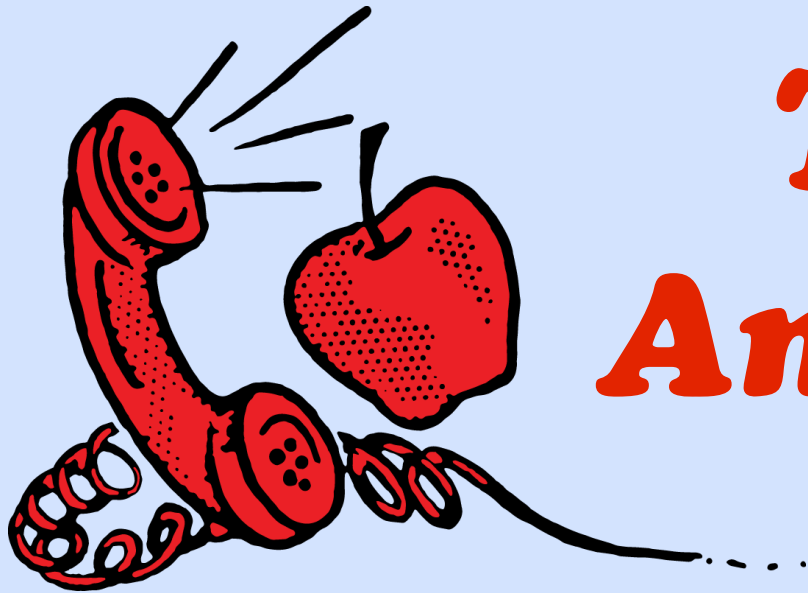
Call-A.P.P.L.E.TM

World's Largest Apple User Group Magazine – *Since 1978*

Volume 28 Number 1

February 2018

www.callapple.org



The 40th Anniversary Issue

- 40 Years of A.P.P.L.E. – From the Basement to the Tower
- The Founding of APDA – The Apple Programmers and Developers Association: Don Williams on How A.P.P.L.E. Created an Institution
- 40 Years On: Bob Clardy's Memories of A.P.P.L.E. Founder Val Golding
- The CRPG Book Project: Role-Playing Game History 1975 - 2015
- Cross Chase: A Multi-platform, Multi-system Game
- Roger Wagner to Keynote KansasFest 2018
- Blankenship BASIC: The Return of the Programmer
- 6502 Assembler Tricks: Self-modifying code based on the 3D-Demo
- 12 Years On: PLASMA 1.0 Released
- VCF Pacific Northwest Recap and Photos
- Freshly Squeezed Reviews: HomePod Gives Me Accessibility



Apple PugetSound Program Library Exchange

Call-A.P.P.L.E.TM

World's Largest Apple User Group Magazine – Since 1978

Volume 28 Number 1

February 2018

www.callapple.org

A.P.P.L.E. Board of Directors

Chairman – Bill Martens

Director – Brian Wiser

Director – Jim Maricondo

Production & Design

Bill Martens

Brian Wiser

Cover Photo

A.P.P.L.E.

Call-A.P.P.L.E. Magazine ISSN

8755-4909 1705-4109



A.P.P.L.E. Staff

Editor-in-Chief – Bill Martens

Managing Editor – Brian Wiser

Staff Writer – Javier Rivera

Staff Writer – Marcus Adams

Disk Digitization – Antoine Vignau

Contributing Authors

Val J. Golding Rick Sutcliffe

Frank Petrie Fabrizio Caruso

Bob Clardy Don Williams

Evan Koblentz Kevin Savetz

Marc Golombeck

Subscriptions

Subscriptions to *Call-A.P.P.L.E.* magazine can be attained by joining the Apple Pugetsound Program Library Exchange (A.P.P.L.E.) user group, founded in 1978. The magazine is one of the premium benefits. For more information, please visit the membership page at: www.callapple.org/members

Membership in A.P.P.L.E.

Membership fees are \$27.95 per year, and include a one year subscription to A.P.P.L.E., user group discounts, and archival materials including *Call-A.P.P.L.E.* magazine, *Mac-A.P.P.L.E.* magazine, legacy software, and other A.P.P.L.E.-produced websites. Annual membership dues are always due the month you first pay.

Advertising

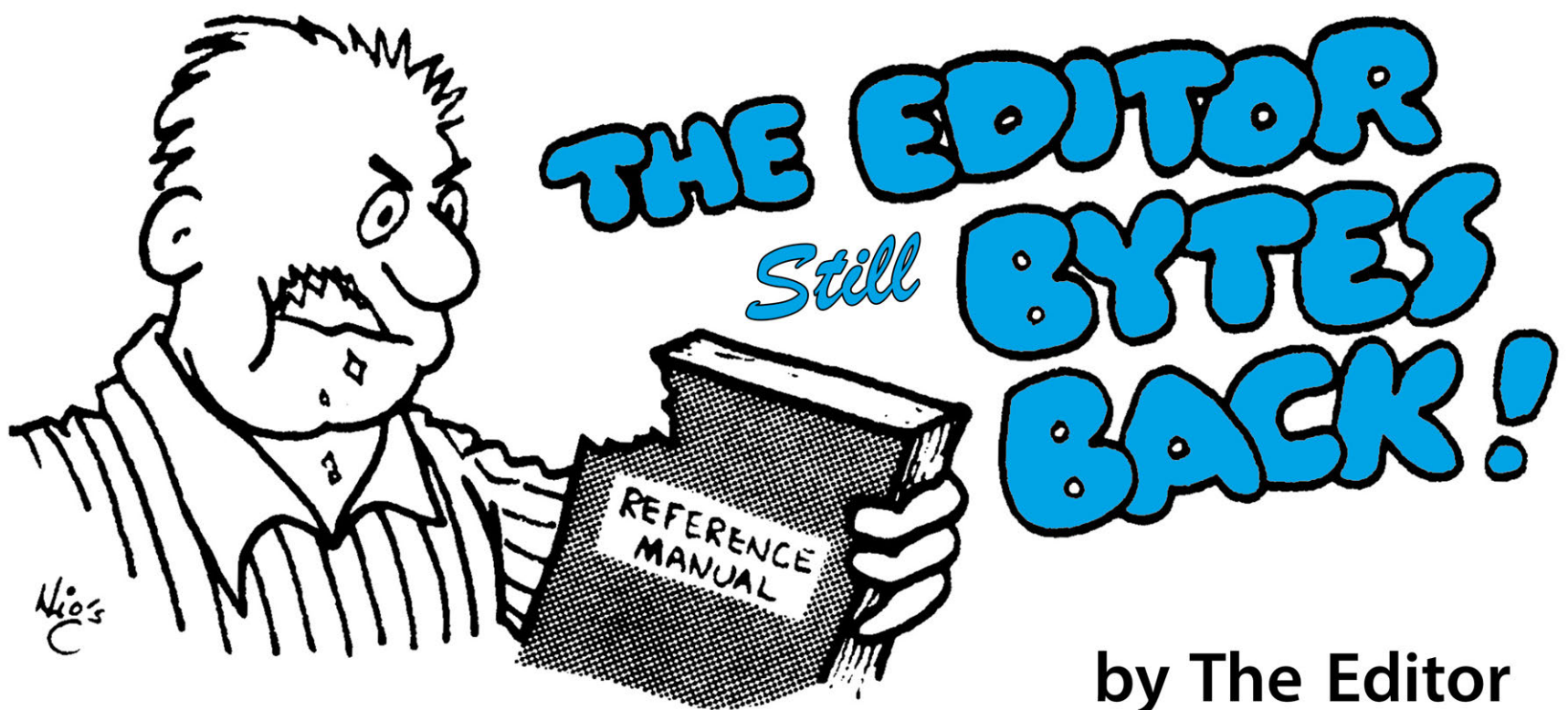
If you wish to advertise in *Call-A.P.P.L.E.* magazine, email sales@callapple.org for our very competitive rates. We offer in-magazine, on-site, and full sponsorship for prospective advertisers.

Submissions

Call-A.P.P.L.E. is always looking for new and interesting ideas and articles related to Apple, Linux/Unix, technology of general interest, technical or editorial, modern or retro. All submissions must be original works of the person submitting and free of distribution restrictions. Please email your proposal to: editor@callapple.org. By submitting materials to us, you agree to give A.P.P.L.E. the royalty-free right to publish and reuse your submission with your name in any form in any media. We reserve the right to edit, change, or not use anything submitted to us.

Copyright

Permission is required before reproducing any material from *Call-A.P.P.L.E.*. Please email editor@callapple.org. *Call-A.P.P.L.E.* magazine is an independent publication of Apple Pugetsound Program Library Exchange (A.P.P.L.E.), not affiliated with Apple Inc. Contents Copyright © 2018 Apple Pugetsound Program Library Exchange. All rights reserved. Apple, the Apple logo, and all Apple hardware and software brand names are trademarks of Apple Inc., registered in the United States and other countries. All other brand names and trademarks are the property of their respective owners.



40 Years and Counting!

2018! Wow, so soon. And yes, 40 years! Another grand anniversary comes and we once again are forced to look into the rearview mirror at the road we have traveled thus far.

From our humble beginnings in Val Golding's basement to a prime address in Renton and Kent Washington, to the closet of Norman Dodge's home in Seattle, to the Tower in the Sky in Tokyo and once again back to the land which gave us our birth, it has been a road which had a few pot holes and speed bumps along the way.

We have persevered and once again are publishing the magazine which so many of us have grown up with and grown to love over the years. *Call-A.P.P.L.E.* magazine began with Volume 1 Number 1 which in reality is just Val's call to arms for the user group which numbered more than 50,000 members at its peak. Now, as we continue to serve the community, we have an absolutely knowledgeable group of people who are fervent Apple users and A.P.P.L.E. members.

As for me personally, I find this issue somewhat melancholy as this will mark the 16th year which I personally have been at the helm of things. It is a duty I thoroughly enjoy and actually enjoy getting feedback from those of you who read my dribble. As for the magazine, Brian Wiser and I would like to expand it to a full year of issues, but with the way online publishing is these days, we have made a decision to publish no more than four times a year in order to best facilitate the magazine's position in the market.

We also publish our *A.P.P.L.E. Member News*, a periodic newsletter email to our members, which details the goings on here at A.P.P.L.E.. Our staff continually posts news on the website, and sometimes we expand those particular news articles in the magazine, making them even more informative and worth reading here.

As a member you know that we continue to have a plethora of *unique* reviews, detailed technical articles, and personal histories shared here in the magazine – to make the magazine an even greater value and resource for our members. To encourage membership and drive awareness of the magazine, we are planning to include snippets of each magazine article on our Web site.

We finally migrated to a new server company and are no longer encumbered by old server software. This means that we can offer more features on the Web site, along with our award-winning news sections, there will be an additional push in the technical realm, giving out readers more of what *Call-A.P.P.L.E.* has been known for over the past 40 years. Our thanks to the authors who have made this technical push possible.

Also this month, we have once again made available later *Call-A.P.P.L.E.* issues for the years 2002 and 2003. We are working on getting the issues through 2010 up and hope to have them up sometime in April or May prior to the next edition of our 2018 magazine. While this may seem like an easy task for issues published in the 2000's, we are actually going back and correcting the issues which were published

by former president, Mike Pfaiffer who was in charge of publishing while Bill Martens was serving his country.

The changes include placing the proper header on the issue, including the required ISSN numbers, as well as the publishing information and table of contents pages. Once we have completed the updates, we will upload them with the 2004 issues.

This week, we are also starting to reinstall the forums system. This is due in large part because we are recovering a lot of the data from the old systems which will hopefully give us everything that we had from 1987-2001. The data recovery will be happening in Seattle next month. It is our hope that the trove of information will be a boon for our readers and members alike, both in depth and knowledge long thought to have been completely lost.

And now that this 40th anniversary is really upon us, we take a few minutes to remember the man who many called simply "The Founder." Val Golding was a mentor and friend to many of us and although he has been gone from this world now nearly 10 years, his legacy lives on. The meticulous detail that he paid to each item and the time he took to ensure that things were right, were always traits that he liked to pass on.

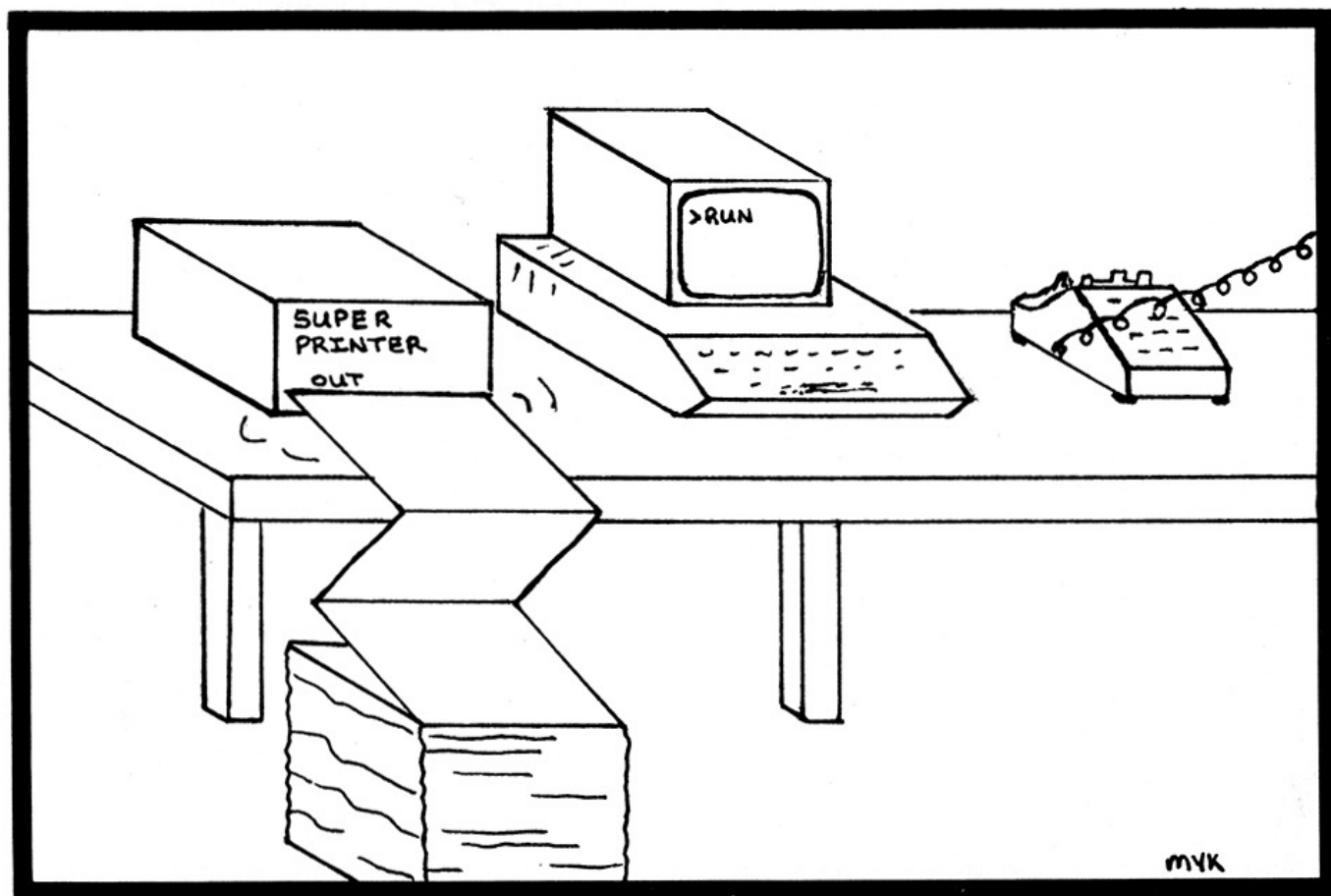
Even Kathryn Hallgrimson, former A.P.P.L.E. editor, called him the "Golden Man" since he always had some

piece of knowledge to impart to those around him. I too benefited from that benevolence when I first restarted the magazine in 2002, and continually find little things that have held over all the years which were items he implemented.

But this is a celebration and as part of that celebration we will be releasing a number of new products that are sure to delight our members. This will start with a challenge program we are putting out to the programming community and will culminate with our completion of our original catalog in addition to releasing Enhanced Editions of several of our books and software.

We expect to finish the *All About Applesoft: Enhanced Edition* in May along with another special book. Also in the coming months will be eBooks for *Cyber Jack* and *Nibble Viewpoints*, which is no small task for Brian to undertake, requiring a lot of additional formatting and preparation.

Stay tuned as the A.P.P.L.E. staff, now recovered from our server migration, turns its sights toward production once again.



... HELLO, VAL? I THINK I'M IN A LOOP.

A.P.P.L.E. Presents: *Call-A.P.P.L.E. In Depth*

Call-A.P.P.L.E. In Depth

Number One \$7.50 (P100 Canada)

All About Applesoft

A Call -A.P.P.L.E. Extra

Apple PugetSound Program Library Exchange

Call-A.P.P.L.E. In Depth

Number Three \$9.00 (P100 Canada, \$12.00 U.K.)

All About DOS

A Call -A.P.P.L.E. Extra

Apple PugetSound Program Library Exchange

Call-A.P.P.L.E. In Depth

Number Two \$2.50 (P100 Canada, \$7.50 U.K.)

All About Pascal

A Call -A.P.P.L.E. Extra

Apple PugetSound Program Library Exchange

Call-A.P.P.L.E. In Depth

Number Four \$15.00

all about Applewriter Ile

by Don Lancaster

PDFs Available to A.P.P.L.E. Members

Join Today:

www.callapple.org/members



<http://beagle.applearchives.com>

The Brutal Deluxe Software
Apple II Cassette Collection
New Cassettes Added!!!



www.brutaldeluxe.fr/projects/cassettes

The Australian Apple Review

The Australian Apple Review

Registered by Australia Post Publication No N860313
* Recommended retail price

A Gareth Powell Magazine Vol 1 No 3 March \$3*

The new Apple Macintosh – the future Apple as a Computer Terminal Logo – the cry of the Turtle Add-ons with Bells and Whistles Numeric keypad Disk drive analyser Buglettes Program a Winner at the Track Worm in the Apple News Letters The new Apple Macintosh – the future Apple as a Computer Terminal Logo – the cry of the Turtle Add-ons with Bells and Whistles Numeric keypad Disk drive analyser Buglett

AAR.AppleArchives.com



Virtual Apple II



Virtual Atari



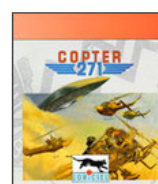
Virtual Game Boy



Virtual SMS



Apple II Atarisoft



Virtual Amstrad CPC
GX4000

GAMEZYTE.COM

Vintage Gaming
at its best!

40 Years: From the Basement to the Tower

APPLE PUGET SOUND PROGRAM LIBRARY EXCHANGE

C/O Val J. Golding
6708 39th Avenue SW
Seattle, Wa. 98136
(206) 937-6588 (Home)
(206) 623-7966 (Office)

February 9, 1978

Dear Apple Owner:

The purpose of this letter is to form an Apple Computer users group, as indicated by the tentative name above, and to further the exchange of information and programs of interest to Apple owners and users. A preliminary meeting has been scheduled for 7 PM Tuesday, February 21st at Computerland, 1500 S. 336th St., Federal Way, Wa. 98003 (Phones 927-8585 and 838 9363).

Regretfully, I do not have the time available to continue this project beyond the formative stage; therefore the first order of business will be (hopefully) to find someone that can. I can see this group as a very useful, self-help type of organization, and am most anxious to see it progress. I will, of course, be available for assistance.

The APPLE goals, as conceived, should include establishing a software library for exchanging programs on a cost-only basis, programming and technical assistance, the exchange of information on Apple-compatible equipment and peripherals and the publication of a brief newsletter to serve as a medium for some of the above items. You are urged, therefore, to attend this meeting and help in the formation of your club. If for any reason you can not attend, but are interested in the group, please contact me for further details as they develop.

PROGRAM EXCHANGE

Bring your recorder and tape with you. We have the following at no cost

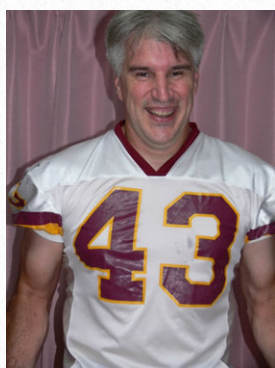
- ZOT = A snappy one-way conversation with your computer.
- STAR WARS = Galactic target practice with your paddles.
- STOP WATCH = A real time clock and stop watch for your Apple.
- HEX-DEC = A program that converts hexadecimal numbers to decimal and back.
- ANNE APPLE = An interactive rap session with your computer.

EQUIPMENT and MATERIAL

DAKHEC-60 cassettes @.73 ea. plus shipping. See me to order by Mar. 10.
I/O board for Apple by Electronics Warehouse, Redondo Beach \$74 assembled.
IR125 dot matrix printer by Integral Data @ 799. They are working with Apple to interface it. See their ad in February Byte.
"HOW TO PROGRAM MICROCOMPUTERS" by Wm. Barden, Jr., SAMS book No. 21459, a MUST at 8.95 from Retail Computer Store and Computerland.
A new RF Modulator from VHF Industries has been ordered by Omega Stereo. Reported to offer much better resolution. Priced at 79.95.
Empire Electronics now carrying Apple line.

HOPE TO SEE YOU — *Val*

by Bill Martens



Over the past 40 years, there have been a multitude of changes in the computing industry. But, here we are, after all those 40 years and Apple Inc. still rules the computing world and A.P.P.L.E. is still producing books, software, and of course our old friend, *Call-A.P.P.L.E.* magazine. While, we are not the powerhouse we once were, we still lead the news world for retro Apple computing and have over 60 Web sites dedicated to the pursuit of the hobby and to educate and enlighten.

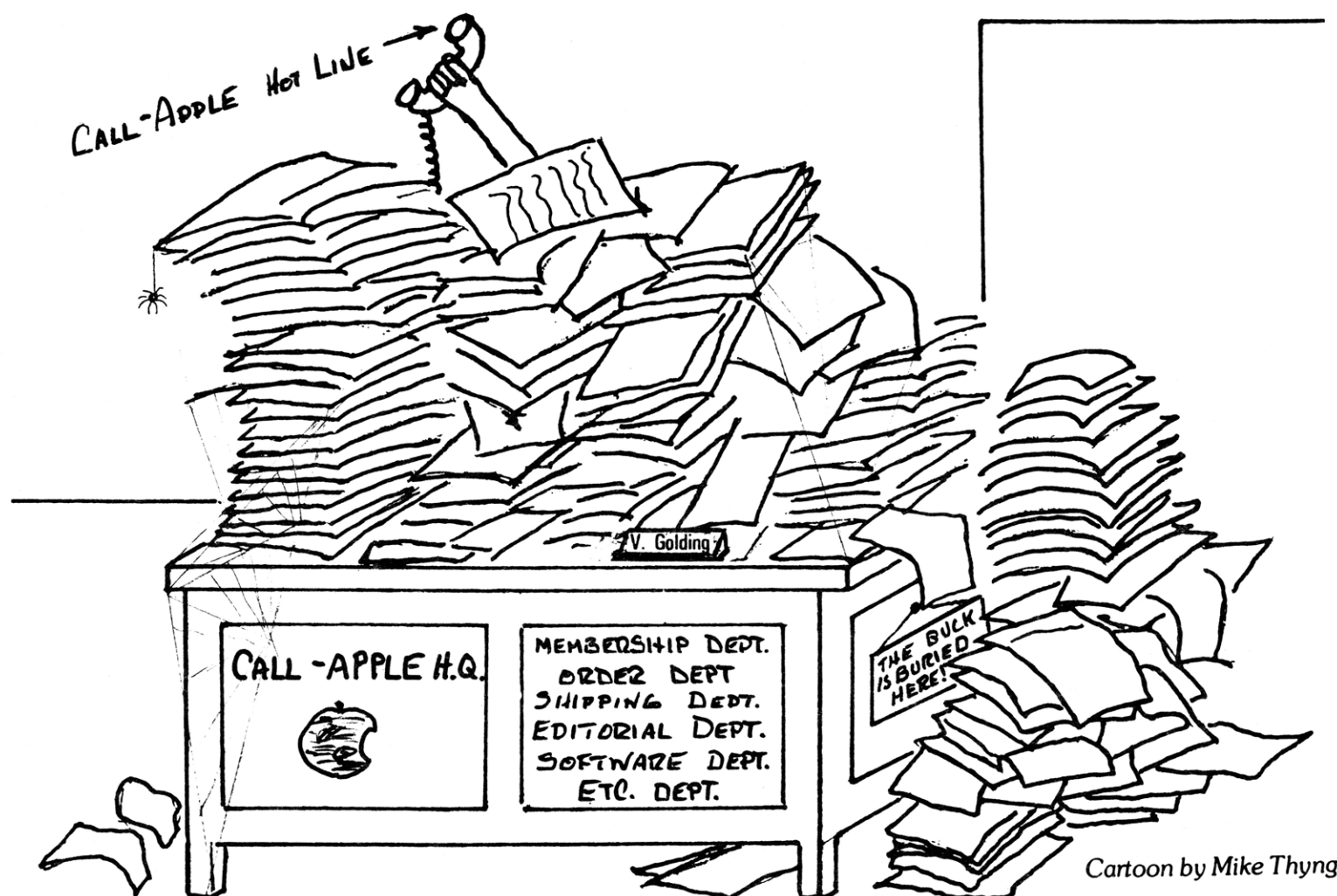
Of course, none of this would have been possible were it not for the founder Val J. Golding who, in his pursuit of knowledge, set out to find other like-minded souls all those years ago.

The newsletter which called for a joining of the minds on February 21, 1978, also became *Call-A.P.P.L.E.* magazine volume 1 number 1. However, little did we know that this

was to become one of the largest computer user groups in the world in a matter of years, just simply based on the idea that users could help other users learn about their computers and share software and solutions with others.

Of course, even Val Golding really had no idea that the dream would become the explosion that it became. Max Cook, who was solely responsible for encouraging Val to reach out, also became the host of the fledgling users group with the very first meeting of A.P.P.L.E. held February 21, 1978 at the ComputerLand in Federal Way, Washington – a computer store run by Cook.

Those hearty few who did appear numbered but 13-15 people, but those few would soon number over 50 within two months. As the members piled up, so did the requirements for more robust plans as well as staffing. Yet Val continued to push everything from his basement in West Seattle, all the while attempting to get everything perfecting the fledgling magazine, do user support, and all the other things that go with being the driving force behind a computer user group.



Cartoon by Mike Thyng

"It's the Boston user's group on the phone, sir . . . They want to know if they can be of assistance . . ."

That basement and its owner, along with the trusty Apple II computer on the shelf became grand central station. And with all that effort also came a deluge of responsibilities – no matter how superhuman Val tried to be, he could not be efficiently attended to by a single person. But trudge on he did. Val did most everything and in the process managed to get completely snowed under by the magnitude of what was actually going on.

Val was responsible for producing the software, the newsletter, and of course running his increasingly popular “A.P.P.L.E. Help Line,” which began ringing at all hours of the day as users began to discover it.

Dick Hubert and others rescued Val in September as they began to go through orders, discovering that they needed to set up some sort of fulfillment system which would facilitate production, packing, and shipping in short order. This all began and thus the real world of A.P.P.L.E. was born.

While there have been many words written over the years about what A.P.P.L.E. meant and what fun it had been when Apple was young, there was always an air about things

that it was still a business. And in business, sometimes things are not as the people involved would like.

Now that the Apple II is not much more than a hobby for most folks, the hobby does still take on an air of business for those of us who are involved from a vendor standpoint.

Val Golding, while he was enthralled about the possibilities of helping other users and of building A.P.P.L.E., he really didn't like the business aspect of things. He was always first and foremost a computer user, an editor, a teacher, and most of all, to those of us under his tutelage, a mentor.

I was privileged enough to have been able to re-build A.P.P.L.E. starting in 1999 with his guidance and the guidance of the others who were there at the beginning of A.P.P.L.E. But now, I must and will give Val the floor for his take on things.



Eaten by a GRUE

A podcast about **Infocom** games, text adventures, and interactive fiction.

Listen in your podcast app or at monsterfeet.com/grue/

My Take on A.P.P.L.E. History

In the words of our founder – Val J. Golding – as written by him in 2004.



by Val J. Golding

About A.P.P.L.E.

I started with the acronym and just played around with it until I found something that fit. I was never happy with combining "Pugetsound" but I never figured a better word. The "Program Library Exchange" of course, was a natural, because that was what we started out doing. A few of us, including Mike Thyng and Bob Huelsdonk bought our computers from Max Cook, a manager at the ComputerLand where I bought my first Apple II, and naturally I was calling him with loads of questions at first, most of which he couldn't really answer.

Bob was pretty savvy, having been in the business (with Honeywell) for quite some time and was able to point me in the right direction for the answers. Max, although a

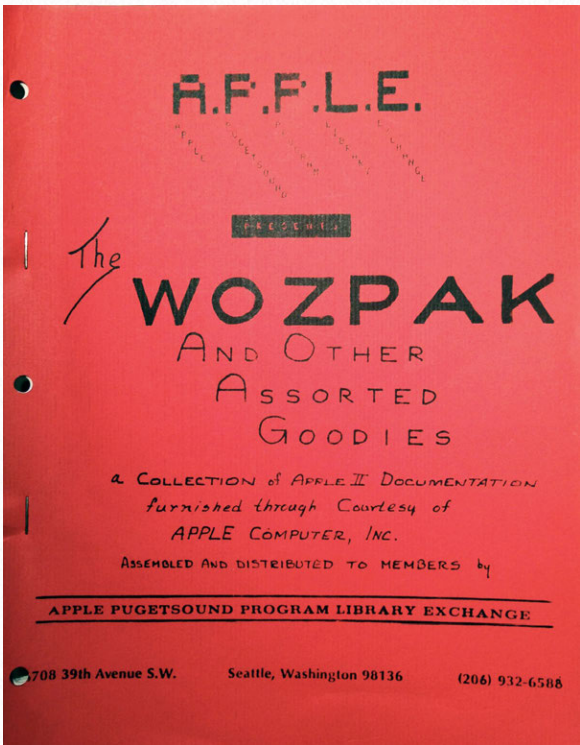
nice guy, was "just" a salesman. Eventually he began calling me (as I started developing some Apple technical expertise) with questions and that was how the A.P.P.L.E. hot line got started.



After the club was formed I gave out my phone number as the hot line number and many times I'd answer it in the middle of the night (like from the east coast, etc.). We separated about 14 or 15 years ago on the best of terms. Usually we have her over for dinner and a movie on Sunday.

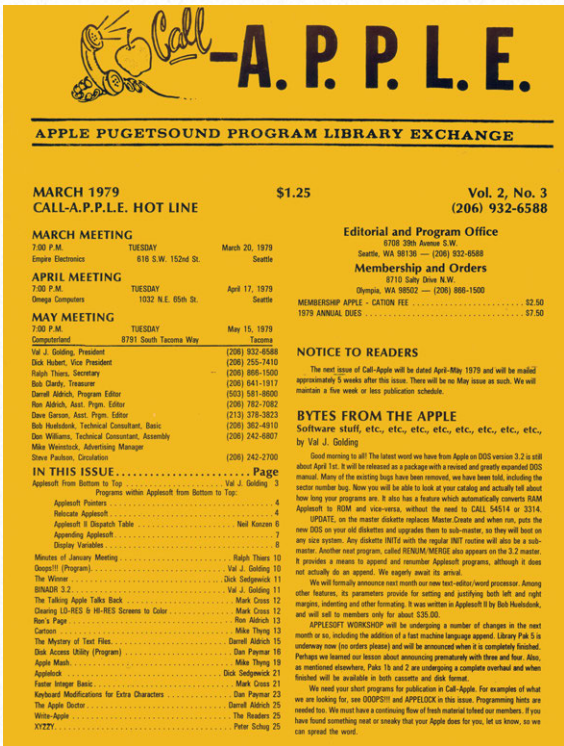
The calls I got most often were from somebody who had inadvertently or otherwise, deleted their program

(Integer or Applesoft, either one) and I'd talk them through manually resetting their program pointers to recover the program. Todd Rundgren, who is still an active rock star, used to call the hot line frequently.

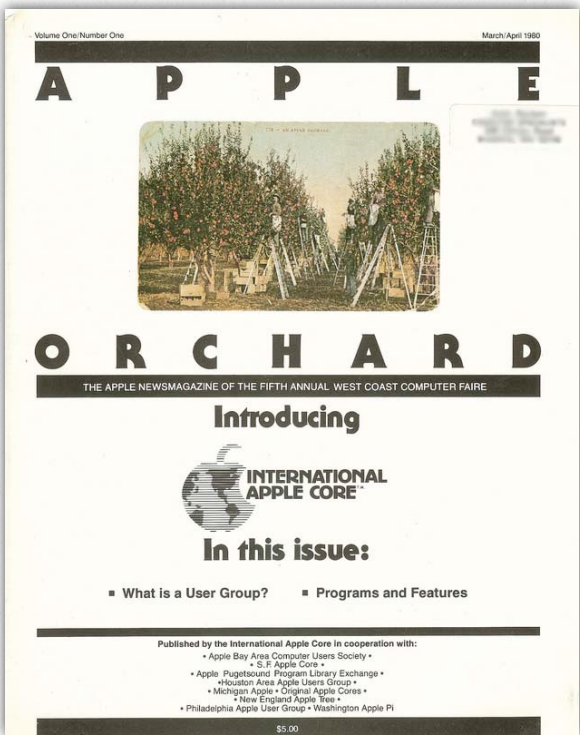


The Magazine

Call-A.P.P.L.E. Vol I, No. 1 was designated as such only after the fact, i.e., when the second issue came out. No. 1 in fact was a form letter mailed to customers of a Federal Way ComputerLand store as an invitation to form an Apple Computer user group.



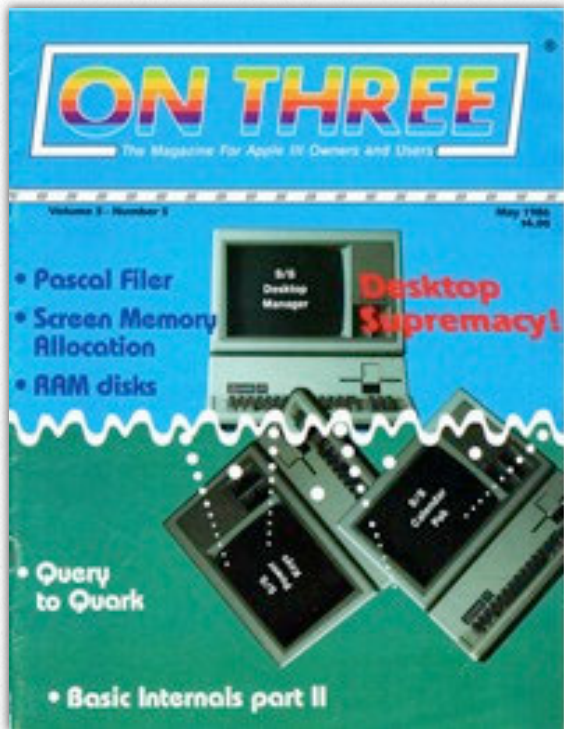
It seems to me we were just a handful of people and I offered the name and we all informally said "That's fine with me" or words to the effect. I'm thinking Call-A.P.P.L.E. Vol. 1 No. 1 may have had that at the top of the page already.



I left A.P.P.L.E. in 1984. I was upset with management which by then had become a corporation with silly rules and useless meetings, etc. I had the opportunity to found a new magazine for kids called *The Apple's Apprentice*, and so I moved to San Diego to do that but it was never a great success. Then I became editor of an Apple /// magazine out of Ventura, so I moved to the Valley and commuted between there and Ventura, working mostly at home.



The Apple /// was a fantastic machine and preceded the PC with many functions such as drag and drop between applications, etc. I researched Apple /// BASIC in much the same way I had Applesoft, which was actually its underlying engine. The /// Basic was mostly written by Randy Wigginton who imparted many of its secrets to me.



Their lead programmer left them to work for Apple and the magazine teetered on the brink of extinction. I got a phone call with an offer of a job with *Softdisk*, an Apple magazine on disk and so packed my bags again and off to Shreveport, LA for a year.



After I managed to get into several disputes with the owners (mostly my own fault) and back to Seattle. Kathryn Hallgrimson, who had been assistant editor of *Call-A.P.P.L.E.* became editor when I left. She and I were very close and when it was determined that *Call-A.P.P.L.E.* was going to go down the tubes, she called me and asked if I would write for the last issue in 1990, which I did.

Postscript by The Editor

I had the privilege of working with Val, seeing him when he was much older and wiser, and meeting his wonderful family. In the years I was able to communicate on a regular basis, it always amazed me that he had seemingly every little tidbit of information stored that one could use from icons to fonts to scraps of information that would put one on a path to more clarity on a topic.

The Founder of A.P.P.L.E. at Work

As it is always best to the the history of a company from the person who experienced it, I made the decision to get it from Val and the others who were there during the time in which it existed. Many of the materials within this profile came directly from Val and his friends.

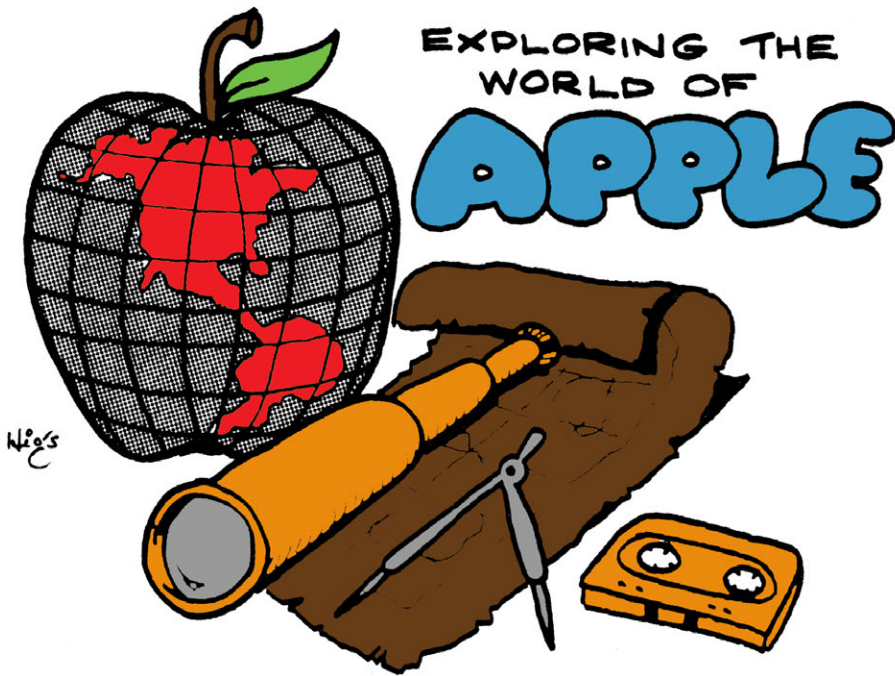
He was a prolific writer, whether it was writing about his beloved street cars in San Francisco and his motorman years, or writing for his children's school newspaper, or writing his article for A.P.P.L.E. each month. He had an affinity for writing and editing but that was just one side of his life.

If you gave him the time, he would tell you all about his Jazz and the band he was with and even the albums he produced. The French horn was his instrument of choice and he was particularly talented with it.

Val, sadly left this world in July, 2008. He is survived by his wife and two daughters, all who still reside in the Seattle area. Val to this day is a topic for those of us who continue to work on A.P.P.L.E. items, and as he always believed in, attention to the details is the order of the day.



A.P.P.L.E. History Across 40 Years



Brief A.P.P.L.E. History

Apple Pugetsound Program Library Exchange (A.P.P.L.E.) was one of the first official Apple User Groups in the United States. The A.P.P.L.E. Users Group was established February 21, 1978 by Val J. Golding. The first meeting of the new users group was called to order at 7pm by Val at ComputerLand in Federal Way, Washington with 13 people in attendance.

A.P.P.L.E. published *Call-A.P.P.L.E.* magazine and others, published books, and provided software, hardware and support services for over 50,000 members in the Apple world through 1990. Distribution of *Call-A.P.P.L.E.* magazine reached over 100,000 in the mid 1980s. From 1990 to 2001, A.P.P.L.E. had a more limited focus providing support to computer users around the globe with many special interests.

Bill Martens, who worked for the founder Val J. Golding and A.P.P.L.E. from 1981 to 1982, started preserving the company's information and rebuilding the company in 1999 by contacting former writers, board members and staff. In February 2002, Bill continued this effort with a new issue of *Call-A.P.P.L.E.* magazine. However, this could not have been achieved without the help of Val Golding, Don Williams, Michael Thyng, Rick Sutcliffe, and Norman Dodge.

Bill continued being a visionary, promoting A.P.P.L.E., restoring its archives, and expanding its offerings. Bill is currently A.P.P.L.E. Chairman of the Board and the Club President, and is in charge of distributing *Call-A.P.P.L.E.* magazine, organizing the web site, and promoting news, among other things.

In 2003, Mike Pfaiffer joined with Bill, bringing his Digital Civilization news to the group and the magazine. He also joined the A.P.P.L.E. team as a director and served both as Vice President and as President of the group for a number of years. The content produced by Mike not only pushed the magazine into the Unix realm, but also allowed the magazine to continue during several years when Bill was working as a USMC contractor.

At this same time, Bill Martens purchased the remains of the WAC, Inc. users group from Neal Layton of the Willamette Apple Connection of Salem Oregon, including their newsletters and software library. It was brought under the A.P.P.L.E. umbrella with all of the libraries and newsletters digitized and placed online. A.P.P.L.E. has also had a hand in the production of a number of external projects as well, including the re-introduction of Nibble Magazine on DVD as well as the Eamon Adventurers Guild online.

In 2008, Brian Wiser joined the executive staff, and made significant art and manual additions to our official Beagle Bros site, as well as creating our Applied Engineering site with his manuals, catalogs, and his self-created brochures.

In 2013, with Bill, he produced *The WOZPAK: Special Edition* as A.P.P.L.E.'s first contemporary book. Since then, Brian continues to design, edit and co-produce all books published by A.P.P.L.E., encompassing several a year. Brian is an A.P.P.L.E. Board member, Managing Editor of *Call-A.P.P.L.E.* magazine, and also shares news and his interviews with many luminaries.

Detailed A.P.P.L.E. History

Profile of the Founder

Val J. Golding founded Apple Pugetsound Program Library Exchange (A.P.P.L.E.) in 1978 with the help of Mike Thyng and Bob Huelsdonk at the suggestion of Max Cook, a manager at the ComputerLand where Val bought his Apple II. Val also wrote for *Softdisk*, *On-three* and other technology magazines over the years primarily making his mark in the early years of Apple computing.



As the founder, Val was instrumental in guiding the company to the position it is in now. Val was the Managing Editor of *Call-A.P.P.L.E.* magazine and also served as the chairman of the board of directors. His wife and daughters were a big part of documenting his stories about his hobby of Cable Cars, and he was the editor of a highly acclaimed newsletter for his daughter's school. He passed away at age 77 on July 2, 2008 after a long battle with cancer.

The New Machine

In December 1977, Val Golding took his Christmas bonus and bought a computer. Although, he originally intended to buy the Chromeco, he caught a glimpse of the brand new Apple II computer which ComputerLand had just gotten in. After purchasing Apple II serial number 759 for \$1,800, he promptly went home and plugged it in.

Although the computer was completely self-contained, in order to connect it to a TV an RF Modulator was needed. This little item was a bit of a problem at times.

Once connected, the Apple would produce color the likes of which the world had not yet seen.

Val hacked on the computer for several weeks, calling Max Cook for questions which Val couldn't figure out. After several weeks, however, it became apparent that Val would have to find another source for information as the number of questions which Max was able to answer on each call was becoming fewer and fewer. As this happened, the roles became reversed with Max calling Val for answers.

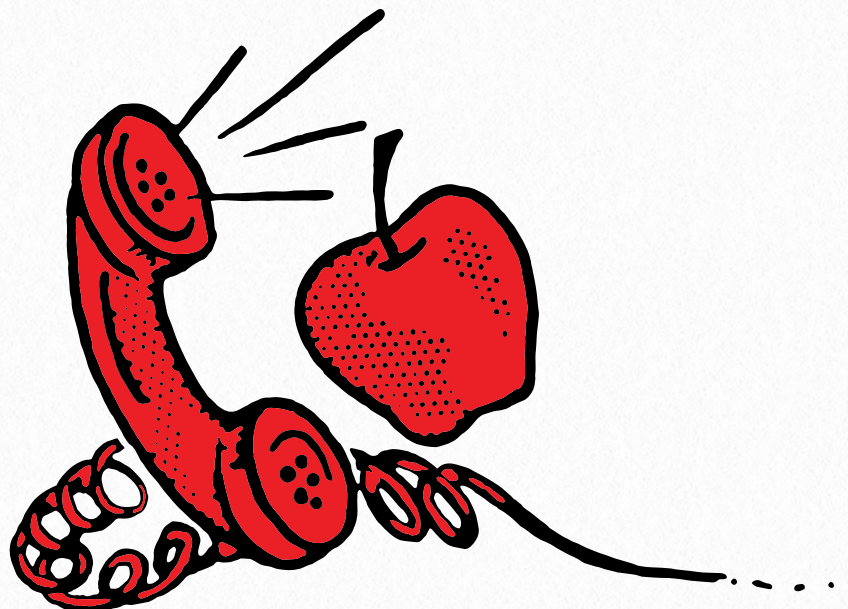
A.P.P.L.E. is Born

In January, during his usual trip to the ComputerLand store, Val met another Apple II owner, Bob Huelsdonk. Val and Bob talked for some length of time with Max Cook and decided that as information about the Apple II was scant, that they should start an Apple User Group.

Val created a one page notice of a meeting and distributed it to all of the Apple II owners who were on the customer lists of ComputerLand, Empire Electronics, and Omega Stereo. The date that was set for this meeting was the February, 21, 1978.

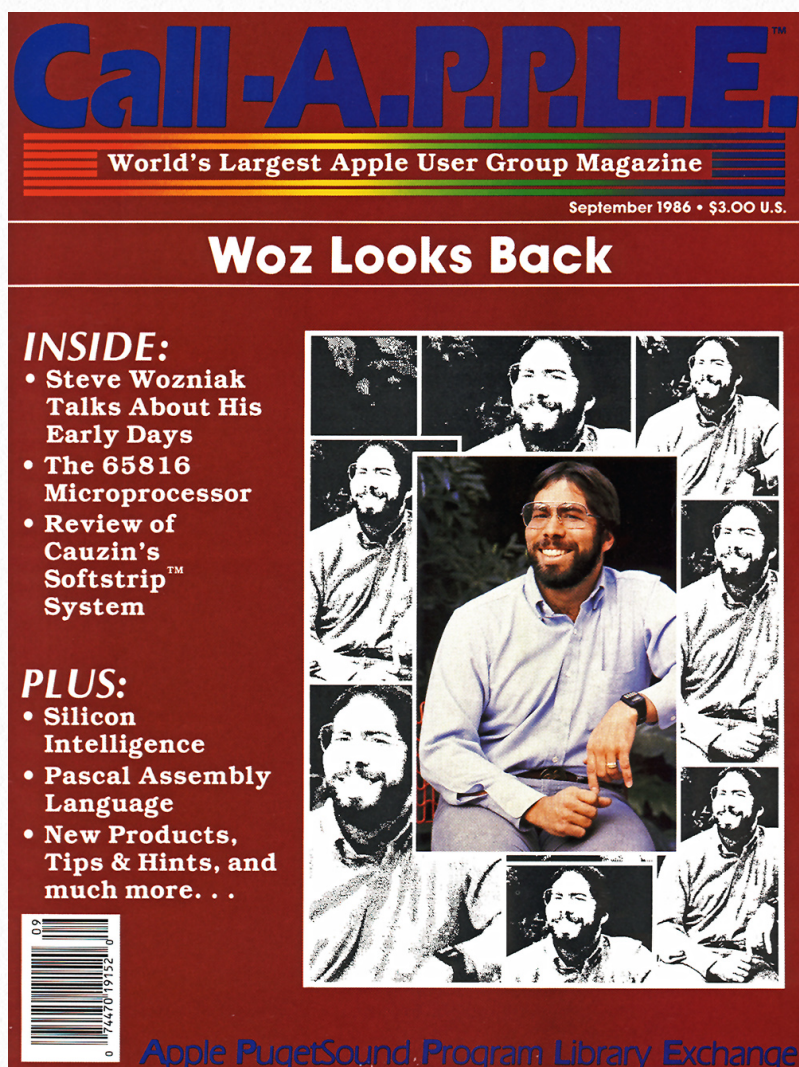
At that first meeting, about 20 people attended with 12 of them joining the group after the meeting. From this humble beginning, the group incorporated in 1979 as a non-profit organization and promptly grew beyond all expectations reaching the 5,000 member mark by 1980 and the 12,000 mark by 1981.

As for the newly-minted user group name A.P.P.L.E., Val said, "I started with the acronym and just played around with it until I found something that fit. I was never happy with combining "Pugetsound" but I never figured a better word. The "Program Library Exchange" of course, was a natural, because that was what we started out doing."



Call-A.P.P.L.E. Magazine

The *Call-A.P.P.L.E.* newsletter, which was established in February 1978, also continued to grow as did the group and the availability of information for the Apple II. The initial flyer, produced for the February 21st, 1978 meeting, became *Call-A.P.P.L.E.* Vol. 1 No. 1, and was designated as such only after the fact, when the second issue came out.



According to Val Golding, "No. 1 in reality was a form letter mailed to customers as an invitation to form an Apple Computer user group. It seems to me we were just a handful of people and I offered the name and we all informally said 'That's fine' or words to that effect. I'm thinking *Call-A.P.P.L.E.* Vol. 1 No. 1 may have had that at the top of the page already."

By the end of 1978, the newsletter had reached 20 pages in length. It became a 24-page printed newsletter in January 1979 and soon thereafter a step further – becoming a 36-page magazine with glossy covers in April 1979. The year finished with the November/December issue reaching 58 pages and was on the road to becoming an even more important technical information resource.

Dick Hubert, who became involved early in 1978, volunteered to help with production of the newsletter. He did all of the production of the magazine, such as putting labels on them and preparing them for mailing from his living room and with the help of his family. Between Val staying up all night to print the labels for the magazines and his home production, Dick managed to get the magazine copies to the post office in time – even in spite of the number of newsletters reaching almost 4,000 members.

This production work generally occurred on the weekend beginning Friday evening with the mailing taking place on Monday. But this was all a tough chore which was done each month and as Dick said, "At this point, we figured it was time for some changes."

The Programmers

Early on in 1978 and 1979, there were several people involved with the group who produced many of the software packages which were made available to the members. Much of this software was facilitated by the relationship between Synergistic Software and A.P.P.L.E.. Bob Clardy, president of Synergistic Software, took on the A.P.P.L.E. treasurer duties at one of the early meetings, making it clear that he would do it because no one else wanted to take on the responsibility.



However, his move would be the one thing which made the group get its books in order and also allowed the group to grow and get out of Val's basement. It was also Bob's influence which led to the hiring of Fred Merchant to handle the finances of the company on a permanent basis.

Upon turning over the responsibilities of the treasury over to Fred, Bob went back to his focus on the software side of things. He started Synergistic Software and promptly hired Darryll and Ron Aldrich as well as Neil Konzen to write software for the Apple II. Much of the software written by the three young programmers eventually became the primary packages used by thousands of A.P.P.L.E. members.

Others involved in the creation of software for A.P.P.L.E. were Bob Huelsdonk and Don Williams. Bob wrote the very first word processor for the Apple II computer. Don was primarily a machine language programmer who wrote software for one of the earliest expansion memory cards available.

The A.P.P.L.E. Hotline

The A.P.P.L.E. Hotline was always a good source of help when the members were challenged with their computing needs. Although this was run in Val's basement, the fact that he answered the phone directly meant that the members were benefiting from all of Val's experience with the Apple II, as well as any information he could scrape out of the hands of Apple Computer Inc.

On the formation of the hotline, Val said, "A few of us, including Mike Thyng and Bob Huelsdonk bought our computers from Max Cook, a manager at the ComputerLand where I bought my first Apple II, and naturally I was calling him with loads of questions at first, most of which he couldn't really answer. A couple of us, including Mike Thyng and Bob Huelsdonk bought our computers from Max and naturally I was calling him with loads of questions at first, most of which he couldn't answer. Bob was pretty savvy, having been in the business (with Honeywell) for quite some time and was able to point me in the right direction for the answers. Max, although a nice guy, was "just" a salesman. Eventually he began calling me (as I started developing some Apple technical expertise) with questions and that was how the A.P.P.L.E. hot line got started."

Val continues, "After the club was formed I gave out my phone number as the hot line number and many times I'd answer it in the middle of the night (from the east coast, etc.). The calls I got most often were from somebody who

had inadvertently or otherwise, deleted their program (Integer or Applesoft, either one) and I'd talk them through manually resetting their program pointers to recover the program. Even Todd Rungren, still an active rock star, used to call frequently."

Changing of the Guard

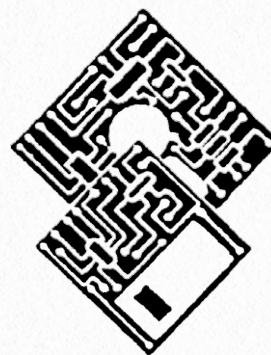
By the time Val turned over the editorial duties to Kathryn Hallgrimson Suther in mid 1984, the group had grown to an incredible 25,000 members. With this number of people came an incredible number of problems as well.

No longer was the group able to maintain the small cozy feeling of the early meetings.

Many of the meetings became too much for Val to handle and he took the step of leaving to see what else he could do in life. Thus, he left the group which he founded and turned over the reigns to another early member, Don Williams.

With the departure of Val came a real change within the company and many of the associated problems which are normally associated with such companies. The success of the group had been noticed.

Apple Programmers and Developers Association

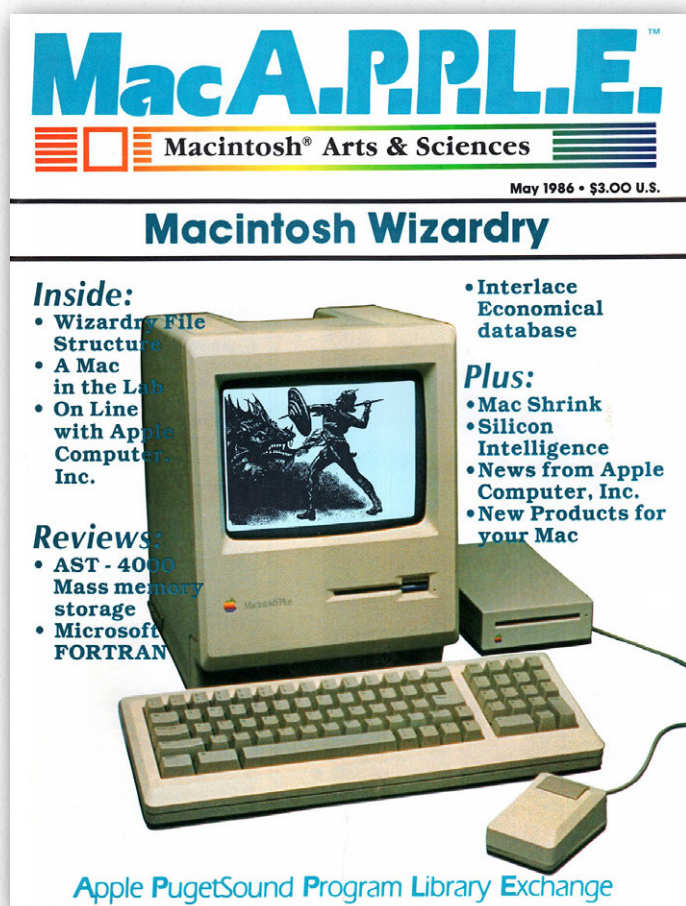


In 1985, Apple asked A.P.P.L.E. to create the Apple Programmers and Developers Association (APDA). Guy Kawasaki and Dan Cochran came out to Renton, Washington to sell the idea of A.P.P.L.E. distributing Apple's documentation and code for the developers – produced up to A.P.P.L.E. standards. It was at this time that a decision was made and that APDA was created. With Don Williams at the helm of the new group, APDA flourished and in 1988, the group was spun off from A.P.P.L.E. and sold back to Apple.

The Co-op Years

In 1985, a taxation dispute with the state of Washington brought about a change in status of the group from a non-profit organization to a standard company. The name was changed from Apple Pugetsound Program Library Exchange to A.P.P.L.E. Co-Op as the state revoked the non-profit status of the group.

The Macintosh was introduced by Steve Jobs in January 1984 to great fanfare leading A.P.P.L.E.'s Lisa and Michael Storrie-Lombardi to create the Lisa / Macintosh Special Interest Group within A.P.P.L.E.. Lisa and Michael followed up this interest with a newsletter entitled *32 Little Apples* "News For the Rest of Us." The newsletter began in November 1984 and then in March 1986, became *Mac-A.P.P.L.E.* magazine with Andrew Himes at the helm.



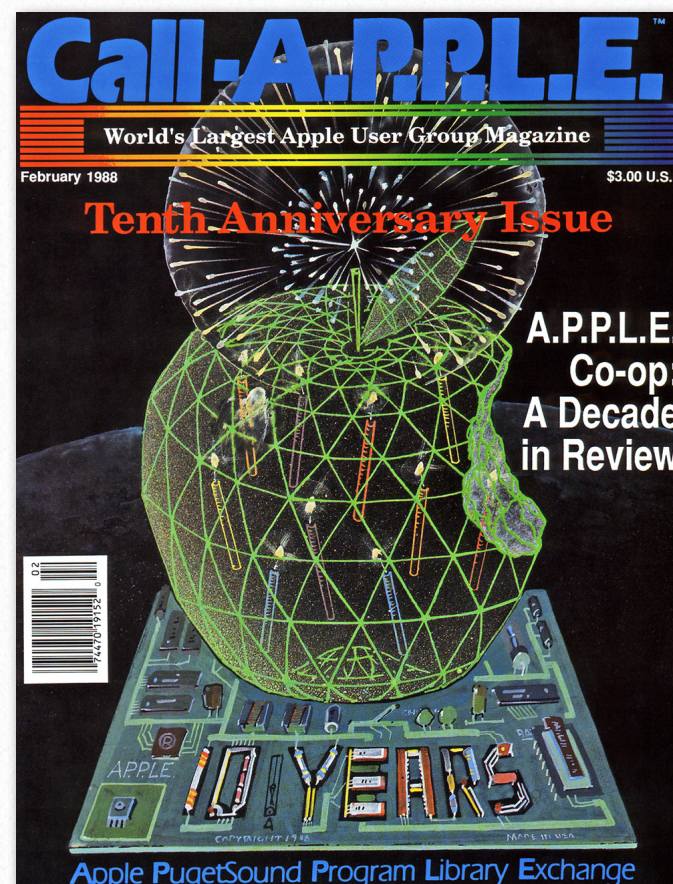
The expanded interests that came with *32 Little Apples* and *Mac-A.P.P.L.E.* magazine also included the expansion of the group's interests into the PC world as well as the Commodore Amiga. During this expansion time, Frank Catalano took over as editor of the fledgling magazine.

However, the expansion ideas and actions didn't last very long. *Mac-A.P.P.L.E.* ran from that time until November 1987, when the name was changed to *Mac Horizons*. The magazine, while filling a need in the market, was not widely read and publication eventually ceased in October 1988.

With the demise of *Mac-A.P.P.L.E.* / *Mac Horizons*, a decision was made to produce *Mac Tech Quarterly* instead of the monthly magazine. This too, however, was short-lived and only five issues of *Mac Tech Quarterly* were produced prior to the publication ceasing in 1990.

In February 1988, A.P.P.L.E. celebrated its 10th anniversary with a 16-page extravaganza in *Call-A.P.P.L.E.* magazine featuring interviews with many early members of the group. Dick Hubert, who was the Executive President of

the Co-Op at the time, mentioned the expansion and a focus of trying to get better speakers for the group.



Two years later, the decision was made to change *Call-A.P.P.L.E.* magazine to publish quarterly. This decision essentially put the magazine on a spiral of non-production and into the historical category. The combination of this decision, the general decline of computer magazines in the late 1980s, and the decision by Apple Computer, Inc. to discontinue selling the Apple IIe and IIGS made interest in the group wane severely. Additionally, one board member made it his personal objective to essentially kill the group in a feeble attempt to acquire its assets. Eventually the office assets went to auction and the office was closed.

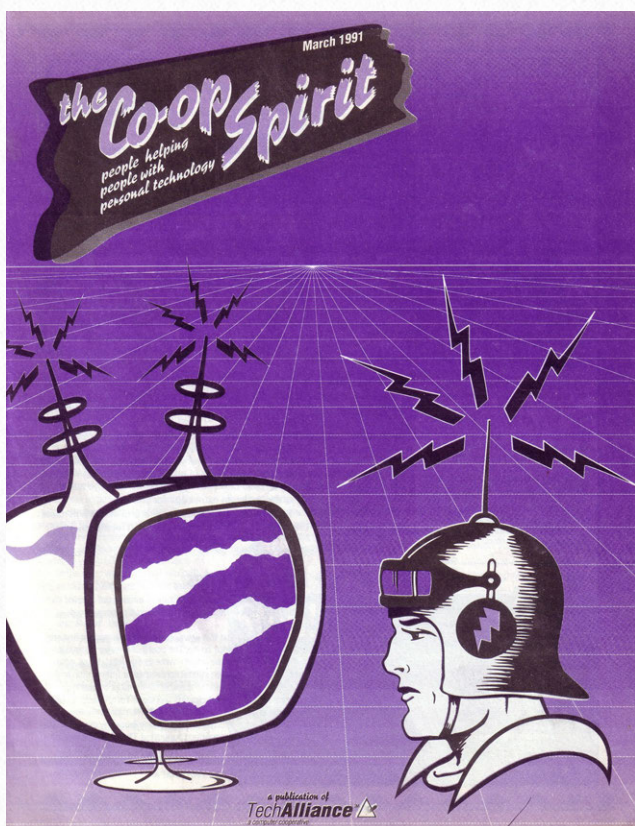
The End of an Era?

The August 1989 issue of *Call-A.P.P.L.E.* magazine became the Autumn Issue. It was to be the final issue of 1989 with only one further issue coming in late 1990. The date on the cover of the final legacy issue is Winter 1990. In the issue, Val wrote a special note about the ending.

With the ending of the magazine came the end of the computing world that had changed from that of the hobbyists and engineers being the only ones using computers, to that of a computer being in almost every home in America. Although this trend would continue in the 1990s with the PC revolution taking over where the Apple revolution left off, the spirit of the early years would never return. Or would it...

Tech Alliance Years

The remaining members of the group formed the Tech Alliance Co-op, the moniker under which the group continued to operate, and continued publishing a newsletter and distributing software and information to about 6,000 members. This information exchange was oriented around several special interest groups, guided by Dick Hubert and a number of others who supported the group. It was during this time in which the *Co-Op Spirit* newsletter was published on a monthly basis by various people in the group.



The Bulletin Board System, which served as the group's main form of communication, continued until 1997 under Norman Dodge with some of the SIG meetings continuing until 2001 including the Apple II and IIGS SIG managed by people like Bill Bredehoft.

The Rebirth

In 2002, Bill Martens made contact with Norman Dodge and began recreating the group, eventually forming what is now the A.P.P.L.E. User Group in coordination with Val Golding and Rick Sutcliffe, with initial materials supplied primarily by Michael Thyng, Norman Dodge, Val Golding, Bill Bredehoft and from Bill's own collection which was built over the previous two decades.

With Val and Rick's help, Bill set out to re-start the magazine, successfully producing the first new generation issue in April 2002. It marked the first time since 1994 that an A.P.P.L.E. publication had been produced.

WAC, Inc. Acquisition

In 2002, Bill Martens purchased the remains of the Salem Oregon based Willamette Apple Connection or WAC, Inc. from Neal Layton. This material was then incorporated into the A.P.P.L.E. realm of websites and all of the materials digitized and made available to A.P.P.L.E. members.

Digital Civilization News

In 2003, Mike Pfaiffer joined bringing Digital Civilization news with him. This new material allowed A.P.P.L.E. to include a spattering of UNIX news in addition to the retro Apple news that had already been a large part of the re-invented group. Mike worked with the group until his passing in 2015, serving for a time as Vice President and then as President of the user group. He also produced a number of the group's magazines from 2004 to 2007.

Golden Grail

In 2003, Bill Martens and Jim Maricondo began looking at the possibility of bringing back The Golden Orchard CD-ROM which was a big seller in the 1990s for Apple IIGS users. Jim and Bill worked on the project for over two years, finally making the entire collection available again and re-formatted for current-day emulation.

Their next project was the initial digitization of what became the *Woz Speaks* DVD. This project began with a loan of an original tape of Steve Wozniak speaking to the A.P.P.L.E. User Group at the monthly meeting in October 1981. The project lasted several months, and with the resulting DVD, the product returned to the A.P.P.L.E. catalog.

The New Breed

In 2008, Brian Wiser joined the executive staff, and made significant art and manual additions to our official Beagle Bros site, as well as creating our Applied Engineering site with his manuals, catalogs, and his self-created brochures. Brian is an A.P.P.L.E. Board member, Managing Editor of *Call-A.P.P.L.E.* magazine, and also shares news and his interviews with many luminaries.

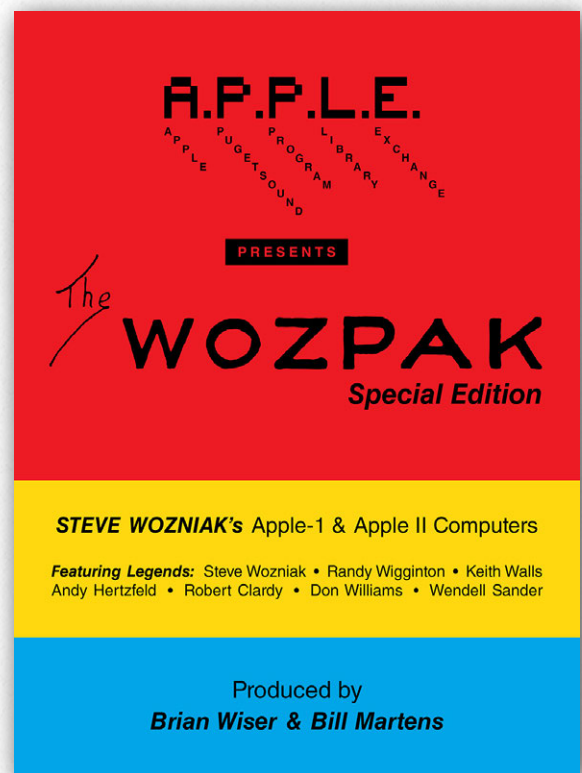


In 2013, Brian and Bill produced *The WOZPAK: Special Edition* as A.P.P.L.E.'s first contemporary book. The fully-restored book took over eight months to produce, starting with a meeting between Bill and Steve Wozniak in Tokyo, culminating with Brian and Steve, along with Randy Wigginton introducing the book to crowds at KansasFest 2013.



Brian continues to design, edit and co-produce with Bill all books published by A.P.P.L.E., encompassing several a year. They go to great lengths for quality, detail, new features, fresh design, and work with a variety of noteworthy authors. Between 2013 and early 2018 they published 13 books encompassing:

- *All About Applesoft: Enhanced Edition*
- *The Apple House: How to Computerize Your Home*
- *The Apple II Monitor Peeled*
- *Call-A.P.P.L.E. Magazine 1978 Compendium*
- *Call-A.P.P.L.E. Magazine 1979 Compendium*
- *Colossal Computer Cartoon Book: Enhanced Edition*
- *Cyber Jack: The Adventures of Robert Clardy and Synergistic Software*
- *GBBS Pro Bulletin Board System: Version 2.2*
- *Nibble Viewpoints: Business Insights From The Computing Revolution*
- *Synergistic Software: The Early Games*
- *Turtlesoft: Turtle Graphics for Applesoft*
- *What's Where in the Apple: Enhanced Edition*
- *The WOZPAK Special Edition: Steve Wozniak's Apple-1 & Apple II Computers*



In 2016, Brian and Bill produced an iOS version of the Apple II game *Structris*. They collaborated with original Apple II programmer Martin Haye, iOS programmer Olivier Goguel, and neo-classical composer Tomoki Takamori.

Brian Wiser and Bill Martens continue to publish additional materials related to Apple computers, including managing all of the A.P.P.L.E. websites, creating and editing *Call-A.P.P.L.E.* magazine, as well as managing all of the aspects of the A.P.P.L.E. User Group.

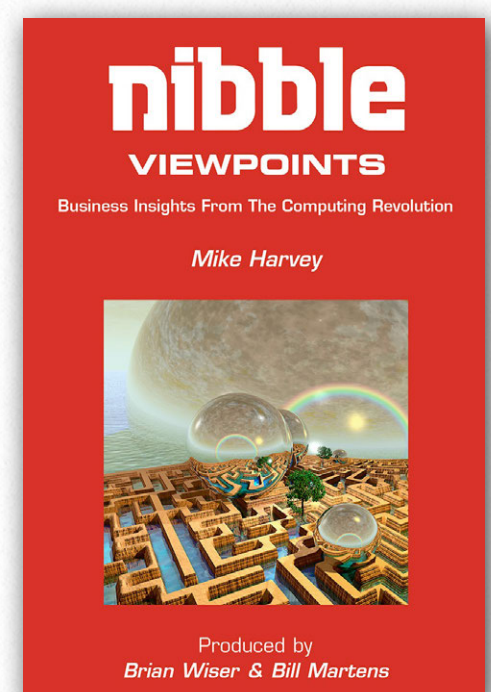
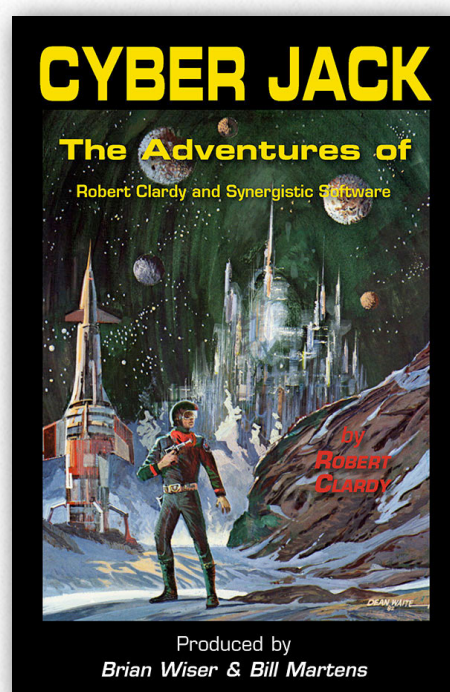
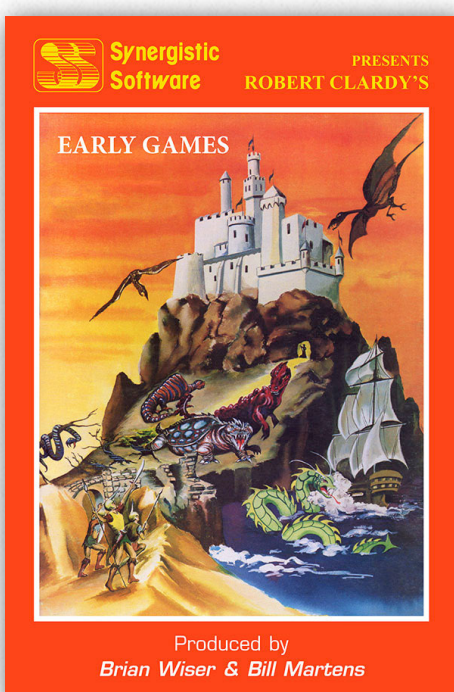
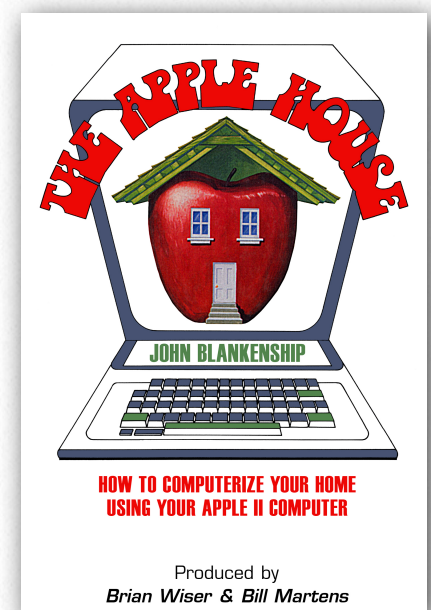
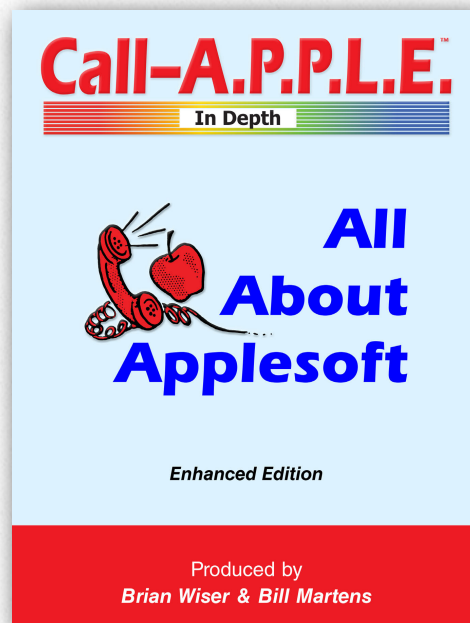
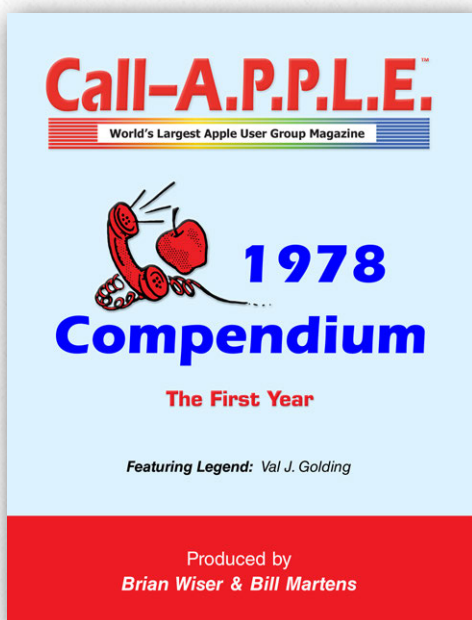
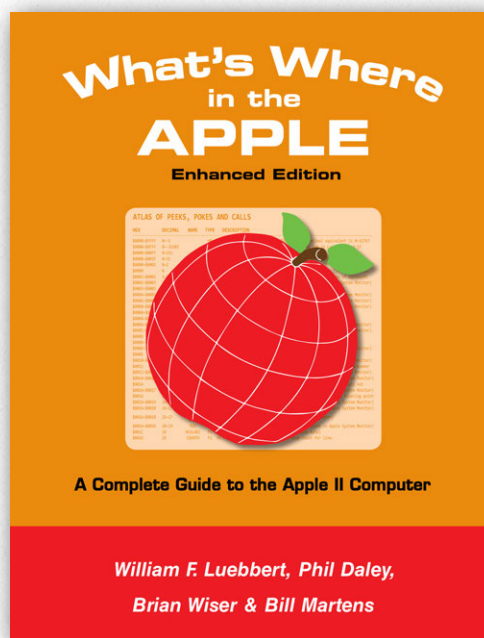
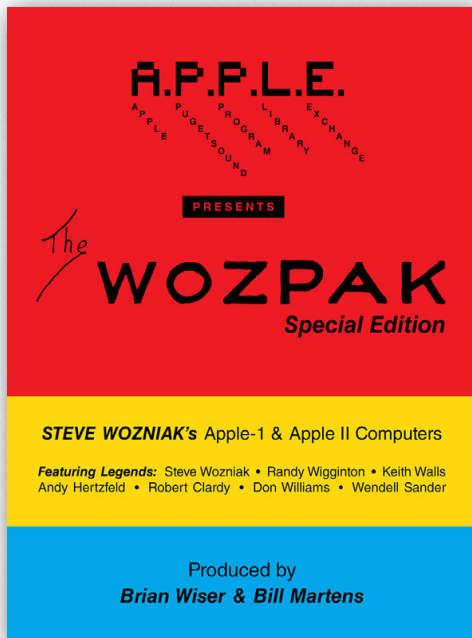
The Future

This year marks the return of a host of application software packages from A.P.P.L.E. in addition to *Blankenship BASIC*, the recovery of the original A.P.P.L.E. Crate BBS hard drives among other items. John Morris, the creator of Applesauce, is assisting with this. He also assisted with the recovery of the original Twilight II source code this past year along with Jim Maricondo.

Also on the horizon is *All About Applesoft: Enhanced Edition* with additional materials making this book the must-have book for every Apple II programmer who is using Applesoft as their primary means of programming. There are a few eBooks which will also be making their way onto the scene with their appearances in the Apple iBooks Store. Stay tuned for more Apple goodness from A.P.P.L.E.!



The Future Looks Bright



Get your copy today in Paperback and Hardback
www.callapple.org/books

nibble

VIEWPOINTS

Business Insights From The Computing Revolution

Mike Harvey



Produced by
Brian Wiser & Bill Martens

“I’m excited about my new *Nibble Viewpoints* book that is an organized culmination of my editorials from 12 years of *Nibble* magazine and over 30 years of experiences running large corporations. It contains a timeless array of shorthand management models that are interesting, powerful, and easy to use. The models cover a wide range of problems and solutions for businesses and individuals – they’re short, easy to read, and sharply focused on workable solutions. *Nibble Viewpoints* also traces the rise and fall of the Apple II computer over its amazing history. I’m looking forward to people reconnecting with *Nibble* and finding guidance and insights to help them with their endeavors.”

Mike Harvey

Publisher of Nibble Magazine

callapple.org/books

40 Years On – My Memories of Val

Bob Clardy looks back at the early years of A.P.P.L.E..



Val Golding

by Bob Clardy



I was digging into my old notes about my earliest days with the A.P.P.L.E. user group and found a few things in my [Cyber Jack](#) autobiography were not quite right. I purchased my first Apple II in August of 1978 and went to my first A.P.P.L.E. meeting the following month.

A.P.P.L.E. had been founded in February of 1978 by Val Golding. By

the time I went to a meeting, Dick Hubert, Bob Huelsdonk, and Mike Thyng were the only other "grown-ups" I remember from those earliest meetings. They each worked with local stores to get the sales folk there to refer new Apple buyers to the club for support. Dick was club president while Val was the editor of *Call-A.P.P.L.E.* magazine. Val was also the lead evangelist who corresponded with Apple Computer luminaries and fans, seeking information to publish and share with other Apple users. In those days, the folks at Apple were eager to share their notes and sketches. A.P.P.L.E., in turn, organized, edited, published and distributed it to everyone they could. Apple and A.P.P.L.E. both thrived.

Today, with Apple being one of the wealthiest companies on the planet, it is difficult to realize how primitive that early company was compared to tech startups of more recent decades. The computer power-houses of 1978 were

Magnavox and Radio Shack with Apple being a rather under-funded upstart with little time and manpower to spend on documentation. Clubs like A.P.P.L.E. sprang up to try to fill that lack and help the aspiring Apple users learn enough about their hardware to produce the software that would eventually become a major industry itself.

That first meeting I went to in August 1978 had about 30 members attending. Dick was president and Val was the editor and they were looking to recruit others willing to work on behalf of the club. Most attending were teenagers, more interested in how to get more and better games for their computer, rather than working for the club. I was 26 and one of the older members present, so I was a target for Dick and Val to recruit. I became club treasurer at that first meeting. I tracked dues and deposits and product costs and paying the bills associated with a magazine. It quickly became a significant task which I enjoyed immensely for years as I also started Synergistic Software, my own Apple software company.

As an A.P.P.L.E. officer, one of my minor duties was to present the treasurer's report to the club members during each meeting. Since members came to learn about the Apple, the business bits of the meeting were not overly popular. We went through them as quickly as we could, but it was not until I started presenting the reports in hexadecimal that there was any real enthusiasm from the members, as they competed to convert my report before others. It was silly, but made the business part of the meeting a bit more tolerable.

During the following years, a spinoff group called APDA (Apple Programmers and Developers Association) was started. It worked very closely with Apple corporate, getting officially-released documentation that was then published by APDA. That, eventually, was taken in-house at Apple as their staffing caught up with the need for such things. As that side of operations dwindled, A.P.P.L.E. started its own store (Apple peripherals, mods, software, books, etc.) and started semi-formal classes to teach programming and game design. All of us that worked with A.P.P.L.E. taught, wrote articles and wrote software. Val Golding gathered everything, organized it, edited it, and published it to share

with the larger Apple community beyond our little club. At that time, the *Call-A.P.P.L.E.* magazine was one of the premier computer magazines being published. While the magazine had many contributors, it was Val Golding that wrote the most, edited it all, and solicited topics from his many connections around the industry.

Val Golding was a real anomaly at the club. Born in 1930, Val was an ancient 48-year-old when I first met him. He put me in mind of a bohemian poet, with a scraggly beard and the yellowed teeth and fingers of a hard-core chain smoker. He looked so out-of-place in the techie environment that he attracted attention easily. His friendly charisma and willingness to talk with anyone about the Apple insured that he always had a host of contacts to mine for stories and information that he could then share with others. Of course, there is no telling whether it all actually got out there. He was certainly prolific. But, when I visited him at his home, to hand off some report or article, I had to run the gauntlet that was his work space. His Apple and desk were in his basement, and 95 percent of that room was crammed at least waist-high with ... well, stuff. There was hardware of different sorts, but most of it was the general detritus that a near hoarder accumulates when they have too little space to keep it well organized. There were narrow pathways that one could walk through to get around that basement. You had to move carefully so as not to tip over any of the many piles. I always wondered whether letters or documents that he received may have been lost in some pile and buried by later accumulations. It was always an adventure to visit there and see what treasures might be revealed.



Val Golding's Office

One last story I have about Val involved the trip I shared with him in the early 1980s. It was supposed to be a business opportunity to form contacts in Japan and China for A.P.P.L.E., Synergistic Software, and several other early American software companies. We went to Tokyo, Taipei, and Hong Kong, attending trade fairs and meeting with local business folk. I believe Val made good use of the trip and formed new information sharing contacts. For myself, though, it was more of a junket. I loved the meetings, but

did not make any new business contacts. It was fun, though. I have fond memories of the bustling cities, shrines and architectural wonders, and crowds of friendly people. It was in some small store in Akihabara that I found my earliest pirated software – a copy of *Odyssey* that had been translated into Japanese, republished and sold openly in the stores there. I purchased one myself. Maybe I should have tried a bit harder to ally Synergistic with some local company so I could at least participate in those sales. But, at the time, I was just flattered that anyone cared enough to make that effort.

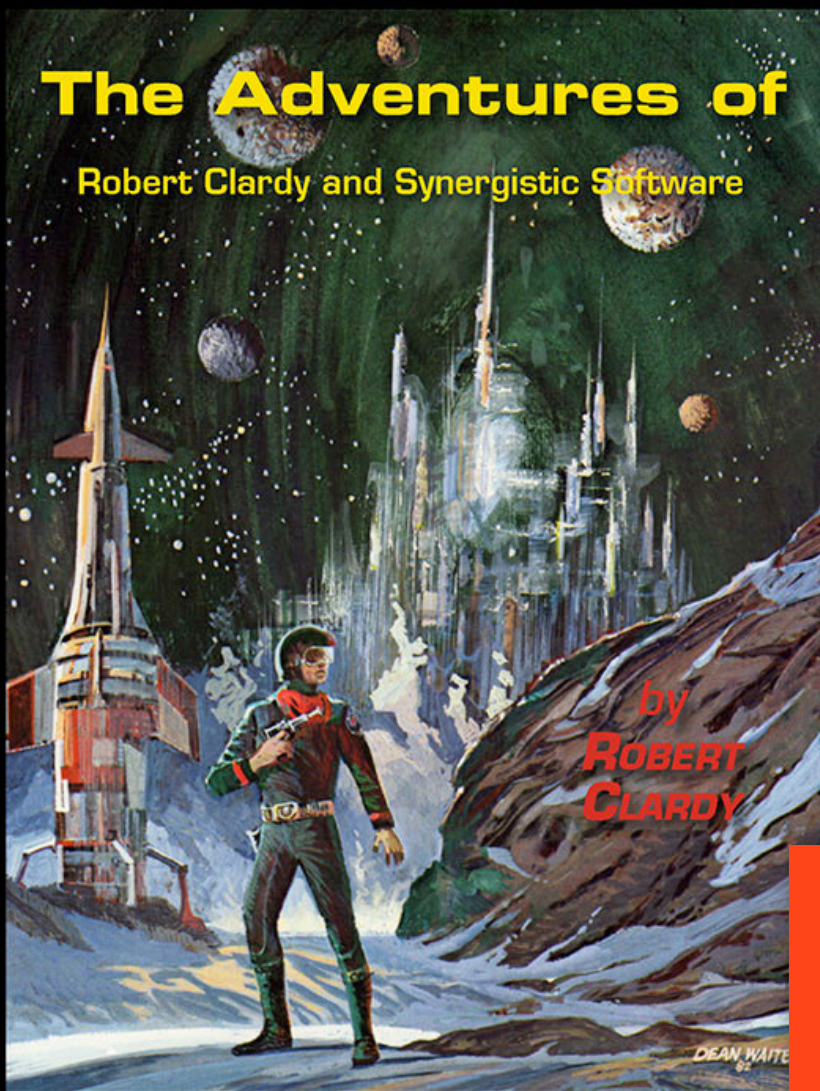
The other cultural awakening we shared was with the school children of Japan. Any time we went to a shrine, temple or other relic of Japan's past, we encountered groups of school children on field trips to see those same sites. They were always dressed the same and incredibly well behaved, walking in line behind a chaperone or two holding flags high. I was amazed that such large groups of young children were so polite and organized. They were also, totally fascinated by us. We weren't just foreigners – we had beards. That was pretty uncommon in those areas and apparently identified us as somewhat barbaric and exotic. When a smiling child screwed up the nerve to test his English on us and say hello, we always stopped to reply. That inevitably led to the entire line of school kids forming up to see us, say hello, and shake hands. They were totally fascinated and we had a blast. Val, with his shaggy hermit look, was always the biggest draw. The chaperones helped us with some polite phrases that we could say to the kids while they, in turn, tried out their English phrases on us. It would sometimes take 20 minutes or so for an entire group to greet us, one by one. In some cases, Val and I would separate a bit so each of us could interact with a different group. There were a lot of them.

Needless to say, we had a great time and I do wish I had made better use of the opportunity to expand Synergistic Software. But, it was just too much fun to spoil it with actual work. I'm afraid I still saw the Apple as a hobby that paid some bills, rather than business that I could grow into something international. While that may have been a poor business decision, it did insure that the memories were all fond ones. There were no failed deals or anguish about the software piracy. It was good fun and sharing with others as best we could and that was what mattered the most.

That's all for now. Best wishes.



CYBER JACK



Produced by
Brian Wiser & Bill Martens

Get your copy today!

Available in
Paperback and Hardback

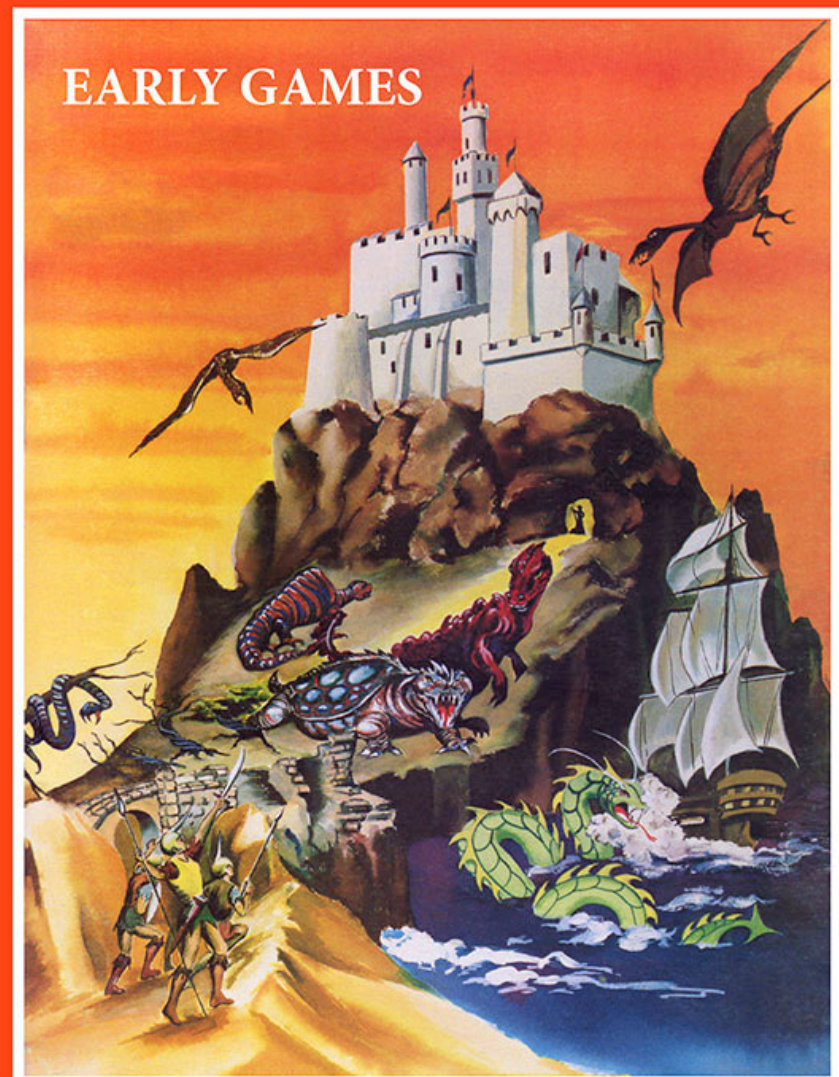
www.callapple.org/books

New from
A.P.P.L.E.
and
Robert Clardy




**Synergistic
Software**


PRESENTS
ROBERT CLARDY'S



Produced by
Brian Wiser & Bill Martens



You Have Died of Dysentery



The creation of
The Oregon Trail

– the iconic educational game of the 1980s

R. Philip Bouchard

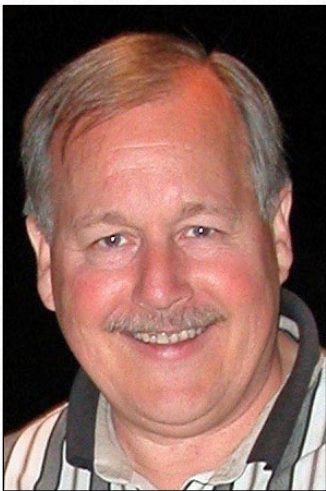
[HTTP://PHILIPBOUCHARD.COM/](http://philipbouchard.com/)

The Founding of APDA

The Apple Programmers and Developers Association: How A.P.P.L.E. Created an Institution

Apple asks A.P.P.L.E. to handle APDA and its distribution to the world.

by Don Williams



It was the best of times and the worst of times for the A.P.P.L.E. user group. We, as a club, had a large group of members who were all excited about the treasures that this new Apple personal computer would surely deliver. Only board-certified optimists were allowed in the realm of Call-A.P.P.L.E.. If any one of us found a pile of road

apples, we would dig through it to find the pony that had to be in there. And then to rain on this sanguine parade, Apple's lawyers were telling us our name infringed on their copyright and we should change it or suffer the slings and arrows of their outrageous tribulations. We had grabbed a tiger by the tail and it got our undivided attention. *The Crazy Ones* by Rob Siltanen seems to apply to A.P.P.L.E. back in the day:

"Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They are not fond of rules. And they have no respect for the status quo.

You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward.

And while some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do."

I have a chart on my wall of a poem that was an Apple product done for APDA in 1985, written by Rich Binell and produced by Clement Mok. It too captures a feeling of A.P.P.L.E.'s spirit:

Ode to the Latter-Day Wizard

*Sun and stars
are rarely seen,
behind the screen.*

*All the day
and all the night,
In darkness broken
by flickering light.*

*Mix rust and dust
and sand and sweat,
And spells that others
don't know of yet.*

*Journey dark
untraveled roads,
With lead and light
and secret codes.*

*With yes and no
and no and yes,
Lure much from naught,
and more from less.*

*With balanced fire
on fingertips,
And words that pass
no mortal lips:*

*Words that hint
at wealth untold,
From turning bits
and dreams to gold.*

There was a huge demand for documentation on Apple internals, yet Apple had none for the masses of club members that were hungry for them. Dr. Stillman, more as a favor to Apple and some of the more technically qualified members of Call-A.P.P.L.E., pencil-whipped a few of the Apple internal documents. Apple software developers and engineers had a habit of writing extraordinarily good code and designing state of the art hardware, but like all good engineers, documentation was an easy to overlook requirement. Dr. Stillman was an extraordinarily good

technical writer. He took great pride in his work and it showed. Apple took note and asked for more.

In defense of the Apple internal documentation, most of the things really needed can be written in a few words for the people who don't really need the documentation for other than reference. This resulted in documents that were written in a linguistic landscape somewhere between technobabble and English. For themselves and fellow travelers, this was not an issue since most documentation becomes shelfware, and that is only used to decorate the shelves of the wizards that flog the code day in and day out. They needed the documentation only to refresh their memory of the incantations that they had developed on those long sleepless nights while they were accomplishing the magic needed to meet impossible deadlines.

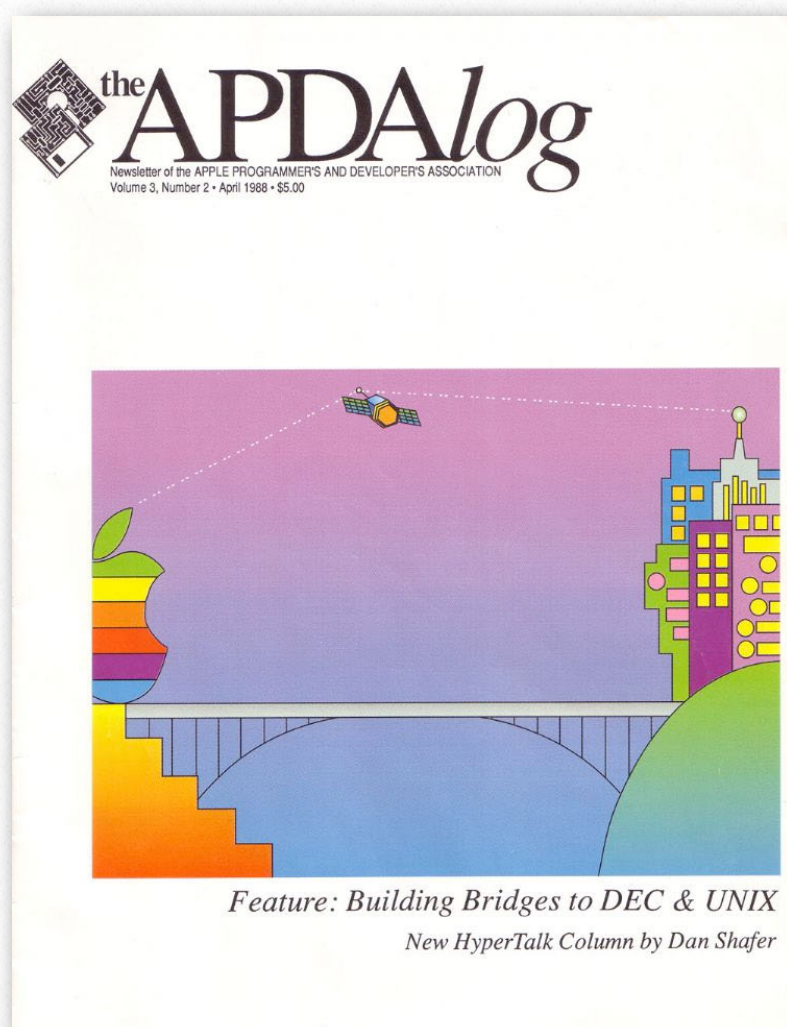
Apple had a philosophy that involvement from the clubs supported their sales. Apple was good to all of the clubs and provided speakers and the availability of software support from members of the Apple software teams. In spite of Apple's positive relationship with the club, they had rudely sent us cease and desist letters relating to our name. Exploiting that positive relationship behind the scenes, Richard Hubert worked out a deal with Guy Kawasaki at Apple (one of the User Group Gurus) to rewrite technical material from Apple and sell it to our members and give them to Apple.



Guy came up to the Call-A.P.P.L.E. building in Kent and pitched the offer to the Board. Most of the board's questions were related to, "can we pull this off and how will it affect our magazine and other services to our members?"

Guy said Apple would provide whatever equipment was needed and all of the technical support needed to produce a quality document, along with any other resources that we might need like funding for staff. I believe Guy offered to have the lawyers stop the harassment on the name issue. The Board was kind of bowled over. Looking back on it, this was a huge challenge, but angels go where wise men fear to tread. This sounded like it was too good to be true. The Board, at the advice of our counsel Bob Walerious, agreed.

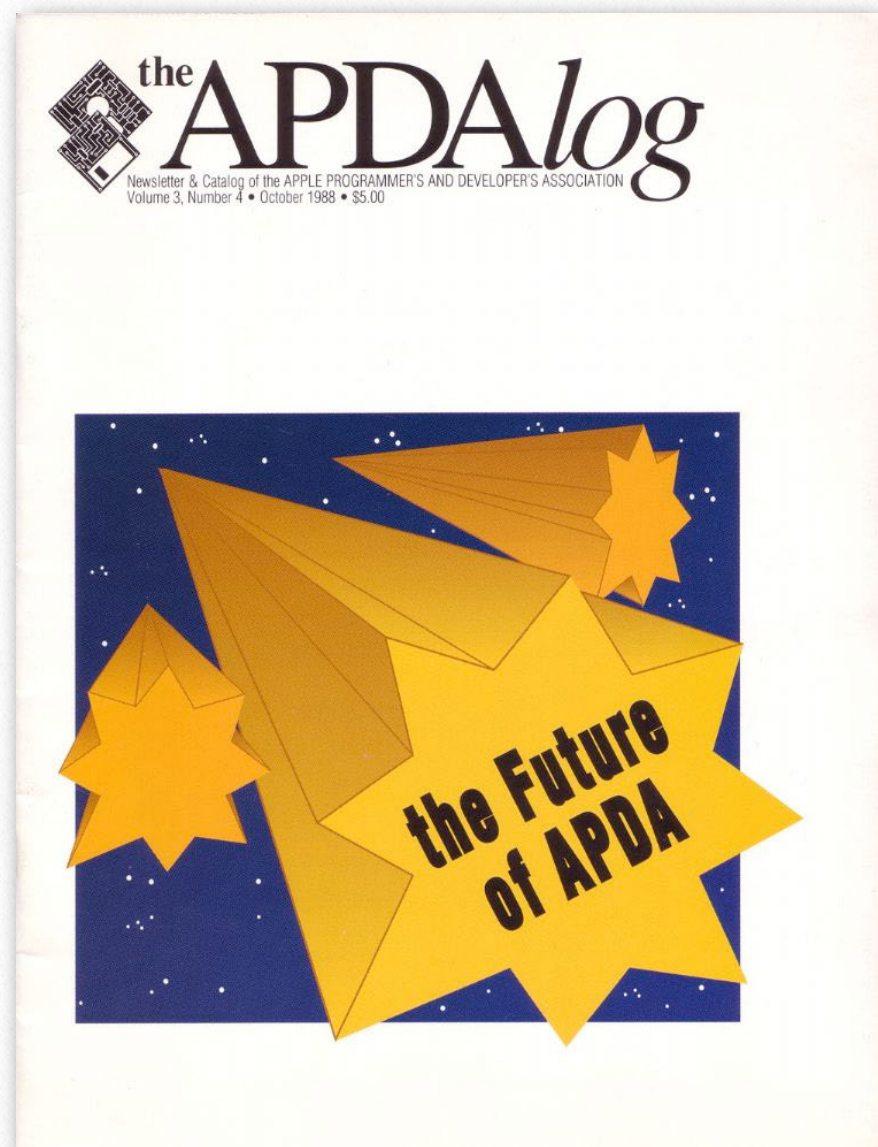
The Board, on the advice of Richard, decided to form a separate organization, with an only slightly better name, Apple Programmer's and Developer's Association. Apple came through with everything and more. Everyone was trying to figure out how to monetize their interest in the



Apple Computer, and APDA provided a guiding light to that process. Before long our user group had 40,000+ members and we were selling Apple technical documents as fast as we could produce them.

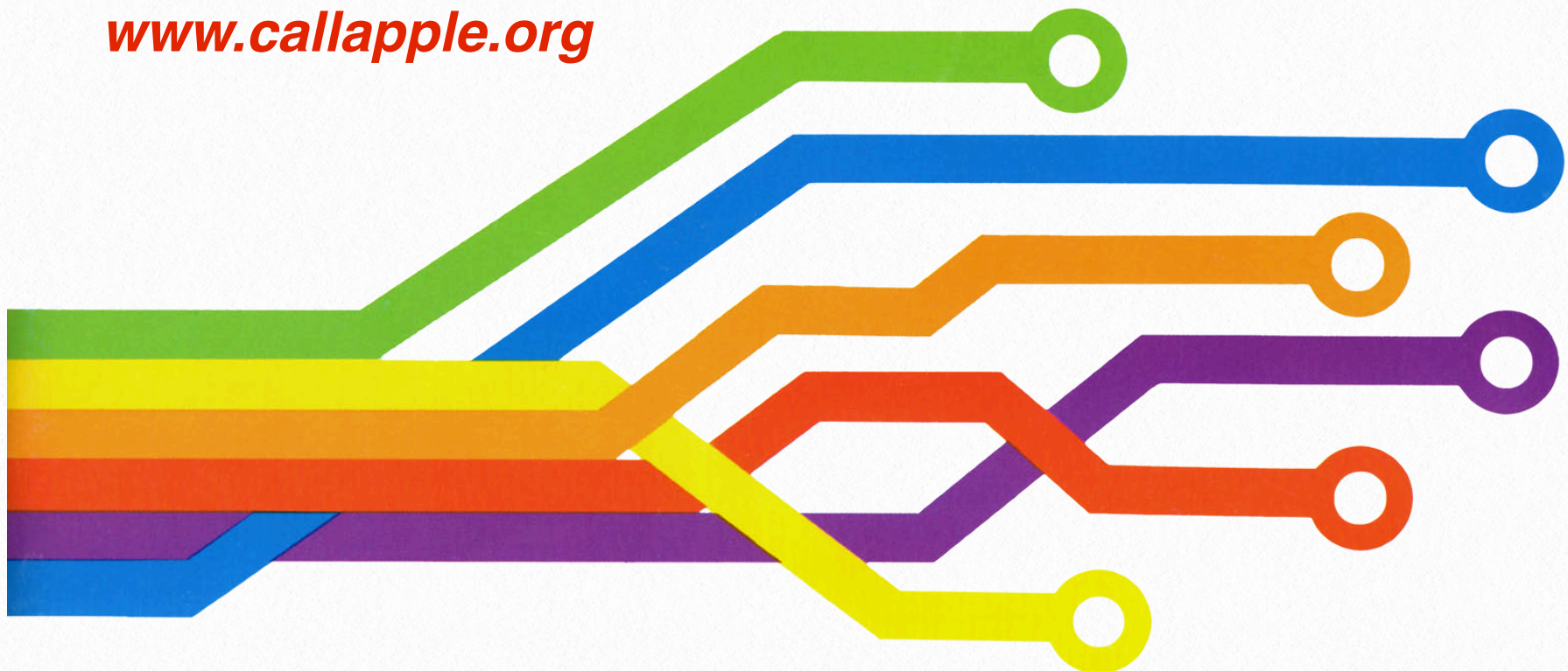
You could think of this business model as someone giving you material in one language and your translating it to another language and selling it to a customer set that bought everything that you translated at whatever price you decided was fair. We were the media kingpins of our day. Kind of like prostitution, what was of value came to you organically through no fault of your own. You could sell it and you still had it to sell again.

We were in fact supporting the largest community of developers likely ever seen on the planet. These were the golden years and Apple turned out to be the best business partner we could have imagined.



40 Years of A.P.P.L.E.

www.callapple.org



The CRPG Book Project: Role-Playing Game History From 1975 - 2015

This month, Felipe Pepe announced the availability of his long-awaited *CRPG Book*. Felipe, a native of Brazil who resides in Tokyo, has been hard at work between jobs putting together reviews of over 400 games and RPG systems from the 1975 to 2015 time frame. The result is an almost 600 page book which not only delves into the realm of the games but also how they affected the overall industry.

The project, Which Pepe started in 2014, was derived in idea from the RPG Codex's Top 70 list. The list focused on what fans decided were the top RPGs and ran several reviews from several of the users.

Pepe has taken that idea a step further with his book by including game reviews solely written by the fans of the games, developers, journalists and a host of other industry people. Some of the highlights come from Chris Avellone, Ian Frazier, Scorpia, Ferhegón, Richard Cobbett, Brian 'Psychochild' Green, Durante, George Weidman and Tim Cain, and over 100 other volunteers. This book was truly a community project with Pepe acting more as the master of ceremonies and the glue that brought it all together.

If you haven't seen the book yet, a PDF of the book as well as the ePub is available for free download. Pepe has also announced that there will be a physical printing of the book which he intends to turn into some type of community charity fundraiser sometime here soon.


True to the spirit of the project, he has also made a RAR archive of the project's InDesign files available under the Creative Commons CC BY NC 4.0 licensing. This project is massive and if you look at downloading it, the file reaches a whopping 1.3 GB.

For those of you who are not native English speakers, you will be happy to know that there are now also community efforts to translate the book into four other languages: Spanish, German, Russian, and Chinese. For more information about the CRPG Book Project, visit the website: <https://crpgbook.wordpress.com>



Wasteland

Interplay, 1988
DOS, Apple II, C64 and Windows*



I almost passed on *Wasteland* on the shelf of EB Games way back when. Like, way back when. I had tried almost every other CRPG in the store, from the big companies like Interplay,SSI, Origin – checked out their games from *Wizard's Crown*, *Bard's Tale*, *Ultima*, *Eternal Dagger*, *Might and Magic*... until *Wasteland* was the only thing left in the store.

Yet I didn't want to get it. It looked weird. Finally, two things lured me in: the *Bard's Tale* character layout screenshot on the back cover, and the *Interplay* name. I loved *Bard's Tale*, I trusted *Interplay*, and I trusted Brian Fargo. And when I sat down and plugged in this spiritual ancestor to *Fallout* into my Commodore 64, I could not stop exploring this unique, highly-imaginative world devastated by nuclear war.

I upheld Desert Ranger justice, came sneezing in, cloned my party members (!), repaired toasters, fired howitzers, got wasteland herpes from a three-legged hooker, and fought a menagerie of enemies from killer robots, giant garden pests and leather jerks to rad angels that glowed with a life of their own.

Heavily derived from tabletop RPGs, *Wasteland* features seven attributes and over 30 skills, but not all of them are equally useful.

NAME	AC	AM	MAX	CON	WEAPON
Angelika Deth	18	6	37	24	M1911A1
Snake Vargas	6	30	31	OP912	9
Corba Gato	6	30	31	OP912	9

At the end... I didn't want it to end (you can keep playing, too). I was floored. I didn't realize CRPGs could be this way. I still refer to *Wasteland*'s mechanics in game design, a brilliant blend of area design context and RPG systems used to create amazing scenarios.

Wasteland has numerous strengths and weaknesses, but the strengths definitely overshadow the weaknesses. The area design, ambiance, the system spread and applications, and the narrative itself were top-notch, while the system balance, attribute use, healing and the rare application of the ability to divide your party diminished the experience somewhat.

The narrative shines through in the game content itself, and also in the well-written (and amusingly so) narrative book included in the game, filled with richly described characters. The wasteland is simply an amazing blend of raider-occupied towns, mutant agricultural centers, robot factories, Las Vegas and even the inside of an android's brain, where I almost feared the game had jumped the shark.

The quests and encounters there are innovative and interesting, and although the overall quest doesn't kick into full gear until over halfway through the game, there's plenty to keep you going. The people of the world respond to your actions, even as soon as the first area of the campaign, and remind you of the harsh world that you've found yourself in.

Wasteland comes with a slight learning curve not present in other RPGs at the time, reflected first in its character creation. Loosely based on the *Mercenaries*, *Spies and Private Eyes* tabletop RPG, its skill-based and attribute-based system was a bit more complex than say, *Bard's Tale*, but allowed for a richer character role-playing. If I wanted to do a Russian explosives expert who liked to throw knives, I could. And that was a much richer development tree than "Fighter."

"I think the things that drew people to *Wasteland* and *Fallout* are the similarities. [...] There was this open sandbox world and we weren't preaching to you as to how to behave, in terms of a morality perspective. The 'correct' thing to do was never clear, and sometimes, there weren't clear, correct things. There was also a lot of cause and effect and a lot of subtlety; layers and layers of gameplay in a post-apocalyptic world, with an interesting combat system."

- Brian Fargo, *Wasteland's Director*

The system design is elegant, difficult and confusing at the same time. The elegance comes in the simple mechanic of being able to select any attribute, item, or skill, and then select an object in the environment for that to act on. An adventure game mechanic taken to the extreme with brilliant results. If you want to use Intelligence on an object, you can. If you want to use your proton axe on a wall or door, you can.

It touches like this where *Wasteland* shines. Similarly, the fact the skill tree grows beyond what's in the manual added a powerful element of mystery, driving you to explore more of the world and see what's in the next library, making the world deeper.

That said, *Wasteland* has its share of design confusion. It's difficult to see the differences in combat between Pugilism, Melee Weapons and Brawling. Some skills are largely useless, while others are critical (Doctor, for example). The same is true for stats: Some attributes, such as Charisma, hold little value.

Wasteland also had an annoying auto-save function that could sometimes trap you in dead-end situations (some area designs can push you out of an area, say, by falling into a river and irradiating everyone, then saves the game right after, almost guaranteeing a slow death). This often forced me to quickly yank the disk or, when I was older, set up copies of the game to prevent being trapped with no hope of salvation.

Wasteland is one of the best role-playing games I've ever played, and it's echoed in the design philosophy and how they accomplish so much by exposing their systems to design. That, matched with the sheer creative brilliance of the levels and the novelty of the setting, has kept it in my heart for over 20 years. Scorpions, androids, bloodthirsty rabbits, and all.

I swore that if I ever had the chance, I'd work on a sequel, and thanks to Brian Fargo, I got the opportunity with *Wasteland 2* (2014). I hope the next generation enjoys the wasteland as much as I did. MCA

Due to memory limitations, most of the game's text is in the printed manual. The game then asks you to read certain paragraphs. To stop players from reading them early, fake ones were added, such as false codes or an entire storyline about a Martian invasion.

It looks like a tank with an armored warrior. 1 Scorpion appears at 92 feet.

NAME	AC	AM	MAX	CON	WEAPON
1. Alan Chen	18	6	37	24	M1911A1
2. Kiyomi Apai	18	6	37	24	M1911A1
3. Karen Becker	18	6	37	24	M1911A1
4. Anna Lowhart	18	6	37	24	M1911A1
5. CHRISTINA	18	6	37	24	M1911A1
6. ACE	18	6	37	24	M1911A1

It is very warm. It is very warm. The ground seems to glow here.

Roger Wagner to Keynote KansasFest 2018



couple of years, the event transformed into a conference for Apple II developers and users alike.

This year, KansasFest welcomes back one of its alumni, Roger Wagner. Though 2018 will mark Wagner's third time as a keynote speaker for the event, most current attendees haven't had the opportunity to hear him speak because his last appearance was 23 years ago. He has deep roots in the Apple II community, back to the beginnings of the platform. In 1978, Wagner started his own software publishing company, Southwestern Data Systems (SDS), as a vehicle for some of his first software products for the Apple II, Programmer's Utility Pack and Apple-Doc, sold on cassette. He

The 2018 rendition of KansasFest is sure to be another one for the record books. This year, an early alumnus of the event has been chosen to give the keynote speech. Roger Wager, publishing giant and programmer from the 1980s has returned once again to the arena that made him infamous.

When asked about his upcoming appearance at KansasFest, Roger told A.P.P.L.E., "KansasFest was the real center of Apple II culture back then, and it had a great influence on my work, including eventually a HyperStudio conference that was called HyperFest. This will be a very meaningful homecoming for me!"

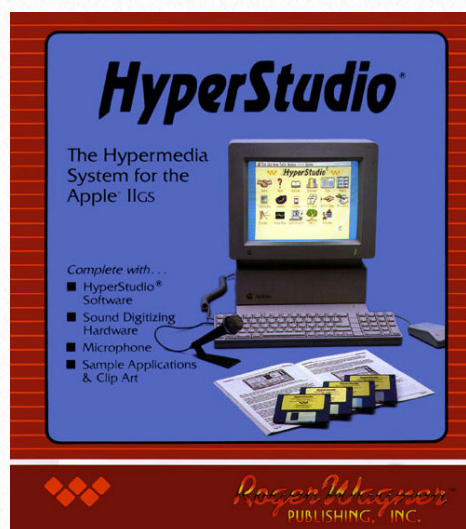
According to the press release from the KansasFest committee, "KansasFest 2018, the premier annual Apple II convention, is scheduled for July 17 – 22 in Kansas City, Missouri. This year marks the 30th time that this event has been held. In July 1989, Resource Central held the first A2-Central Developer Conference, focused on the individuals and companies who were still producing hardware and software for the Apple II and IIGS computers. Within a

also wrote a word processor for the Apple II, The Correspondent. SDS sold software written by other authors, including Glen Bredon's popular Merlin assembler, The Routine Machine by Peter Meyer (Applesoft extensions), and ASCII Express and Z-Term by Bill Blue, as well as games such as BEZARE by John Beznard and NORAD.

During the years he also wrote articles for the major publications of the day, including Call-A.P.P.L.E., Nibble, inCider, A+ Magazine, and GS+. Wagner is best remembered for his long-running Assembly Lines column in Softalk, teaching that first generation of Apple II users how to write software in 6502 assembly language.

He later renamed his software company to Roger Wagner Publishing, and continued to provide quality software for both the 8-bit Apple II and the 16-bit Apple IIGS. His most famous contribution to the IIGS was the HyperCard-inspired program, HyperStudio, which linked pictures, audio media, and text with clickable links, a foretaste of the hyperlinked web that was to later arrive in the 1990s. With his original background in teaching, Wagner

continued the HyperStudio legacy by later developing it for Windows and Macintosh computers, and focusing on its application in the school environment, to teach students to create presentations and to learn about computers. He further developed HyperDuino, an Arduino-based hardware extension for the HyperStudio to allow students to control real-world devices with their projects.”



For those of us who grew up with the Apple II in the 1980s, Roger’s software was a mainstay, between *Merlin*, *MouseWrite*, and of course, the unmistakable *HyperStudio* for the Apple IIGS, his hands were everywhere in the Apple world.

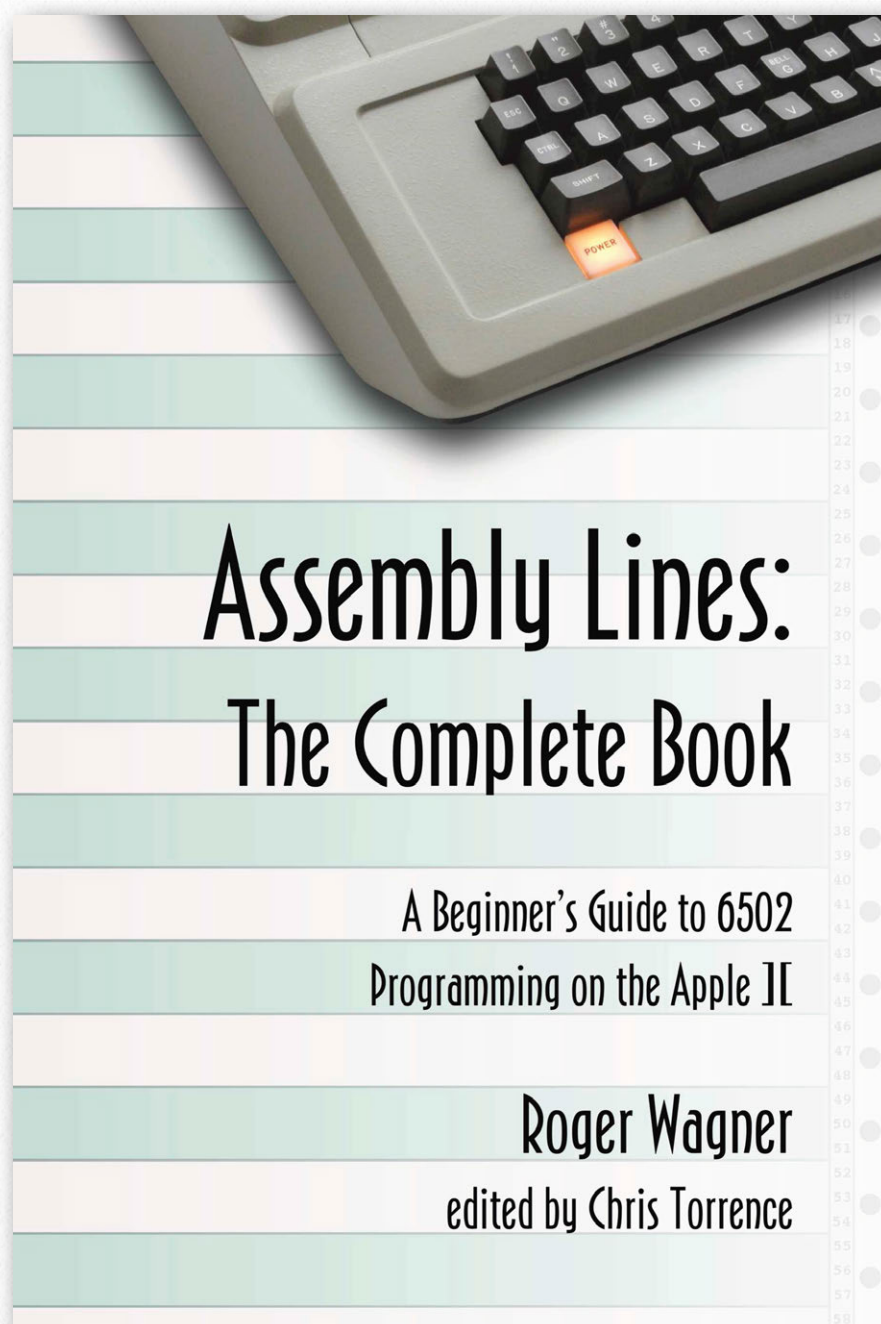
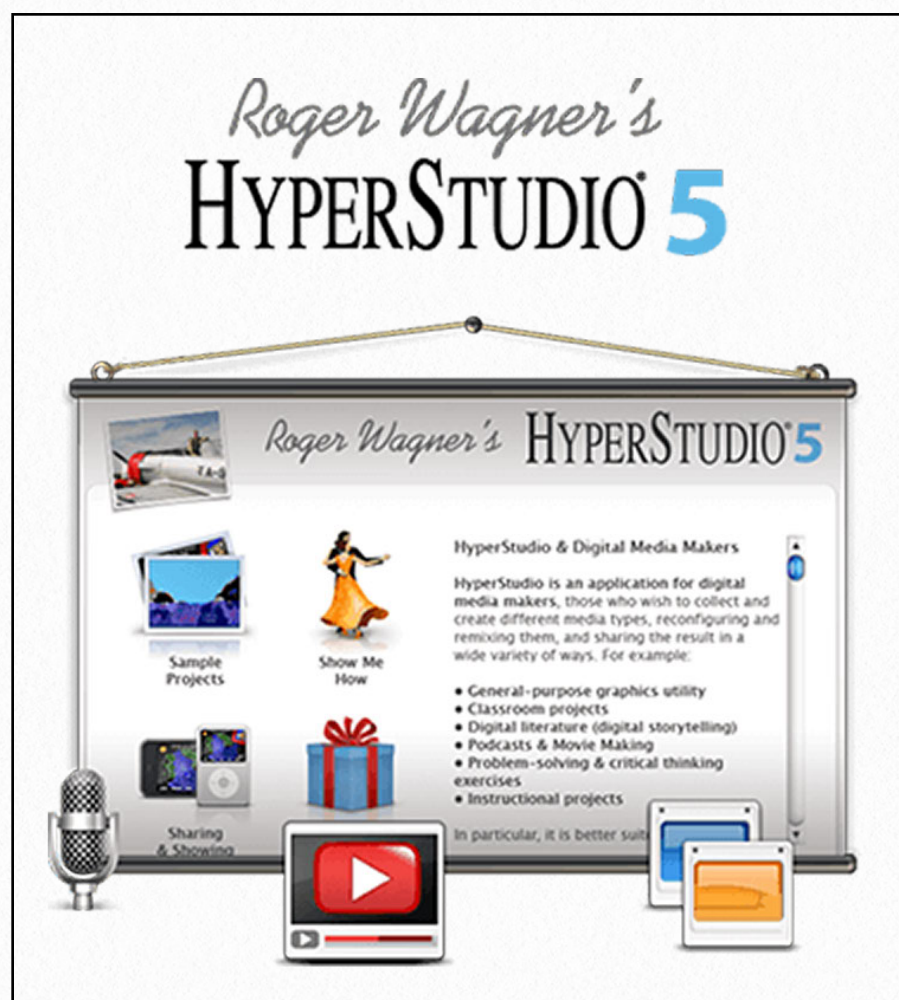
Roger could have quit in the 1980s, but instead he continues to educate the younger generation, developing Arduino-type projects. *HyperDuino* is a project which Roger uses to teach young people about the merits of engineering.

Additionally, *HyperStudio 5* for Mac has evolved from its Apple II roots. He has also released *HyperStudio AUTHOR*, that allows for the creation of simple and beautiful interactive iPad books with HTML5-based media that can be merged into Apple's *iBooks Author*.

While the 2017 rendition of KansasFest limited the attendance to just 100 people, there will be no such limitation this year according to KansasFest Committee member, Sean Fahey. He does note that, “We encourage people to double up though as single rooms may become scarce in Corcoran.”

This year, Rockhurst University has also agreed to work with the KansasFest committee to accommodate the overflow with both accommodations and also venue halls for any additional attendees that need the additional space. This is a positive change for the event which sold out completely last year and left a number of people not being able to attend. That will not be the case this year.

You can read more about Roger Wagner, HyperStudio, and get the latest news on his HyperDuino project at: <http://rogerwagner.com>



Are you upset?

Angered by the way the GS has been treated lately? Sick of your friends bragging that they have the better computer? Does seeing After Dark on the IBM or Mac make you wonder "Why can't my GS do that?" Do you like dazzling effects that will protect your valuable monitor from becoming useless when the same image is left onscreen for so long that it burned into the glass? Are you tired of dull screen savers that slow you down and interfere with your work, from companies that don't want to upgrade their products? *If so, read on!*

Twilight II v1.1 Feature Summary

- **Effects!** Nothing else can match the variety inclded with Twilight II! We've got *over forty* different colorful and stunning screen saver effect modules included, such as Clocks, Color by Color, Cyclone, Dissolve, Drip Drop, Fading Clock, Fazer, Fireworks, Headlines, Impulse 3-D, Inverter, Kaleidoscope, L.E.D. Message, Life, Meltdown, MiniFireworks, Modern Art, Moiré, Mountains, Movie Theater, Perspective, Phantom, Plasma, Puzzling, Quotes, Scanner, Scroll, Sharks and Fish, Short Out, Snow, Spirographics, Static, String Art, Strobe, Tiler, TunnelVision II, Twilight, Universe, Worms, and YouDrawIt! (which allows you to draw the individual frames that Twilight II will animate!) And more are in the works! What good is a screen saver that comes with only a few meager effects? • **Prevent Phosphor Burn In**, a permanent and real condition that happens when the same image is left on your monitor for too long! • **Protection!** Twilight II supports and protects all your favorite desktop programs (e.g. AppleWorks GS, etc.), text mode-based programs (e.g. ProTerm, AppleWorks, America Online, etc.), and even other programs such as Publish-It! • **Minimal Overhead!** Twilight II *won't* slow down you or the way your computer operates. • **Ease of Use!** Our

interface is the result of extensive testing and user feedback. • **Phantom!** Via the included Phantom module, you can run all Phantasm modules for compatibility. • **Power!** Nothing else for the Apple IIGS can match Twilight II's screen saving capabilities, over two years in the making! Our effect module format is more flexible than the competition's, and even contains support for some features not found in other screen savers for any computer! A more powerful and versatile module format means better effects! And we will publicize our format and offer tech support to anyone writing their own effect modules. • **AppleShare® Aware!** AppleTalk networks are fully supported! • **Easy Installation!** Just point and click, using Apple's Installer™ program. • **Compatibility!** Twilight II is fully compatible with all hardware and virtually all software for the GS! Twilight II works great with The Manager and Switch-It, as well as with RamFAST SCSI cards! • **Comprehensive Manual** details all aspects of use. • **Intelligence!** Twilight II won't interrupt important operations, such as file copying, printing, and file transfers! • **System 6** fully supported and required.

Twilight II™

The screen saver with a future!

<https://digisoft.callapple.org>

DigiSoft Innovations

P.O. Box 380
Trumbull, CT 06611
Phone 203.375.0837

We're a small group of die hard Apple II enthusiasts who love expanding the "limits" of the GS. We strive to create high quality products at modest prices. We are very committed to supporting and updating our products in the long run. Twilight II v1.1 is our latest example of this, and it certainly won't be our last! Many more treats are in store (such as even more effects!) for the next version of Twilight II! We're not content to sit back and watch our product sell, issuing only bug fix updates or additional effect packages. We listen to user's suggestions, feedback, and requests for features, and incorporate a great deal of these into future versions. You're buying a product with a future—a future that you can have some say in. We're committed to you!

Cross Chase: A Massively 8-Bit Multi-System Game

by *Fabrizio Caruso*

CROSS CHASE is my personal project with a main purpose and objective to create a simple, yet fun, game for literally *all* 8-bit computers/consoles/handhelds from the 1970s and 1980s. By “all” I mean any 8-bit computers with enough RAM and for which a capable ANSI C cross-compiler exists.

The project is open source and it can be followed at: <https://github.com/Fabrizio-Caruso/CROSS-CHASE> where both the source code and pre-compiled binaries for more than 60 different systems and configurations are freely available. Please go back often to the GitHub page if you want the latest version because I make frequent updates.

HISTORY

This idea came to me when I found out about the CC65 cross-development toolkit for 6502-based systems: <https://github.com/cc65/cc65>.

I realized that CC65 provided a common language (ANSI C) for all its supported targets and some common APIs for most of its targets. I started playing around with it by modifying the “hello world” example, which is provided with it. After 1400 Git commits from the “hello world” I have now an arcade-like game that can run on about 60 different systems.

CC65 was clearly not enough and I had to resort to other toolkits such as: Z88DK at: <https://github.com/z88dk/z88dk> which comes with two ANSI C cross compilers (ZSDCC and SCCZ80) for Z80-based systems and WinCMOC and CMOC proper at: <https://perso.b2b2c.ca/~sarrazip/dev/cmoc.html> which are development tools for 6809-based computers.

I also plan to use GCC for TI for the TMS9000-based Texas Ti-99/4A, which I intend to support even if it is a 16-bit system but for the same 8-bit era.

THE GAME

The idea of the game is my original idea. You are chased by some bad guys. You can kill the bad guys by luring them into some mines that they do not see. While I believed my idea was original and I was not particularly inspired by anything or any other program, I later found out that similar games do exist. The closest one may be Robots (<https://wiki.gnome.org/Apps/Robots>) as included in Gnome.

This did not deter me and I pushed on with my project. The main difference between my game and Robots is that my game is an action game and Robots is turn-based. With respect to the gameplay the game has three versions in order to fit into as many different memory (and in some cases video) configurations as possible.

The VERY SAME code (or meta-code) produces all three versions:

- TINY: just you, the bad guys chasing you and the mines;
- LIGHT: same as TINY but: 1. there is an enemy (the skull) who does not die on the mines and 2. you can get some power-ups including a gun that can kill all including the skull;
- FULL: as as LIGHT but with different enemy types, more power-ups and multiple levels with different walls and missiles.

THE FRAMEWORK

A very important by-product of this project is a C library/framework that I have written to create an abstraction layer so that the very same code can be used for all systems.

This means that it is already possible to create other universal 8 bit games within the graphical and sound limitations of the framework.

THE APPLE VERSIONS

The game supports quite a number of systems (see next section) including the Apple][and Apple //e systems.

Currently, the Apple versions currently only use the text mode and have no sound. Both sound (through the bit banging technique), as well as graphics (through the CC65's TGI libraries) are planned. Nevertheless the game is already fully playable in these versions.



THE SUPPORTED SYSTEMS

The supported systems are in principle all 8-bit systems but the game currently only works for just about 70 different systems and configurations.

I have been actively adding new systems, with the list of supported systems changing all the time. Thus I am only providing a partial list of the supported systems in each category:

Partial List of FULL Versions:

- Sega SC 3000
- Luxor ABC80 32k
- Jupiter Ace 16k
- Apple //c
- Apple][e
- Mattel Aquarius 16k
- Atari 5200 (console)

- Atari 400/800 (color low resolution)
- Atari 400/800 (high resolution)
- Tangerine Atmos and Oric 1 48K
- Commodore 128 (native 40 column mode)
- Commodore 128 (native 80 column mode)
- Commodore 16/116/+4 (32k min)
- Commodore 64
- Commodore CBM 510
- Commodore CBM 610
- CoCo 1/2/3 and Dragon 32/64 (multiple versions)
- Amstrad CPC
- Galaksija 22k
- Gamate (console)
- Lambda 8300 16k
- CCE MC-1000 48k
- MicroBee
- MSX 32K (cassette and rom version)
- MTX
- Nascom computer series 32k
- NES (console)
- Ohio Scientific 1P 32k
- Philips P2000
- PC-6001 32K
- Commodore PET 16k
- Sam Coupe
- Sharp MZ series
- Sinclair Spectrum 48K
- Spectravideo SVI 328
- VG-5000 with 16k expansion
- Vic 20 with 16k expansion
- VZ 200 family (Vtech Laser 200/310 & VZ 200/300) with 32K
- Robotron Z 9001 32k
- Sinclair ZX80 with 16k expansion
- Sinclair ZX81 with 16k expansion



SAME CODE FOR MORE THAN 100 SYSTEMS

I am using the very same code for all systems. This is possible because:

I am using ANSI C, which is a universal language, which is compiled by multiple cross-compilers into executables for the specific systems.

I have created a universal 8-bit framework for very simple graphics and sounds. The framework provides an abstraction layer so that hardware-specific code is used for graphics and sounds.



ACKNOWLEDGEMENTS

This is a personal project but I have been helped and supported by different people.

I have had a lot of support and help from Stefano Bodrato from the Z88DK team. I have also been supported by some people from the "scene" (Simon Jonassen from the CoCo/Dragon scene) and from some of the authors of the other toolkits (Christian Groessler from the CC65 team and Pierre Sarrazin who is the author of the CMOC).

ABOUT THE AUTHOR

Fabrizio Caruso is a software engineer and a retro-computing enthusiast and collector, who has amassed a collection of about 80 computers from the late '70s, '80s and early '90. His primary computer interest is those machines which are from the 8-bit era or computing.

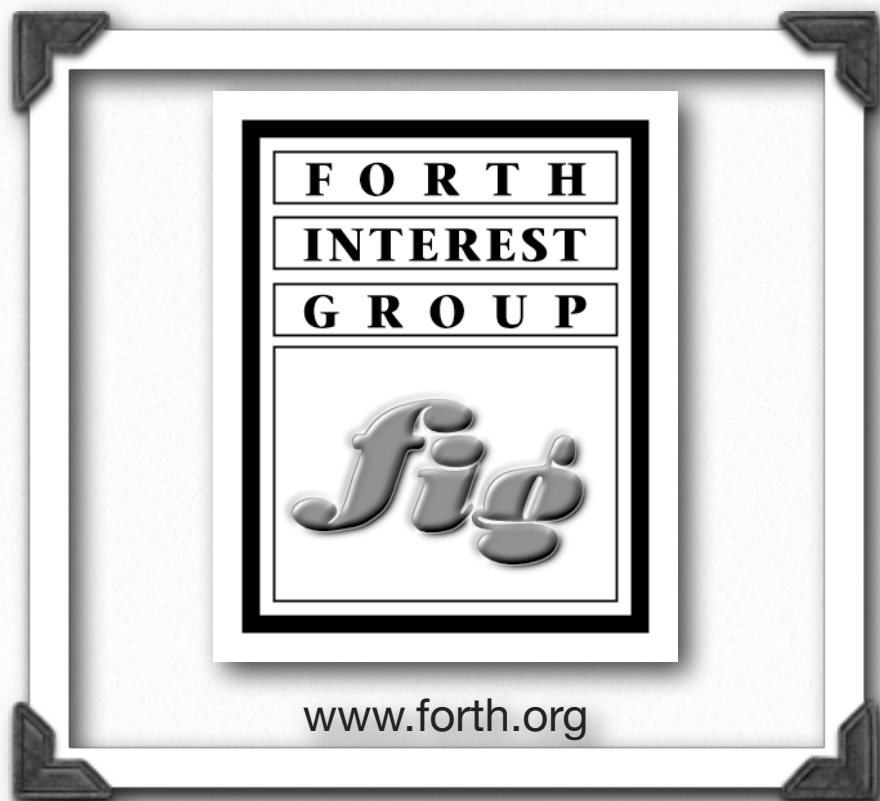


Partial List of LIGHT Versions:

- Atari Lynx (handheld)
- Luxor ABC80 16k
- Atari 400/800 (high resolution)
- Commodore 16/116 (unexpanded)
- CCE MC-1000 16k (unexpanded)
- MSX 16k
- Nascom computer series 16k
- Oric 1 16k (unexpanded)
- Philips P2000 16k (unexpanded)
- Spectravideo 318 16k (unexpanded)
- VG-5000 16k (unexpanded)
- Vic 20 with 8k expansion
- VZ 200 family (Vtech Laser 200/310, VZ 200/300) with 16K
- Robotron Z 9001 16k

Partial List of TINY Versions:

- Mattel Aquarius with 4k expansion
- PCEngine (console) 8k rom version
- Creativision (computer/console hybrid) 8k rom version
- Ohio Scientific 1P 8k
- Commodore PET 8k
- Sinclair Spectrum 16k (unexpanded)
- Commodore Vic 20 with 3k expansion



Huibert-Aalbers Software

Jigsaw!

A very simple game, written in less than a week. It was never meant to be published but my editors at Britannica liked it and convinced me. Jigsaw! sold over 100,000 copies on multiple platforms.

LaserForce

A 3D high speed action game for the Apple II gs with sound using the Ensoniq chip.

SoundSmith

This is probably the most rewarding program I have written to date. SoundSmith became an instant hit. People had purchased their Apple IIGS to enjoy their Ensoniq chip and there wasn't a decent music application. SoundSmith filled that gap.

AZERTY

A simple NDA that allowed French IIGS users to fully use their keyboard when working with GS/OS applications.

Jigsaw Deluxe

An improved version of my best selling game, Jigsaw! This application was lost for almost 30 years and is now finally available for download.

Get These Titles at:

www.huibert-aalbers.com/AppleIIgs

Call-A.P.P.L.E.
Apple Pugetsound Program Library Exchange

Turtlesoft
Turtle Graphics for Applesoft

Robert W. Gallup

Produced by
Brian Wiser & Bill Martens

Turtlesoft

DOS & new ProDOS version
free for Members:
www.callapple.org

Manual for Everyone
Just \$11.25
www.callapple.org/books

STRUCTRIS

for iOS – Just \$1.99

www.structris.com

A.P.P.L.E. In Depth Vol. 1:
All About Applesoft
Enhanced 40th Anniversary Edition

Call-A.P.P.L.E.TM
In Depth



Enhanced Edition

Produced by
Brian Wiser & Bill Martens

Coming Soon to the A.P.P.L.E. Book Store!

Complete with all programs available on FREE floppy disk Image

6502 Assembler Tricks: Self-modifying code based on the 3D Demo

by Dr. Marc A. Golombeck

Neukirchen/Erzgebirge, Germany



Introduction

Many programmers are of two minds about self-modifying code. Some are awed by its mere existence and considering it being the Holy Grail of assembler programming. Other users deem it to be sorcery, which should not be used in 6502 assembler programming at all.

In this article I want to give some insight in the technique of programming self-modifying 6502 assembler code. The coding examples given are taken from my current development project: a 3D-engine for the Apple II so there is a close connection to a real life project. Detailed general background information on this 3D-engine can be found in the Fall 2017 issue of *Call-A.P.P.L.E.*

In the early 3D-engine versions there was no self-modifying code at all. This issue came into play at a later time when I came across two general coding problems:

1. Algorithm execution time
2. Code size

And these two points really sum up the main reasons why one would benefit from using self-modifying code in a program: to make the algorithm faster and/or to save RAM space.

More about these benefits follow later in this article. First let's answer the most important question: "What is self-modifying code?"

The answer is quite simple: self-modifying code is code that changes itself at run time in order to fulfill a certain task. That means that the code patches itself at well-defined memory locations during execution.

Let me illustrate this concept using a very simple example 6502 assembler listing. Think about a loop where data is read from a certain memory address into the accumulator using the X-register as an index that is incremented:

```
myLOOP    LDA    DATA,X
           INX
* do something in the loop body...
*
*
           JMP    myLOOP    ; jump back to loop start
```

Simply put, this loop reads a byte from the location `DATA + X` into the accumulator. `X` is incremented and then there might be a lot of arbitrary other commands in the loop body to fulfill a certain task as well as branches out of the loop. At the end the loop is restarted with the `JMP` command and the incremented `X`-register (which we of course assume to be preserved in this example despite of the other operations performed in the body of the loop).

Now we have created a loop with increasing `X`-register. However, what if we need a loop with a decreasing `X`-register with the rest of the loop body staying the same?

We could introduce some branches with conditions and switch between an `INX` and `DEX` for incrementing or

decrementing the X-register as needed. This is possible but makes the code more complicated and always costs some additional cycles since we need to put the branching conditions inside the loop body. If you want your code to be fast you should not waste cycles for checking conditions and doing branching inside a loop!

Another possibility is to duplicate the loop code and just exchange the INX by a DEX. If you can spend the extra space in RAM for the duplicate code then you may go for that very fast solution.

However, you might have noticed that the required code change of the loop body is pretty small. To be more precise: you just need to change one byte of the code. As I have written in the paragraph above you need to exchange the INX by a DEX-opcode in order to change the X-register from counting up to counting down. What if we just change this single byte during runtime to the opcode we need?

Self-modification takes place before the loop is entered. We need a small piece of code that decides whether we should put an INX or a DEX at the required position in the loop body. So we have a small additional overhead requiring some extra cycles and RAM space but if we do not need to change too often between INX and DEX we save processor cycles in the end and get a very reasonable compromise between algorithm speed and RAM usage.

How do we realize the self-modification part in our example? First we should introduce a label (modi1) at the position we want the modification to take place:

```
myLOOP    LDA    DATA,X
modi1     INX
* do something...
*
*
        JMP    myLOOP ; jump back to loop start
```

Then we need to add some code with condition checks in order to choose which opcode we need in the loop body and last but not least do the code modification before the loop is executed. I do not want to go into details of the condition checking since this is part of the individual algorithm logic and must be designed to your individual

needs. The self-modification part is rather simple and straightforward:

```
* condition for INX true
        LDA    #$E8      ; hexadecimal value for opcode INX
        STA    modi1     ; store the opcode for INX
*...
* condition for DEX true
        LDA    #$CA      ; hexadecimal value for opcode DEX
        STA    modi1
*...
```

And we are done! This is a tiny example for self-modification as a starter but it already demonstrates how powerful self-modification can be. If you run into RAM shortages or want to make your algorithm faster by optimizing loops without using too much additional RAM this is a simple but powerful solution.

However, there are also some drawbacks when using self-modifying code. First of all the source code gets more complex and debugging of the modified code at runtime is more complicated. Secondly, you need to make sure that the mechanism, which checks the conditions and performs the self-modification, is working properly under all circumstances. If something goes wrong the self-modification can screw up your code leading to program crashes or other funny behavior during runtime.

This is particularly true if you are using self-modification techniques with relative addressing, which will be illustrated later. The above example uses direct addressing, i.e. the memory address where the new opcode is stored is defined by a label (here: modi1) with a fixed address at compilation time.

After this short introduction with a simple example I want to demonstrate this technique based on solutions that I have integrated into my 3D-Demo assembler code.

Coding Examples

Changing a line drawing routine to an undrawing routine: Drawing HIRES lines on an Apple II computer is quite an ordeal. Using the integrated HPlot-ROM-routines is possible, however, drawing speed is mediocre at most. Therefore I decided to implement a faster line drawing algorithm in my 3D-Demo.

I was pointed to Andy McFadden's fdraw high-speed drawing routines and I deduced my FASTLINE-implementation from his library. Before we take a closer look on an interesting self-modification part of this routine I want to give a short overview on how HIRES pixel are set.

HIRES screen data is not organized in a coherent way on the Apple II. As you might recollect each HIRES line with 280 pixels is stored in 40 consecutive bytes of memory. Each byte represents 7 pixels on the screen in x-direction (in reverse order to be more precise and different for even versus odd "columns" when considering colored pixels – but I do not want to get more into these details here!). The 8th bit of each byte encodes the color palette.

To make it even more complicated consecutive HIRES lines in Y-direction are not stored sequentially in memory so you need to implement a function that gives you the base address of the first byte of the desired HIRES line number. After determining the base address you need to figure out in which of the 40 bytes the pixel information has to be stored. Since every byte holds 7 pixels you need to divide the X-position of the pixel by 7 in order to get the position of the corresponding byte in the row of 40 bytes and use the remainder of the division for determining the pixel position inside the byte. A fast way to perform these tasks is using lookup tables and a pixel mask.

So basically the following steps have to be performed to plot a pixel at (x,y):

- Calculate the base address of the desired line number
- Divide the pixel number by 7 to determine the corresponding byte of the line and load that byte
- Calculate the remainder to determine the pixel position inside the byte
- Load the corresponding pixel mask
- Do an OR-operation of the corresponding screen byte with the pixel mask this preserves the current screen content and sets an additional pixel.
- Save the modified byte back to the corresponding HIRES screen memory location.

In 6502 assembler this can be represented for example as:

```
* Get base address of HIRES line
LDX  YPOSN      ; line number
LDA  YADRLO,X   ; get low address byte from table
STA  YBASE      ; store low-byte in zero page
LDA  YADRHI,X   ; get high byte
STA  YBASE+1    ; store high-byte

*
* Get byte number & pixel position for desired x-value
LDX  XPOSN      ; assume x-position <= 255 here
LDY  DIV7,X     ; get byte number from div-by-7 table
LDA  MOD7,X     ; get pixel number from remainder
TAX           ; move accumulator to X-register
LDA  PIXMASK,X  ; load pixel-mask
STA  PIXEL      ; temporarily save pixel-mask

*
* Get desired HIRES-byte and set new pixel
LDA  (YBASE),Y  ; load HIRES byte
ORA  PIXEL      ; set new pixel preserving content
STA  (YBASE),Y  ; store byte back to HIRES memory
```

Basically this is the way that a pixel is drawn in the 3D-Demo. Please note that this draw routine does not consider a color mask and hence only is a monochrome algorithm. Nevertheless this is sufficient for e.g. black & white wireframe models. If we would want to consider colored pixels we would need to pay more attention on how we set the pixel-mask.

Some more remarks on certain steps in the algorithm:

- The base address YBASE is a two-byte pointer somewhere in the zero page. I use indirect-indexed addressing for retrieving the appropriate byte from HIRES memory. The two byte zero page pointer holds the memory base address for the desired line number. This value is loaded via a fast double table lookup.
- The x-position of the pixel is assumed to be 255 or less here for the sake of simplicity in this example. In the 3D-Demo the algorithm is capable of drawing to all 280 possible x-positions.
- As stated above 7 pixels are stored per byte so a division by 7 gives the byte number which has to be changed in order to draw a pixel. The determination of the byte number is again done using a lookup table. The corresponding byte number is stored in the Y-register for further use as an index offset of the base address YBASE.
- The pixel position from 1 to 7 from in the corresponding byte is given by the remainder of the division by 7 and is

also provided by a fast table lookup. This value is then moved to the X-register in order to be used as an index for loading the corresponding pixel mask (PIXMASK).

Examples:

Remainder = 1 set pixel no. 1 PIXMASK = \$01 = % 0000 0001

Remainder = 2 set pixel no. 2 PIXMASK = \$02 = % 0000 0010

Remainder = 3 set pixel no. 3 PIXMASK = \$04 = % 0000 0100

And so on...

Note: pixel positions are stored in reverse order in the HIRES memory!

- The corresponding byte for the pixel-mask is loaded and stored in a temporary zero page location.
- Setting the new pixel is rather straightforward after the slightly complicated way in retrieving the correct pixel mask. First the HIRES byte is read using indirect-indexed addressing. Then the magic is done using the ORA-opcode which performs a logical OR of the HIRES byte with the pixel mask. Spirit and purpose of this operation is to preserve the already stored pixels in the byte and only adding the new pixel. The logical OR preserves all bits that are already set to 1 in the HIRES byte. If we would use a logical AND instead we would delete all other bits in the HIRES byte. This leads to unwanted graphical effects if there are more than one line of the wireframe model intersecting the HIRES byte.

This was a very short side trip into Apple II HIRES screen programming with many simplifications but no word about self-modifying code yet. I am sorry about taking this detour but I needed to prepare what I am talking about in the next paragraphs.

The routine for setting pixels given above is pretty fast but it has a major drawback: it can only set new pixels! What if one wants to delete a certain pixel in a HIRES byte? Storing a zero in memory would delete all 7 pixels, which would do for a general screen erase (see below) but what can we do to only delete one pixel and leave the others unaltered?

Yes, the answer is we need to store a zero at the corresponding bit position and leave the other bits unchanged. This could be done by using a logical AND-operation with a special delete pixel mask where the bit to change is set to 0 and all the other bits are set to 1, for example:

- Delete pixel 1 DELMASK = \$FE = % 1111 1110
- Delete pixel 2 DELMASK = \$FD = % 1111 1101
- And so on

We would need to do an exclusive-or of the PIXMASK with \$FF in order to get the delete mask and exchange the logical OR opcode (ORA) with a logical AND-operation.

We could copy the FASTLINE-routine and implement the changes to create a FASTDELETE-routine or insert conditions and branches inside the FASTLINE-routine in order to switch logical operations when we want to delete a line. The first solution is fast but needs a relevant amount of extra RAM, the second approach slows the line drawing speed down significantly. Hence, this is a good example for introducing some self-modifying code.

We are adding code which patches the appropriate opcode-location changing ORA to AND and vice-versa. Furthermore we need to patch the PIXMASK to become the DELMASK. Since line deleting and drawing can be organized to be strictly sequential for each animation frame we only need to change the PIXMASK twice per frame (first to DELMASK then back to PIXMASK).

First we add a label in front of the ORA-opcode and then we add the code which changes the ORA to AND and back again:

```
* Get desired HIRES-byte and set new pixel
      LDA (YBASE),Y ; load HIRES byte
modiORA ORA PIXEL ; set new pixel preserving content
      STA (YBASE),Y ; store byte back to HIRES
                        ; memory
*
...
* change ORA to AND
      LDA #$25      ; opcode for AND zero page
                        ; addressing
      STA modiORA   ; modify ORA to AND
```



```

...
* change AND to ORA
    LDA  #$05 ; opcode for ORA zero page
            ; addressing
    STA  modiORA ; modify AND to ORA

```

This is very short and simple but highly efficient! You just need to distinguish between lines deleted and drawn in an animation frame and patch the appropriate opcode in the FASTLINE routine and change the PIXMASK to the DELMASK.

You might claim that deleting an area by just storing zeros into appropriate memory locations is faster. Yes, that is true, but there are use cases where you just need a single line in a wireframe model to disappear without the need to redraw all the other lines. In this case you need a FASTDELETE option!



Considerations About Data Management

After the detailed description of the HIRES line drawing and undrawing algorithm I want to discuss some niceties regarding data management. When it comes to displaying several 3D-objects in an animation frame one needs a method for a fast data access and management in order to load and store 3D-object data.

The 3D-object data is loaded at different locations in memory and needs to be accessed every time an animation frame is drawn. The general solution accessing array-like data in RAM is by using a zero page pointer that is updated to the new data location for the next 3D-object to be drawn.

Let pDATA be the pointer to the storage location of the 3D-object then data access is usually realized by using indirect-indexed addressing, e.g. like:

```
LDA  (pDATA),Y
```

This solution is very handy but has a small drawback. To be honest for most applications this is not really an important drawback but if you are short of CPU cycles and need to squeeze out every microsecond of processor time you should think about the cycle count needed for an opcode.

The given LDA with indirect-indexing addressing needs at least 5 cycles (and 6 cycles if memory page boundaries are crossed). If we choose LDA with absolute indexed addressing we can save one cycle! Almost the same is valid for a STA-command: 6 cycles if we use indirect-indexed addressing and 5 cycles if we use absolute indexed addressing.

This sounds very nitpicking but if you consider that during an animation frame a lot of LDA- and STA-commands are necessary the change of addressing mode to save cycles can be useful.

So we should preferably write LDA DATA,Y in order to access the 3D-object data but we still need to find a solution on how to set DATA to the correct address. You got it right – we can again use self-modifying code to accomplish this task!

A first step would be to label the respective line of code in order to make it accessible for self-modification. However, this time we do not need to change the opcode we need to change the address i.e. the value of DATA to the desired memory location. Hence we could write the following lines of code:

```

*
modiDATA LDA  DATA,Y
*
...
*
* change high byte of address of DATA
*
    LDA  newADDRESS
    STA  modiDATA+2
*

```




As you may have noticed a value of 2 is added to the label modiDATA. This is done to patch the location where the high byte of the address of DATA is stored. If you also want to change the low byte of DATA you need to add only a value of 1 to the label (remember byte order is little endian!):

```
STA modiDATA+1
```

If you do not add a value to the label modiDATA then the opcode (in this example LDA) will be changed unintended and lead to unexpected results of the program at runtime. This is a bug that is often introduced when trying to fiddle around with address bytes!

If we assume that we use absolute addressing with LDA newADDRESS and an absolute STA we need 2 + 4 additional cycles for the self-modification. This would pay off if we issue at least seven consecutive modified LDA- or STA-commands e.g. in a loop for saving cycles.

If we stick to the truth we need to consider also the necessary branching logic to handle the switching between different objects before the self-modification takes place. This needs also to be considered in the cycle balance of the algorithm.

This approach might seem to be inefficient at a first glance but it really makes sense to take a closer look onto addressing modes and to balance out the two addressing methods described above.

Optimized HIRES-screen Erase

One of the most time-consuming algorithm parts of the 3D-Demo with direct impact on the frame rate is the HIRES screen erase. The basic idea is to erase the old animation frame and draw the new wireframe or raster-filled 3D-objects. The first approach was to use the HIRES screen erase which is implemented in the Apple][ROM. This, however, takes about 240ms for a screen erase with HCOLOR=0 (black1), which is far too slow.

Basically, the ROM routine is a nested loop making this piece of code very compact. One solution in order to gain more execution speed is to unroll parts of the nested loops and limit the part of the screen that needs to be erased to certain boundaries.

To clear a screen to black it is necessary to store the value zero (\$00) into specific HIRES RAM locations. As we have already learned that a HIRES line consists of 40 consecutive bytes in memory we just need to know the base address of the desired line and store zeros in consecutive bytes beginning with the base address.

The base address of a line can be retrieved via a simple table lookup operation. Since the FASTLINE algorithm already uses a lookup table for the base addresses we can easily reuse it for this purpose.

However, if we still want to be faster and we are willing to spend some extra RAM storage we can improve our algorithm by explicitly including the line base addresses with STA-commands in the source code.



This means that we unroll the loop in the y-direction hence omitting the table lookup for the line base address and only loop over the x-coordinate. The following source code shows parts of the large amount of necessary STA-commands, which I refer to as STA-slide in the following paragraphs. The arrangement of the address byte values of the single STA-commands corresponds to the sequential order of HIRES lines on the screen, i.e. address \$2200 corresponds to line 32 on HIRES page 1, address \$2600 to line 33, address \$2A00 to line 34 and so on.

```
* loop over X-coordinate and store zero in HIRES RAM
    LDA    #$00
    STA    $2200,X
    STA    $2600,X
    STA    $2A00,X
    STA    $2E00,X
    STA    $3200,X
    STA    $3600,X
    STA    $3A00,X
    STA    $3E00,X
...

```

This routine is pretty fast. A screen erase only takes about 20 – 40ms depending on the size of the HIRES screen part that is going to be erased. In the 3D-Demo the active drawing area is about 200 x 128 pixels large and earlier versions of the demo always deleted the whole active drawing area in every frame. I was wondering if this brute force erase is always necessary especially if objects are drawn that are relatively small so most of the screen gets erased for no purpose. How can the screen erase become even faster?

A solution came to my mind when I was coding the 3D-object bounding box algorithm, which calculates the screen dimensions of each object (XMIN, XMAX; YMIN, YMAX). Why not using the bounding box information for achieving a faster screen erase? The basic idea was to erase only the bounding box area of each object on the screen, which is in most use cases significantly smaller than the total active drawing area.

Evaluating the bounding box information could be done by two nested loops with a table lookup for the line base address. However, I wanted to keep my fast STA-slide solution since nested loops would be relatively inefficient if the bounding boxes reach a certain size and hence there wouldn't be any advantage anymore in using the bounding box erase approach.

This again is the point where self-modification is playing an important role in the game! Considering the x-coordinate there is only a small change in the code necessary since the x-loop can be easily adapted to the XMIN and XMAX values of each bounding box taking XMAX as the starting value for the x-loop and decrement until XMIN is reached.

The idea for the y-coordinate is the following: what if we can modify the code in a way that we jump into the STA-slide according to the value of YMIN and jump out of the STA-slide at YMAX omitting all HIRES lines that are not part of the bounding box?

The solution is to modify the address bytes of a JMP-command to jump into the STA-slide and to insert an exit branch in the STA-slide. The easy part is to calculate the jump into the STA-slide. The proper branch out of the slide is a bit more complicated, as you will learn in the next paragraphs.

Branching In...

Branching in the STA-slide is relatively easy compared to do the branching out. The idea is to skip the line numbers that are lower than YMIN. In order to perform the correct jump forward in the STA-slide we need to calculate the appropriate relative jump size in bytes. Since one STA-command with address bytes uses three bytes we just need to multiply the number of lines we want to skip with three and add the base address of the first STA-command in the STA-slide.

The multiplication by three is done using a lookup table. The correct target address is calculated by a simple 16-bit addition of the relative jump size and the base address of the STA-slide. The generated jump address is then simply patched into the code after the JMP-command. The source code is structured as follows:

```
* calculating the relative jump size and store as YJUMP1
    LDA    YMIN
    TAX
    LDA    MULT3LO,X
    STA    YJUMP1
    LDA    MULT3HI,X
    STA    YJUMP1+1

```


YJUMP1 holds the relative jump size in bytes. This value is now added to the base address of the STA-slide and patched after the JMP-command:

```
CLC
LDA #<STASLIDE ; low-byte of STA-slide start
ADC YJUMP1
STA jmpSLIDE+1 ; modify JMP-address low-byte
LDA #>STASLIDE ; high-byte next
ADC YJUMP1+1
STA jmpSLIDE+2
...
jmpSLIDE JMP STASLIDE ; address STASLIDE gets patched
...
STASLIDE STA $2200,X ; beginning of STASLIDE
```

The code example reveals the “magic”: the JMP-command initially points to the beginning of the STA-slide but the target address is modified every time when new object bounding box parameters are available.

This approach allows for a fast skip of a desired number of HIRES lines. But what about branching out of the STA-slide? You might think that we could also simply use the JMP-address change approach as just described. Yes, that is possible in principle but it is very cumbersome.

Not only would we need to patch three consecutive bytes in the STA-slide (one STA-command with its corresponding two address bytes) but it is also necessary to undo the patch of the STA-slide and restore the former STA-command with its address bytes! This would need to store the address bytes before patching the STA-slide or do some other tricks using the line base address lookup table. We can do this in a more elegant way, as I will describe in the next paragraphs.



Branching Out...

As described above we need a good idea in order to achieve a cycle-saving solution for an early exit out of the STA-slide when we reach YMAX. The first step is to calculate the exit address where we should branch out of the slide by determining the memory address of the last STA-command (corresponding to line YMAX) to be executed. This is straightforward by calculating a jump distance and adding it to the STA-slide base address much like it has been described above about branching in the STA-slide. So at first there is no artifice yet:

* calculating the relative jump size for the slide exit

```
LDA YMAX
TAX
LDA MULT3LO,X
STA YJUMP2
LDA MULT3HI,X
STA YJUMP2+1
```

The variable YJUMP2 holds the relative jump width referring to the beginning of the STA-slide. Hence we need to add the STA-slide base address in order to receive the address where we need to put our exit code into the STA-slide. This time we store the two address bytes in the zero page so we can use it later as a pointer:

```
CLC
LDA #<STASLIDE ; low-byte of STA-slide exit
ADC YJUMP2
STA zpPOINTER ; store into zero page
LDA #>STASLIDE ; high-byte next
ADC YJUMP2+1
STA zpPOINTER+1
```

The crucial point is to patch the STA-opcode at the calculated address to an RTS (\$60) operation in order to exit the STA-slide:

```
LDA #$60
LDY #$00
STA (zpPOINTER),Y
*
* this STA-operation conforms to 6502 coding standards
* using 65C02 operations this can be written shorter
* saving
* some cycles:
* LDA #$60
* STA (zpPOINTER)
```


An RTS might be rather unexpected at this point since patching an RTS in the STA-slide would cause a return from the subroutine which performs the screen erase back to the calling routine and hence would lead to an unfinished screen erase!

In order to prevent this unwanted subroutine exit we need to do a little trick. Remember how the RTS-command works: an RTS normally follows a JSR (jump to subroutine) and performs a jump back to the location where the JSR-command is located in the code continuing execution directly after the JSR-command.

In order to perform the JSR the return address is pushed on the stack before executing the subroutine code - to be more precise this is the address of the next operation after the JSR minus 1 byte! When an RTS is encountered the two address bytes are pulled back from the stack and moved to the program counter which is then incremented by one resulting in the address of the next command executed by the processor.

If we manage to push an appropriate address for further program execution after leaving the STA-slide onto the stack, an RTS inside the STA-slide would work like a JMP command e.g. branching to code which follows at the bottom of the STA-slide! The advantage of this method is that we just need to modify one opcode in the STA-slide that is to replace one STA by an RTS and we can leave the following address bytes of the STA-command untouched. This makes it a lot easier to restore the slide for the next erase operation with modified YMIN/YMAX-values since we do not need to restore the address bytes after the STA which would have been overwritten if we inserted a JMP-instruction instead of a simple RTS!

So preparing the stack with return address bytes is rather straightforward. We just need to define an entry point in the code where we want to branch out of the slide and subtract 1 and push these bytes onto the stack:

```
LDA #>staRTS-1 ; generate pseudo-jump address
PHA           ; first high-byte then low-byte
LDA #<staRTS-1
PHA           ; low-byte
...
```

The label staRTS is defined at the bottom of the loop where the loop management is done for example:

```
* loop bottom: X-register is loop variable
staRTS      INX ; entry point from RTS in unrolled loop
...
```

When the stack is prepared in this manner an RTS will lead to a jump to the loop bottom (staRTS) and the X-register will be incremented which holds the loop variable in this example.

After we have left the STA-slide we need to tidy up a bit which means to remove the RTS-opcode from the slide and restore the original STA-command:

```
LDA #$9D      ; STA-opcode, absolute indexed
LDY #$00      ; set index-variable to zero
STA (zpPOINTER),Y ; revert the RTS-entry to STA
```

This could also be abbreviated using 65C02-opcodes without the need of using the Y-register by just writing a STA (zpPOINTER).

This example shows how self-modifying code can be used to break down the execution of a long static code to its really needed parts and save a lot of execution time by only erasing on screen what is really needed. We have seen a simple technique to branch into code by manipulating the target address of a JMP-operation and we have learned how we can easily patch an exit branch in a routine by using an RTS-operation, which in fact behaves like a JMP.

Remark: Using the zero page as storage for the pointer and the 65C02 indirect STA-opcode seems to be the fastest method to patch the RTS-command into the slide if one can afford to spend two bytes in the zero page as temporary storage. Alternatively one could use absolute STA-commands and patch the address information directly in the code which needs more cycles and write:

```
CLC
LDA #<STASLIDE ; low-byte of STA-slide exit
ADC YJUMP2
STA modiSTA    ; store into zero page
LDA #>STASLIDE ; high-byte next
ADC YJUMP2+1
STA modiSTA+1
LDA #$60
*
modiSTA STA rtsADDRESS ; address gets overwritten
```


But remember that you also need to spend some extra cycles when changing the RTS back to the STA-opcode when cleaning up after you branched out of the STA-slide if you are not using the zero page pointer method as described above!

Conclusion

This article covers some examples of using self-modifying coding techniques with 6502 assembler. The coding examples should illustrate that self-modifying strategies get handy if one wants to reduce both code size and execution time. Sometimes only small changes are necessary in order to get rid of branches and conditional logic in loops or subroutines that are called often resulting in a noticeable saving of processor cycles.

Self-modification is not vicious or evil at all. It can be of great help if you want to do things even faster or a little bit different without the need of introducing a lot of branches or almost duplicate code. However, the major drawback of this approach to me should not be concealed: debugging self-modifying code can be very exhausting – but it really recompenses if the algorithm finally works as expected!

6502 Assembler Listing

The following listing shows the main part of the fast screen erase routine with both self-modification techniques for branching in and out of the long STA-slide as described in the corresponding paragraphs of the article.

The code given in this listing erases the bounding box area of a 3D-object on HIRES page 1. For page 2 the line base addresses need to be adjusted accordingly.

```
*****
* fast bounding box erase of a *
* 3D-object on HIRES page 1 *
*****
*
bbERASE
*
* calculating the relative jump size for branching in
*
LDA YMIN          ; minimum y-value of box
TAX
LDA MULT3LO,X     ; multiply by 3
STA YJUMP1        ; store relative jump size
LDA MULT3HI,X
STA YJUMP1+1
```

```
*
CLC                ; patch branch-in
LDA #<STASLIDE     ; low-byte of STA-slide start
ADC YJUMP1
STA jmpSLIDE+1     ; modify JMP-address low-byte
LDA #>STASLIDE     ; high-byte next
ADC YJUMP1+1
STA jmpSLIDE+2

*
* calculating the relative jump size for the slide
exit
*
LDA YMAX           ; maximum y-value
TAX
LDA MULT3LO,X     ; multiply by 3
STA YJUMP2        ; store relative jump size
LDA MULT3HI,X
STA YJUMP2+1

*
CLC                ; store exit address in zero page
LDA #<STASLIDE     ; low-byte of STA-slide exit
ADC YJUMP2
STA zpPOINTER     ; store into zero page
LDA #>STASLIDE     ; high-byte next
ADC YJUMP2+1
STA zpPOINTER+1

*
LDA #$60           ; patch RTS in STA-slide
LDY #$00           ; needed for STA-addressing mode
STA (zpPOINTER),Y

*
* setting of XMIN, XMAX for inner loop
*
LDY XMAX           ; get maximum x-value of box
LDX DIV7LO,Y       ; divide by 7 to get byte number
INX                ; offset correction
STX XCOMP          ; store as X-loop limit

*
LDY XMIN           ; get minimum x-value of box
LDX DIV7LO,Y       ; get loop variable in X-register

*
eraseLOOP
LDA #>staRTS-1     ; generate pseudo-jump address
PHA                ; first high-byte then low-byte
LDA #<staRTS-1     ; for patched RTS & push it on
PHA                ; the stack
LDA #$00           ; put 0 in accumulator
jmpSLIDE JMP STASLIDE ; address STASLIDE gets patched
STASLIDE STA $2200,X ; beginning of STASLIDE
STA $2600,X
STA $2A00,X
STA $2E00,X
STA $3200,X
STA $3600,X
STA $3A00,X
STA $3E00,X

*
* abbreviated here: a total of 128 STA-lines as STA-
slide
*
STA $21D0,X
STA $25D0,X
STA $29D0,X
STA $2DD0,X
STA $31D0,X
STA $35D0,X
STA $39D0,X
STA $3DD0,X

*
* loop bottom: X-register is loop variable
*
```



```

staRTS      INX          ; entry point from STA-slide RTS
            CPX  XCOMP    ; XMAX reached?
            BEQ  eraseRTS ; YES -> exit loop
            JMP  eraseLOOP ; erase next column
eraseRTS     LDA  #$9D     ; revert patched RTS to STA
            LDY  #$00     ; patch address is still in ZP!
            STA  (zpPOINTER),Y
*
            RTS          ; all done
*
* end of fast erase
*
Contact information:
* http://www.golombeck.eu use the online contact form

```

Downloads for the Program

YouTube Video:

<https://youtu.be/goNGzdJIVAI>

DSK image:

http://golombeck.eu/fileadmin/downloads/PLOT3D_242.dsk

General Information on the 3D-Demo:

<http://golombeck.eu/index.php?id=34&L=1>

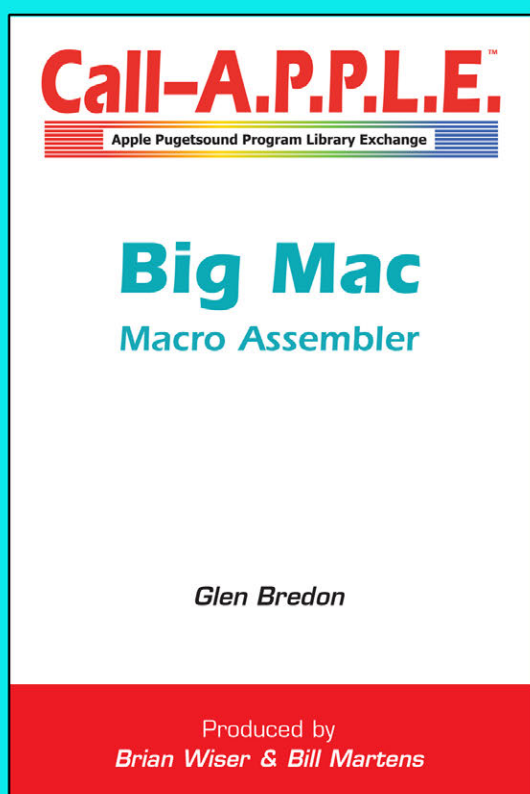


WRITERS and PROGRAMMERS! We're Looking for People Like You

If you are an experienced programmer or author for the Apple / Macintosh computers or iOS, please consider submitting your work to *Call-A.P.P.L.E.* for possible publication in *Call-A.P.P.L.E.* magazine.

Read more and submit at:

www.callapple.org/contact



Big Mac is an advanced editor-assembler for Apple II computer programmers. This book includes documentation for *Big Mac: Macro Assembler*, along with *Big Mac.LC*, *Symbol Cross-Reference*, and *Symbol Symon* that expand its capabilities.

Book and Software at:
www.callapple.org/books

Happy Birthday Steve Jobs!



by **A.P.P.L.E. Staff**

A special program this month comes across our desks just in time to celebrate the life and times of Steve Jobs whose 63rd birthday would have been on the 24th of February.

Artist Pinot W. Ichwandardi of Jakarta, Indonesia has taken the time to sit down and do a point by point pencil type photo, giving everyone a bit of a glimpse of his artistic talents via the Apple II with this program which plots the face of Steve Jobs in low resolution graphics.

```
10 TEXT:HOME: GR
20 COLOR= 15
30 ROW = 1
40 HLIN 0,39 AT ROW
50 ROW = ROW +1
```

```
60 IF ROW <40 THEN GOTO 40
90 COLOR= 0
100 HLIN 7,8 AT 0
110 PLOT 32,0
120 PLOT 34,0
130 HLIN 6,7 AT 1
140 PLOT 9,1
150 PLOT 31,1
160 HLIN 33,34 AT 1
170 HLIN 5,6 AT 2
180 PLOT 8,2
190 HLIN 32,34 AT 2
200 HLIN 5,7 AT 3
210 PLOT 31,3
220 HLIN 33,34 AT 3
230 HLIN 5,6 AT 4
240 PLOT 32,4
250 HLIN 34,35 AT 4
260 PLOT 5,5
270 PLOT 7,5
280 HLIN 33,34 AT 5
290 PLOT 36,5
300 HLIN 4,6 AT 6
310 HLIN 9,10 AT 6
320 HLIN 30,31 AT 6
330 HLIN 34,35 AT 6
```

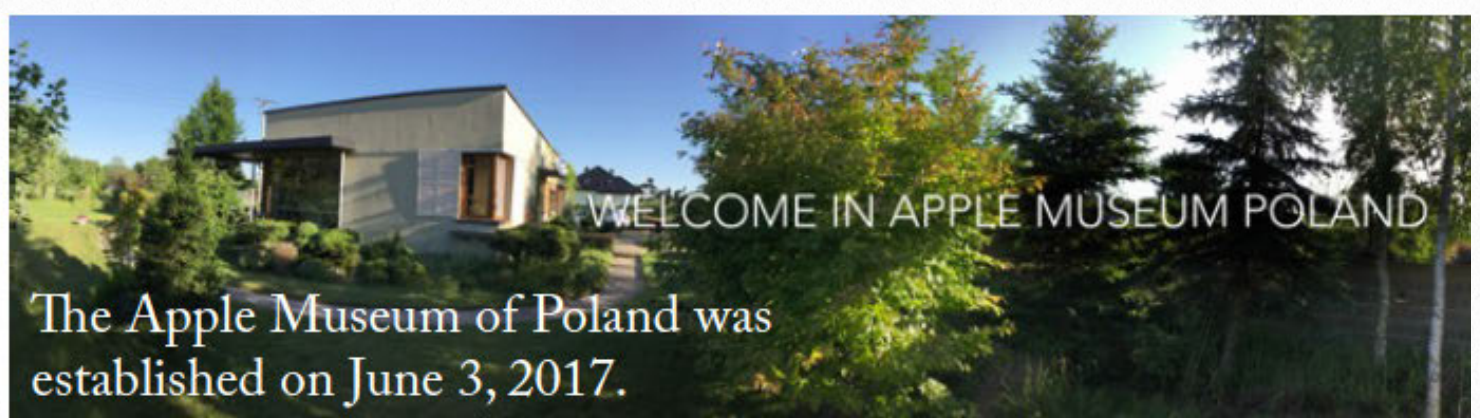
```
340 HLIN 3,5 AT 7
350 PLOT 7,7
360 PLOT 11,7
370 PLOT 13,7
380 HLIN 27,29 AT 7
390 PLOT 32,7
400 HLIN 34,35 AT 7
410 HLIN 3,6 AT 8
420 HLIN 11,16 AT 8
430 HLIN 23,31 AT 8
440 HLIN 33,36 AT 8
450 HLIN 4,5 AT 9
460 PLOT 7,9
470 HLIN 10,12 AT 9
480 HLIN 14,15 AT 9
490 PLOT 17,9
500 HLIN 23,27 AT 9
510 HLIN 31,32 AT 9
520 HLIN 34,36 AT 9
530 HLIN 4,5 AT 10
540 PLOT 8,10
550 PLOT 10,10
560 HLIN 12,13 AT 10
570 HLIN 16,23 AT 10
580 HLIN 25,28 AT 10
590 PLOT 30,10
```


600 HLIN 32,36 AT 10
610 PLOT 5,11
620 PLOT 7,11
630 PLOT 9,11
640 PLOT 15,11
650 HLIN 18,19 AT 11
660 HLIN 22,25 AT 11
670 PLOT 28,11
680 PLOT 29,11
690 PLOT 31,11
700 HLIN 33,36 AT 11
710 PLOT 5,12
720 PLOT 7,12
730 PLOT 18,12
740 HLIN 22,23 AT 12
750 PLOT 25,12
760 PLOT 32,12
770 HLIN 35,36 AT 12
780 PLOT 4,13
790 PLOT 7,13
800 HLIN 18,19 AT 13
810 HLIN 22,23 AT 13
820 PLOT 33,13
830 HLIN 35,36 AT 13
840 HLIN 4,5 AT 14
850 PLOT 9,14
860 PLOT 18,14
870 PLOT 22,14
880 HLIN 32,36 AT 14
890 PLOT 4,15
900 PLOT 6,15
910 PLOT 9,15
920 PLOT 17,15
930 HLIN 22,23 AT 15
940 PLOT 31,15
950 HLIN 33,35 AT 15
960 PLOT 4,16
970 PLOT 6,16
980 PLOT 10,16
990 PLOT 16,16
1000 HLIN 22,24 AT 16
1010 PLOT 30,16
1020 PLOT 35,16
1030 HLIN 5,6 AT 17
1040 HLIN 11,15 AT 17
1050 HLIN 22,23 AT 17
1060 HLIN 25,29 AT 17
1070 HLIN 33,35 AT 17
1080 PLOT 6,18
1090 HLIN 22,23 AT 18
1100 PLOT 32,18
1110 PLOT 34,18
1120 PLOT 7,19
1130 HLIN 22,23 AT 19
1140 PLOT 33,19
1150 PLOT 9,20
1160 HLIN 22,23 AT 20

1170 HLIN 32,33 AT 20
1180 PLOT 7,21
1190 PLOT 9,21
1200 HLIN 22,23 AT 21
1210 PLOT 31,21
1220 PLOT 33,21
1230 PLOT 8,22
1240 PLOT 19,22
1250 HLIN 22,23 AT 22
1260 PLOT 32,22
1270 PLOT 9,23
1280 PLOT 20,23
1290 HLIN 22,24 AT 23
1300 HLIN 31,32 AT 23
1310 PLOT 8,24
1320 HLIN 20,23 AT 24
1330 PLOT 25,24
1340 PLOT 30,24
1350 PLOT 32,24
1360 PLOT 9,25
1370 PLOT 11,25
1380 PLOT 22,25
1390 PLOT 24,25
1400 HLIN 31,32 AT 25
1410 PLOT 8,26
1420 PLOT 10,26
1430 PLOT 19,26
1440 PLOT 21,26
1450 HLIN 23,25 AT 26
1460 PLOT 28,26
1470 HLIN 30,31 AT 26
1480 HLIN 8,9 AT 27
1490 PLOT 11,27
1500 PLOT 15,27
1600 PLOT 24,27
1610 HLIN 28,29 AT 27
1620 PLOT 31,27
1630 PLOT 8,28
1640 PLOT 10,28
1650 PLOT 12,28
1660 HLIN 16,26 AT 28
1680 HLIN 30,31 AT 28
1690 PLOT 9,29
1700 PLOT 11,29
1710 PLOT 15,29
1720 HLIN 24,26 AT 29
1730 PLOT 29,29
1740 PLOT 31,29
1750 HLIN 9,10 AT 30
1760 PLOT 15,30
1770 PLOT 19,30
1780 PLOT 21,30
1790 PLOT 23,30
1800 HLIN 25,26 AT 30
1810 PLOT 28,30
1820 PLOT 30,30
1870 PLOT 9,31

1880 PLOT 11,31
1890 PLOT 14,31
1900 PLOT 18,31
1910 PLOT 20,31
1920 PLOT 22,31
1930 PLOT 24,31
1940 HLIN 26,27 AT 31
1950 HLIN 29,30 AT 31
1960 PLOT 10,32
1970 PLOT 12,32
1980 HLIN 14,15 AT 32
1990 PLOT 21,32
2000 PLOT 23,32
2010 HLIN 25,26 AT 32
2020 HLIN 28,30 AT 32
2030 HLIN 12,13 AT 33
2040 PLOT 16,33
2050 HLIN 20,25 AT 33
2060 HLIN 27,30 AT 33
2070 PLOT 10,34
2080 PLOT 13,34
2090 PLOT 17,34
2100 PLOT 19,34
2110 PLOT 21,34
2120 HLIN 23,31 AT 34
2130 PLOT 14,35
2140 PLOT 16,35
2150 PLOT 18,35
2160 PLOT 20,35
2170 HLIN 22,31 AT 35
2180 PLOT 10,36
2190 PLOT 15,36
2200 PLOT 17,36
2210 PLOT 19,36
2220 HLIN 21,32 AT 36
2230 HLIN 9,12 AT 37
2240 HLIN 16,32 AT 37
2250 HLIN 8,14 AT 38
2260 HLIN 19,36 AT 38
2270 HLIN 7,39 AT 39
2280 GET A\$:END



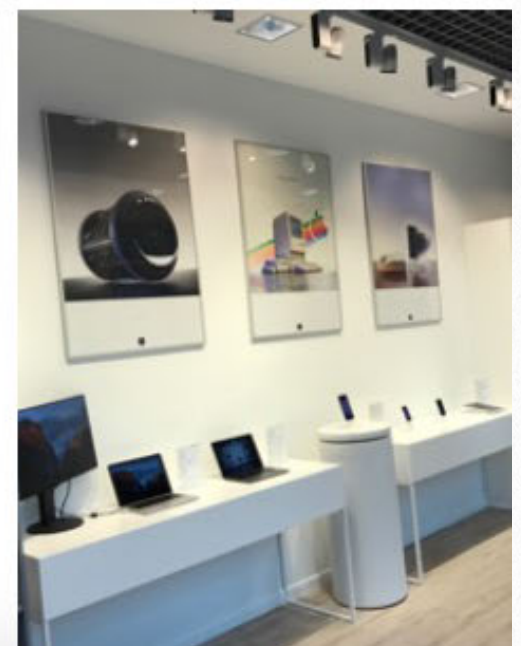


This is a unique private collection started on 1st April 2014, which includes not only old Apple, Macintosh and NeXT computers, but also software, hardware, peripherals, publications and all promotional forms and information from their time. The collection of the Apple Museum includes also historical cult PC computers which competed with Apple during personal computers pioneer time. The collection is live, everyone who is technology or design passionate may learn something, it's a history you can walk on and touch. An idea of the Apple Museum is to educate, discover and promote all these fantastic machines which shouldn't be destroyed or forgotten. The museum is located in an untypical space, in the midst of garden, in the province near Warsaw. You can visit the museum for free.



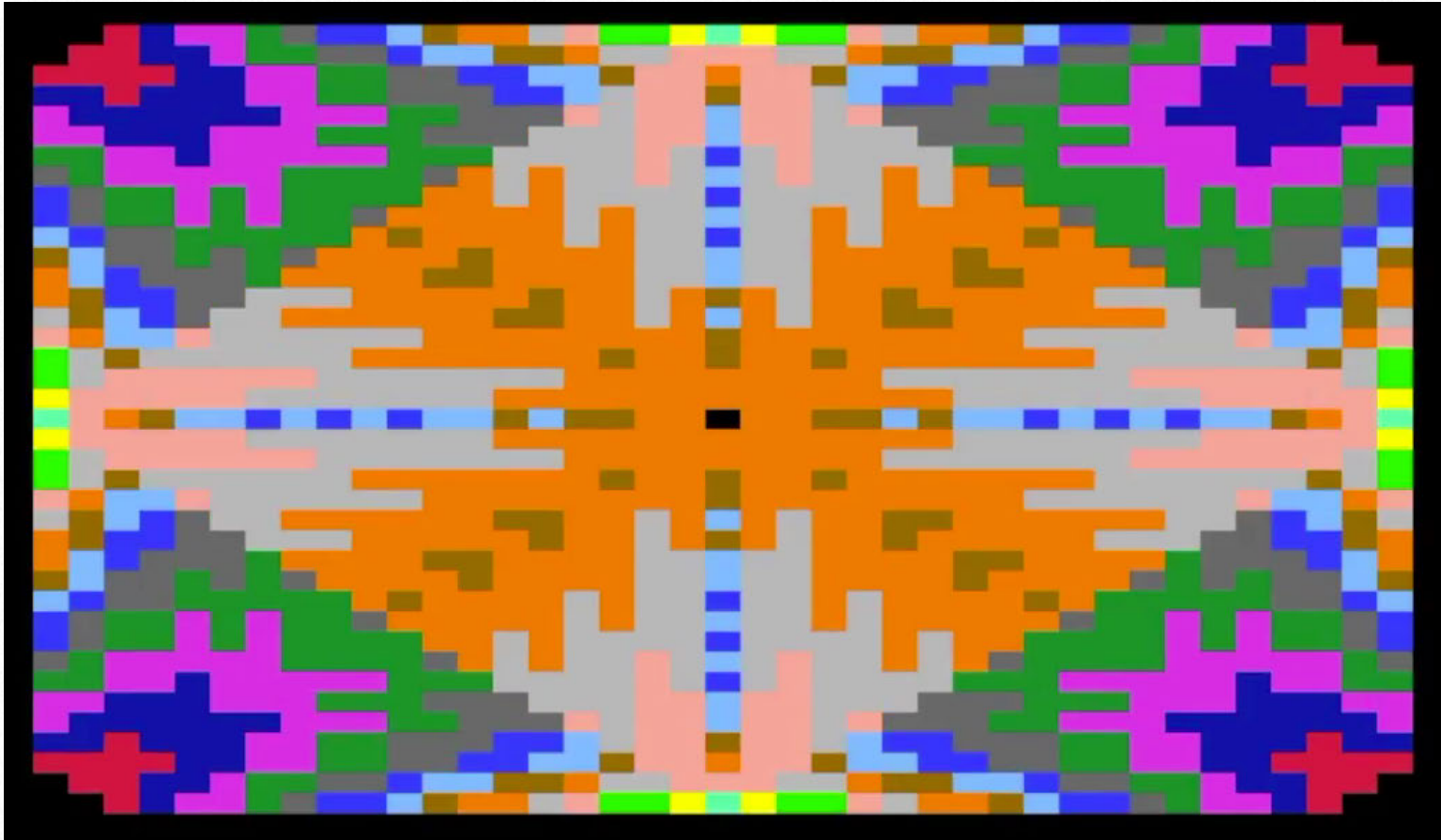
Our second passion is the photography of the old computers and setups with peripherals from the past. Thanks to this, we create original and unique posters. Income from our posters strongly support the museum, help us to maintain the collection in good condition. You can find the posters on eBay.

Look for us : <https://www.facebook.com/AppleMuzeum/> https://www.instagram.com/apple_muzeum_pl/



PLASMA 1.0

Programming Language Released



by David Schmenk

After 12 Years, PLASMA version 1.0 is finally here! You might be asking yourself, "What is PLASMA?" Well, PLASMA is the Grand Unifying Platform for the Apple-1,][, and ///.

Here in, you will find a bit about PLASMA and its associated compiler. You can download the four disk images from the following links (You only need three if you don't plan to boot an Apple ///):

- PLASMA 1.0 System
- PLASMA 1.0 Build Tools
- PLASMA 1.0 Demos
- PLASMA 1.0 Apple /// SOS Boot

PLASMA can be run from floppies, System in Drive 1, and Build or Demos in Drive 2. Mass storage is the

recommended installation that looks like (replacing HARDISK with your volume name of choice):

- System Files => /HARDISK/PLASMA/
- Build Files => /HARDISK/PLASMA/BLD/
- Demo Files => /HARDISK/PLASMA/DEMOS/

Use the System Utilities to copy the floppy images into the above mentioned directories.

Apple-1

The Apple-1 is a very constrained system compared to the][and ///. It is required to have the CFFA1 disk adapter installed to provide file storage and a full 32K of RAM. To get the files onto the CompactFlash card required the use of CiderPress, they must be placed in one directory. Most PLASMA programs won't work on the Apple-1 due to limited filesystem support, video/graphics capabilities, and lack of audio output. It does, however, make a good place to start when porting PLASMA to a new platform.

Apple][

To boot directly into PLASMA, you will need to put the system files in the root prefix of the boot device and make sure PLASMA.SYSTEM is the first SYSTEM file in the directory. Otherwise, start PLASMA.SYSTEM from your program launcher of choice. All Apple II models with 64K and two floppy drives are supported up to a maxed out IIGS with accelerator and hard drive.

65802/65816 Support

PLASMA can use the 16-bit features of the 65802 and 65816 processors to improve performance of the PLASMA VM operation. This is transparent to the programmer/user and doesn't make any additional memory or capabilities available to PLASMA. Launch PLASMA16.SYSTEM to use the 16 bit PLASMA VM. If you don't have the right CPU, it will print a message and restart.

Apple ///

The Apple /// gets the environment it always wanted: The ability to navigate the filesystem with a command line interface. The Apple /// always boots from the floppy drive, even if a hard disk is installed. The PLASMA.SOS floppy should be updated with the SOS.DRIVER configured for your machine. Once booted, type S /HARDISK/PLASMA (or your install directory of choice) to change to, and set, the system directory. This can be automated by creating an AUTORUN file on the boot floppy with the above command in it.

PLASMA Command Line Shell

PLASMA incorporates a very basic command line shell to facilitate navigating the filesystem and executing both SYSTEM/SOS programs and PLASMA modules. It has a few built-in commands:

COMMAND	OPERATION
C [PREFIX]	Catalog prefix
P <PREFIX>	change to Prefix
/	change to parent prefix
V	show online Volumes
S <PREFIX>	set System prefix*
+SOS <SOS.INTERP> [PREFIX]	launch SOS interpreter*
-<SYSTEM PROGRAM> [PARAMS]	launch SYSTEM program**
+<PLASMA MODULE> [PARAMS]	exec PLASMA module

Key to the Above Statements:

[Optional parameters] <Required parameters>

* Apple /// only

** Apple][only

The shell is very brief with error messages. It is meant solely as a way to run programs that accept command line parameters and take up as little memory as possible. It does, however, provide a rich runtime for PLASMA modules.

Included Modules

PLASMA comes with many library modules used by the tools, demos and sample code. The PLASMA system volume must remain in place for the duration of PLASMA's run otherwise it won't be able to find CMD or the system libraries. Probably the most useful included module is the editor. It is used for editing PLASMA source file, assembly source files, or any text file. Execute it with:

+ED [TEXT FILE]

Compiler Modules

The build disk includes sample source, include files for the system modules, and the PLASMA compiler +optimizer modules. The compiler is invoked with:

+PLASM [-W][O[2]] <SOURCE FILE> [OUTPUT FILE]

Compiler warnings are enabled with -W. The optional optimizer is enabled with -O and extra optimizations are enabled with -O2. The source code for a few sample programs are included. The big one, RPNCALC.PLA, is the sample RPN calculator that uses many of PLASMA's advanced features. The self-hosted compiler is the same compiler as the cross-compiler, just transcribed from C to PLASMA (yes, the self-hosted PLASMA compiler is written in PLASMA). It requires patience when compiling: it is a fairly large and extensive program.

Demos

There are some demo programs included for your perusal. Check out ROGUE for some diversion. You can find the documentation here: <https://github.com/dschmenk/PLASMA/blob/master/doc/Rogue%20Instructions.md>. A music sequencer to play through a MockingBoard if it is detected, or the built-in speaker if not. A minimal Web server

if you have an Uthernet2 card (required). Bug reports appreciated.

Source Code

Most sample source code is included from the project. They build without alteration and should be a good starting point for further explorations. The header files for the included library modules are in the INC directory.

Video Playlist

There is a YouTube playlist created for learning PLASMA. It is a WIP, with updates every week or so.

Issues

All the modules and runtime are written mostly in PLASMA; the compiler and editor as well. This means that there may be some startup delay as the PLASMA module loader reads in the module dependencies and performs dynamic linking. But a 1 MHz, 8-bit CPU interpreting bytecodes is never going to match a modern computer. As noted earlier, an accelerator and mass storage are your (and PLASMA's) friend.

All the project modules are included. They have been tested, with the exception of the Uthernet2 driver. I seem to have misplaced mine. If someone can try the Web Server demo in /PLASMA.DEMOS/NET and leave feedback would be very appreciated.

The Apple /// may not always report errors properly or at all.

The documentation is sparse and incomplete. Yep, could use your help...

Changes in PLASMA for 1.0

If you have been programming in PLASMA before, the 1.0 version has some major and minor changes that you should be aware of:

1. Case is no longer significant. Imported symbols were always upper case. Now, all symbols are treated as if they were upper case. You may find that some symbols clash with previously defined symbols of different case.

Hey, we didn't need lower case in 1977 and we don't need it now. You kids, get off my lawn!

2. Modules are now first class citizens. Translation: importing a module adds a symbol with the module name. You can simply refer to a module's address with it's name. This is how a module's API table is accessed (instead of adding a variable of the same name in the IMPORT section).
3. Byte code changes means previously compiled modules will crash. Rebuild.
4. BYTE and WORD have aliases that may improve readability of the code. CHAR (character) and RES (reserve) are synonyms for BYTE. VAR (variable) is a synonym for WORD. These aliases add no functionality. They are simply syntactic sugar to add context to the source code, but may cause problems if you've previously used the same names for identifiers.
5. When declaring variables, a base size can come after the type, and an array size can follow the identifier. For instance:

```
res[10] a, b, c
```

will reserve three variables of 10 bytes each. Additionally

```
res[10] v[5], w[3]
```

will reserve a total of 80 bytes ($10 * 5 + 10 * 3$). This would be useful when combined with a structure definition. One could:

```
res[t_record] patients[20]
```

to reserve an array of 20 patient records.

6. Ternary operator. Just like C and descendants, ?? and :: allow for an if-then-else inside an expression:

```
puts(truth == TRUE ?? "TRUE" :: "FALSE")
```

7. Multiple value assignments. Multiple values can be returned from functions and listed on variable assignments:

```
def func#3 // Return 3 values
```



```
return 10, 20, 30
```

```
end
```

```
a, b, c = 1, 2, 3
```

```
c, d, f = func()
```

```
x, y = y, x // Swap x and y
```

8. DROP allows for explicit dropping of values. In the above func() example, if the middle value was the only one desired, the others can be ignored with:

```
drop, h, drop = func()
```

9. The compiler tracks parameter and return counts for functions. If the above func() were used without assigning all the return values, they would be dropped:

```
a = func() // Two values silently dropped
```

To generate compiler warning for this issue, and a few others, use the -W option when compiling.

10. Lambda (Anonymous) Functions. The ability to code a quick function in-line can be very powerful when used properly. Look here, https://en.wikipedia.org/wiki/Anonymous_function, for more information.
11. SANE (Standard Apple Numerics Environment) Floating Point Library. An extensive library (two, actually) of extended floating point (80 bit IEEE precision) functionality is supported. A wrapper library has been written to greatly simplify the interface to SANE. Look at the RPN CALC.PLA source code as an example.
12. Library Documentation. Preliminary documentation is available on the Wiki: <https://github.com/dschmenk/PLASMA/wiki>
13. Significant effort has gone into VM tuning and speeding up module loading/dynamic linking.

14. The VM zero page usage has changed. If you write assembly language routines, you will need to rebuild.

Credits and Thanks

I wish to thank the people who have contributed to the PLASMA project. They have greatly improved the development of the language and documentation:

Martin Hays: PLASMA programmer extraordinaire. Mr. Lawless Legends has requested many of the crucial features that set PLASMA apart.

Steve F (ZornsLemma): Has taken the optimizer to new levels and his work on porting PLASMA to the Beeb are amazing: <http://stardot.org.uk/forums/viewtopic.php?f=55&t=12306&sid=5a503c593f0698ebc31e590ac61b09fc>

Peter Ferrie: Assembly optimizer extraordinaire. He has made significant improvements into the code footprint in PLASMA so all the functionality can exist in just a few bytes.

David Schmidt (DaveX): His help in documentation have made it much more accessible and professional. Of course any errors are all his. Just kidding, they're mine ;-)

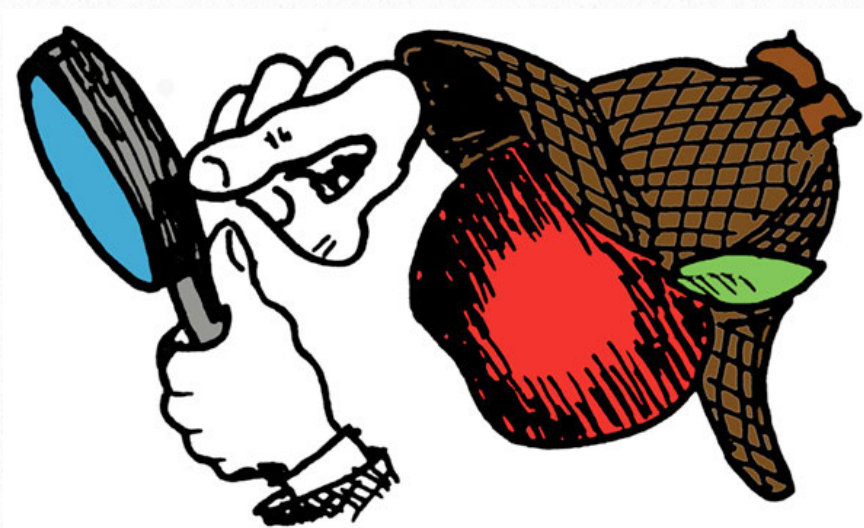
Andy Werner (6502.org): Catching the grammatical errors that I ain't no good at.

John Brooks: Apple II Guru par excellence. His insights got 10% performance increase out of the VM.

Dave Schmenk
<http://schmenk.is-a-geek.com>



The Northern Spy: The X-Factor – X Stands for...



by Rick Sutcliffe

Wrong About the iPhone X

Well, it has happened before...back in '83. See, from a feature point of view, the iPhone 8 looks doomed beside the iPhone X. But price points do come into play in such matters. Apple is curtailing the production of the X, and the Spy assumes it will be discontinued once the X year is done. As the physician said to the 12-year-old Spy when he came into his office looking like scarlet fever, "Allergic to the phenobarb, eh?" Well, back to the old drawing board. Somehow anniversary special products just don't do well. Are they too hastily conceived?

Wrong About Grammar

A local radio ad grates on the Spy every time (often!) he hears it. It's a plug to take out a second mortgage on one's home equity for immediate spending wants (high interest contract no doubt, and very high risk behavior) that ends with "our criteria is less strict". Has no one ever been told about singular and plural noun-verb agreements? And don't let's get started on the correct pronunciation of "Wednesday" and February", much less split infinitives. Yes and the Spy votes against Churchill in that the rule that one must not end a sentence with a preposition is one up with which he will indeed put.

OS X

Though announced well before, it was shipped in spring 2001. The Spy still has the black leather jacket with

the Aqua X between the shoulder blades that was given to the first 1200 registrants at WWDC that year. Like his copy of the red book, a "forever" keepsake. Will there ever be an OS XI? Stay tuned. The Spy is once again expecting big things at WWDC – perhaps more than just a new Mac Pro. Seriously, though, it may be time for a revamp.

Wrong About Toasters

PC's running W*nd*s did indeed become commoditized – interchangeable (and cheap) parts, dime-a-dozen plain-joe designs and a practical desk-life in constant use of perhaps three years before parts begin to fail. Trash and start over. The Mac avoided this fate because its designers thunk different. (Bonus question for old-time dweebs: What is/was a "thunk"?) Even under very heavy use, one can count on a Mac to retain value for years after most PCs bought the same year have been moved to a landfill.

But the main point here is that even in the worst (PC) case, computers are not, and never will be, toasters. They are compound sliding miter saws – versatile tools without a predefined or locked-in task set so the craftsman can experiment, innovate, and find never-thought-of ways to use the tool for creative tasks. Assume the hardware will outlast any PC in comparative use. It'll be the desire to run updated (and often bloated) software bigger and faster that obsoletes a Mac. The Spy has fifteen-year-old machines that still work – with the software of a past era.

But speaking of saws, the Spy has just given away his older 10" blade compound sliding miter saw (still works) and purchased (today) a Bosch 12" dual-bevel-compound-sliding miter saw in its place. He was lured by the increased capacity, the ability for the articulation mechanism to sit against the wall, glowing reviews online, and a pretty good (though still expensive) price at KMS tools (last day of the sale, though). A review will follow in a few months. However, what letter of the alphabet does the promo picture make with the bevel ghosted in both left and right positions?

Wrong About the Stock Market – So Far

The Spy was convinced last summer that the stock market was overpriced and due for a big fall. Not. The euphoria over tax breaks for business and the super rich at others' expense has sent the markets to dizzying highs. All this sometimes physics fellow can say is, "What goes up must come down." Thing is, the downslide could be steep and unprecedentedly deep – 2008 was just practice. Yikes! Other thing is, in the information friction-free and instantaneous happenings of our electronic age, when it starts, everyone will want out at once. If (when) this scenario does play out, the very advantages of our rapid info society will turn against us big time.

Crossover

No, not crosswalks. The Spy wants to retrieve his old Apple][files, bring the over to a Mac, and run them in a simulator. Ah....why? Well, he's teaching the hardware course, which includes among many other things, sections on machine language, op codes, and their ilk. Last few times through, the students had a very hard time getting their minds around the low-level ideas when expressed in the rich (but therefore complex and confusing to the novice) Intel op code set and assembler. He thought a nice simple 6502... Well, that might not work out, but it's worth a try.

The software link between an Apple][(though in this case it's an Apple IIGS) is ADTPro by David Schmidt. The hardware link is a pair of cables—a null modem (crossover cable don't ya know) from the Apple modem port to a DB-9 connector and a serial adapter from there to USB on the Mac. In the old days, one analyzed the transmit and receive signals (to determine whether the two sides wanting to transact information exchange had been designed as

senders or receivers) and built a cable for such purposes. Yup, every computer to printer cable was an adventure... could tell stories... But the Spy has forgotten where he stored his breakout box, so gave up that project and ordered a ready made null modem cable. Will let'cha know.

'Course he's a CP snow-defying crossover himself, having one foot firmly planted in computing, another in pure mathematics, and a third in the semi-literary world of an SF author – not to mention many other interests (such as working on a merger of two churches). Good thing there's only thirty hours in the day, eh?

URLs for Rick Sutcliffe:

Northern Spy Home Page: www.TheNorthernSpy.com

Author Site: www.arjay.ca

Arjay Blog: www.arjay.bc.ca/blog

Publisher's Site:
www.writers-exchange.com/Richard-Sutcliffe.html

The Fourth Civilization--Ethics, Society, and Technology (4th 2003 ed.) : www.arjay.bc.ca/EthTech/Textindex.html

URLs for resources mentioned:

ADTPro: adtpro.com

Bosch: www.boschtools.com

KMS Tools: www.kmstools.com



Apple Archives

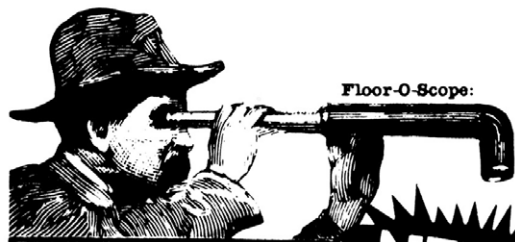
The Best Vintage Apple & Mac Websites

www.applearchives.com



DON'T BLOW YOUR BUCKS ON Locked-Up Software!

**Beagle Bros Apple Utilities are Listable, Backup-able
Customizable and Compatible with Normal Apple DOS.**



Apple Mechanic

Shape Writer/Byte-Zap Utility
by Bert Kersey

Another hot multiple-utility disk—Nine useful, listable, copyable & customizable programs—

SHAPE EDITOR: Put professional hi-res animation in your programs. Keyboard-draw any shape & let your Apple write a shape table & store it on disk. Design large/small custom typefaces too, with special characters. Many fonts on disk. LIST-able demos show how to use shapes to animate games, displays, and CHARTS & GRAPHS. A valuable time-saving utility/learning tool.

BYTE ZAP: A MUST utility. Rewrite any byte on a disk by loading a sector onto the screen for inspection. HEX/DECIMAL/ASCII display optional. Examine bytes via cursor control; enter hex, dec or ascii to change. Create illegal filenames, restore deleted files, change greeting program name, repair/protect disks, change DOS, examine program files. Clear illustrated instructions show how disk data is stored and how to access it.

MORE: A disk PACKED with useful music, text & hi-res tricks FOR USE IN YOUR PROGRAMS. Demo-writer, hi-res utilities and excellent, educational, entertaining documentation.

ONLY \$000 ☐ Apple Mechanic disk (48K min.)
☐ Beagle Bros Tip Book #5 (60 pgs.)
☐ Peek, Pokes & Pointers Chart

Tip Disk #1

by Bert Kersey

100 programs from Beagle Bros' Tip Books 1, 2, 3 & 4—Dozens of tricks to make your Apple do things it's never done! All 100 programs are listable, copyable and changeable; each teaches another fascinating Apple programming technique.

ONLY \$000 ☐ Tip Disk #1 on disk (32K or 48K)
☐ Peek, Pokes & Pointers Chart
(Note: No tip book with this disk)

Alpha Plot

Hi-Res Graphics/Text Utility

by Bert Kersey & Jack Cassidy

Here are a few of Alpha Plot's useful features. Compare with others on the market—

HI-RES DRAWING: Create hi-res pictures & charts with text, on both pages; all APPENDABLE TO YOUR PROGRAMS. Optional Xdraw cursor (see lines before drawing). Mix colors & Reverse (background opposite). Circles, Boxes, Ellipses; filled or outlined. COMPRESS HI-RES TO 1/3 DISK SPACE. Relocate any portion of an image anywhere on either page. Superimpose too & convert hi-res to lo-res for colorful abstracts!

HI-RES TEXT: Beautiful upper/lower case with descenders (no hardware required). Color & reverse characters positionable anywhere (no tab limits). Professional-looking PROPORTIONAL SPACING; adjustable character height & letter spacing. Multi-directional typing for graphs!

ONLY \$000 ☐ Alpha Plot on Disk (48K min.)
☐ Beagle Bros Apple Tip Book #4
☐ Peek, Pokes & Pointers Chart

DOS BOSS

DISK COMMAND EDITOR
by Bert Kersey & Jack Cassidy

A classic Apple utility you will ENJOY! Rename DOS commands (CATALOG can be "Cat", etc.). PROTECT PROGRAMS: any unauthorized save-attempt produces a "Not Copyable" message. Also LIST-PREVENTION & 1-key program-run from catalog. Custom catalogs: Change Disk Volume message to your title; Omit/alter file codes. Rewrite error messages: Syntax Error can be "Oops!!" or anything! Fascinating documentation included; Hours of good Apple reading!

Dos Boss's change features may be appended to your programs. Anyone using your disks (booted or not) formats their DOS as YOU designed it.

ONLY \$000 ☐ Dos Boss on Disk (32K/48K min.)
☐ Beagle Bros Apple Tip Book #2
☐ Peek, Pokes & Pointers Chart



Utility City

21 Useful Utilities on One Disk
by Bert Kersey

LIST FORMATTER makes properly-spaced and indented listings with page breaks; each statement on new line, if-thens & loops called out; a great de-bugger! MULTI-COLUMN CATALOG in any page-width to printer or screen. Auto-post Run-Number & last-used Date in programs. Put INVISIBLE working commands in listings. Access program lines in memory for repair & illegal alteration. Alphabetize & store info on disk. Run any program while another stays intact. Renumber to 65535. Save inverse, trick and INVISIBLE FILE NAMES. Convert dec to hex & binary, or INT to FP. Append programs. Dump text screen to printer... 21 LISTABLE PROGRAMS TOTAL!

21 PROGRAMS FREE ☐ Utility City on disk (48K min.)
☐ Beagle Bros Apple Tip Book #3
☐ Peek, Pokes & Pointers Chart



**GOTO your
Apple User Group**
www.callapple.org

Apple Pugetsound Program Library Exchange (A.P.P.L.E.) founded in 1978, is a resource for retro and modern Apple news, software, and new magazines and books. Check out our Member Benefits!

NOW WILL YOU MARRY ME,
VICKY? NOW THAT I'VE
GOT MY OWN BEAGLE BROS
PEEK & POKES CHART?

Bonuses With Every Disk!

Poke your Apple all night long with this free reference poster!

The most useable **PEEKs, POKES, POINTERS** and **CALLs**, scrounged up from every source imaginable!

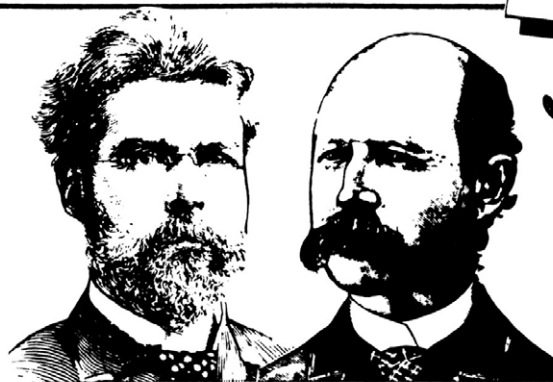
Apple Tip Books too—

Each disk comes with a Gold Mine of valuable Apple information and hours of entertaining reading matter, including dozens of tips and keyboard experiments on all subjects—DOS, Copy Protection, Graphics, Shape Tables, Hardware and More. Sample programs too, such as "Programming the Reset Key" and "Copy Stoppers".

Each disk comes with its own unique book.

24 Hour TOLL FREE Web Site

beagle.applearchives.com



Beagle Bros
MICRO SOFTWARE

<http://beagle.applearchives.com>

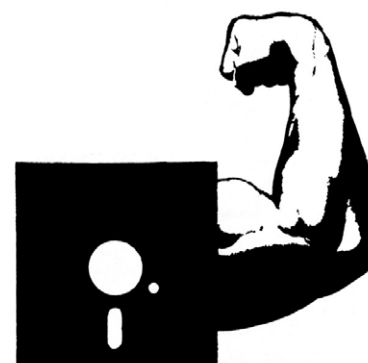
"APPLE" is a registered trademark of Apple Computer, Inc.
Our "BEAGLE BROS" site is the only Officially Approved one.

OR ORDER BY CARRIER PIGEON

RUSH! The disk packages checked below
Plus the Tip Book & PEEKS/POKES CHART.

☐ Alpha Plot ☐ Dos Boss ☐ Tip Disk
☐ Apple Mechanic ☐ Utility City

(Use this coupon or separate sheet.)



Blankenship BASIC: The Return of the Programmer

```
BLANKENSHIP BASIC VERSION 2.7
COPYRIGHT JANUARY 1984
BY JOHN BLANKENSHIP
ALL RIGHTS RESERVED!

SELECT THE NUMBER OF YOUR CHOICE

1. ENTER BBASIC (WITH HIRES GRAPHICS)
2. ENTER BBASIC (MAX MEMORY-NO HIRES)
3. INITIALIZE A BLANK DISK WITH BBASIC
4. ADD BBASIC TO AN OLD DISKETTE
5. DOCUMENTATION & INFORMATION
6. ENTER NORMAL APPLESOFT

ENTER YOUR CHOICE *
```

by **Bill Martens**

This past month, I was communicating with John Blankenship on the possibility of bringing back the flagship product of his 1980s production, Blankenship BASIC. Originally produced in 1984, Blankenship BASIC offered a highly-expanded version of BASIC for the Apple II computer, using the Ampersand hooks in the the BASIC Interpreter to create something that not only gave the programmer a greater power over their programs, but also allowed highly-structured programming to be performed on a machine known for its unstructured Applesoft II.

Out of the blue, a package arrived in the mail with the original manual for Blankenship BASIC as well as a printout of the entire source listing for Blankenship BASIC 2.7, the current version of the program.

Since we have access to all this information and data, we set about actually re-producing the original manual in an effort to not only make it more readable but also to make it more useful to more people.

As we were beginning our quest to locate both copies of the program, a surprise appeared on Facebook. Robert Knepp had found the ProDOS version of the program amongst his disks. Initially, we were a bit skeptical due to

the fact that most of the copies of the program we had seen were of version 2.6 or 2.7.

But sure enough, Robert's copy was the ProDOS version, meaning that we now had both versions in hand as well as the source code. Instead of just typing the manual, we decided to put Siri to the test to see if she was capable of transcribing a computer manual in a usable form.

Once again, I was reading computer data aloud. The effort from Brian Wiser and I with *What's Where in the Apple* in 2015 had been the other time where reading the memory locations for the *Atlas* and *Gazetteer* sections proved to be the fastest way to handle things. Only in that case, Brian was performing the transcription.

Thus, with a bit more cleanup, we will have a brand new manual for BBASIC 2.7 that Brian is laying out and assembling into an easy-to-read publication, along with DOS and ProDOS versions of the software.

We must thank John Blankenship for allowing us to distribute the package as well as Robert Knepp for finding the ProDOS version of the program. The source code for the program will also be available sometime in the summer.



MICRO™

THE 6502/6809 JOURNAL



micro.applearchives.com


EAMON

Adventurer's Guild Online



www.eamonag.org

A 80's Apple II BBS



Where The Real Old School Meets

a80sappleiibbs.ddns.net:6502

```

/*****
    ****
    /@      Entering:  %@#
    /@ *@@(      Captain's  %@
    @,      @@@      Quarters  @.
    .@      /@@%      BBS      @###
    #@ #&      ###@#      @
    @@ @@      @@@*      @ @#
    &@ &@      ,.      @@@@,      @ @@
    @@@@(@@@@@@      &@@@@@@@@@@@@ @@
    @@ @@@@@@@@@      @@@@@@@@@@@@@/
    @@ @@@@@@@@@&.@@ @@@@@@@@@& @@
    @@      ,@@@@ @@@&      @@
    @@* @@@@@ @@@@@ @@@@ @@(
    @@@@ @, @@ .@ @% (@@@@@
    @@@@@@.      .&@@& @.
    &@ #@.( @ @ @ & @ @ @
    @* @@@@@@@@@@@@@@@ @
    @ @ , # @ @ /, # @
    @@& @&@(@ %&, . @@@
    *@@      *@@*
    |@@@@@@@@@%
  
```

Running on a real Apple IIGS! Featuring Door Games, Text Files, Apple II Downloads, and Retrocomputing Forums. Relive your youth at:

<telnet://cqbbs.ddns.net:6502>

VCF Pacific Northwest 2018

by Evan Koblentz



Seattle has 1,000 kinds of coffee, several professional sports teams, and even a volcano – but until now it lacked a Vintage Computer Festival. That changed on

February 4-5 this year with the

Vintage Computer Federation's

inaugural Vintage Computer Festival Pacific

Northwest, hosted at Living Computers: Museum+Labs owned by Microsoft co-founder Paul Allen.

The original festivals began in the 1990s. Today, the Federation as a 501(c)(3) non-profit organization operates VCF East (May 18-20 at the official VCF museum in Wall,

A glimpse into the excitement in Seattle.
For the first time, VCF comes to us!

Not exhibiting but spotted in the audience was John Morris of Applesauce fame. We're sure he gave many elevator pitches and answered many questions about the latest-and-greatest approach to Apple II disk cracking. Living Computers: Museum+Labs also had their own exhibits open for everyone to see, including their Apple-1 setup for use!

We know there are plenty of Apple II hobbyists who live not-so-far from the Seattle region. Our intention is for Vintage Computer Festival Pacific Northwest to be an annual event around the same time of year. So, if you live within a day's drive of Seattle – or if you're willing to fly there, as other exhibitors and attendees did from all over the U.S. and Europe – then consider bringing the Apples out in force for this show next year.



New Jersey), VCF West (August 4-5 at the Computer History Museum in Mountain View, California), and the new Seattle event. Planning is underway for additional locations. Other regional editions such as VCF Southeast (Atlanta, April 21-22), VCF Midwest (autumn: dates TBD) and the European shows are independent.

Twenty exhibitors signed up for VCF-PNW. Two were of special interest to Apple II hobbyists: the Alpha Syntauri, by Jason Howe of Lake Forest Park, Washington, and the MOnSter6502 by Eric Schlaepfer of Sunnyvale, California. Jason's exhibit of the hard-to-find digital synthesizer brought a great background soundtrack to the exhibit hall, while Eric's discrete homebrewed 6502 added its own excitement.

Thank you to all who attended in the Apple II community!

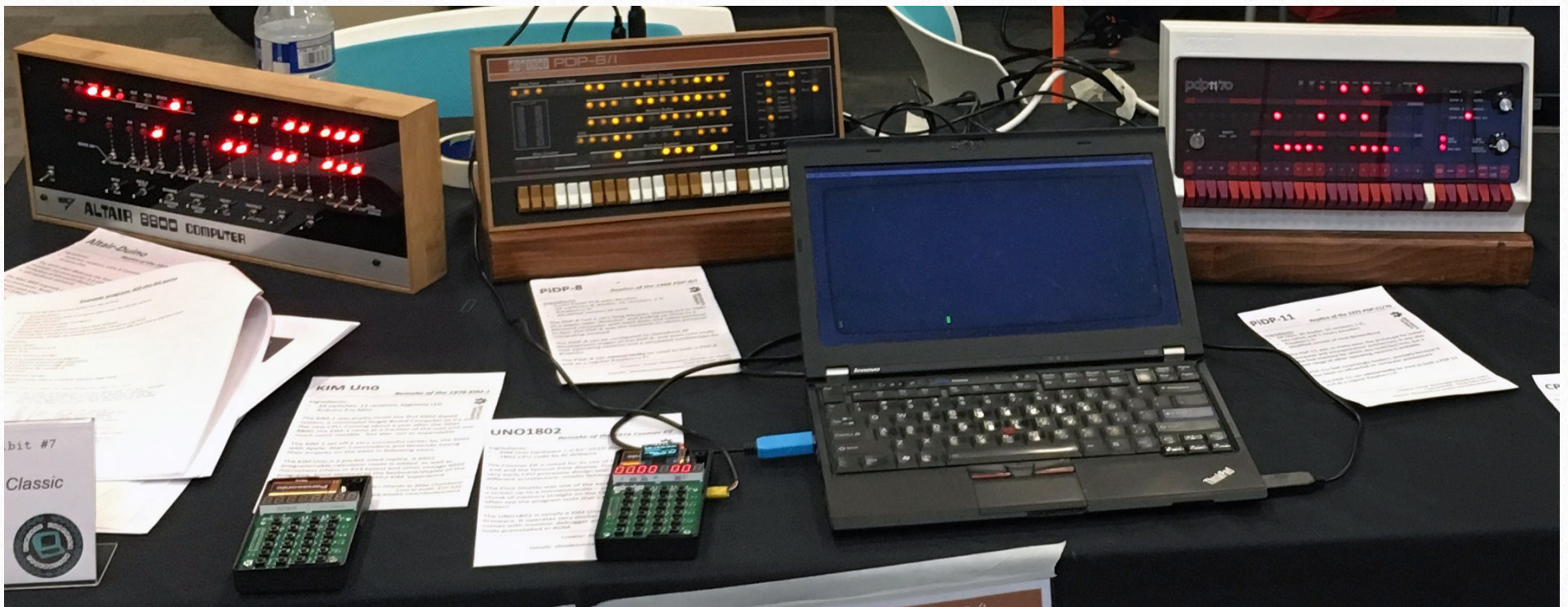
Evan Koblentz
Executive Director
Vintage Computer Federation
evan@vcfed.org
<http://vcfed.org>

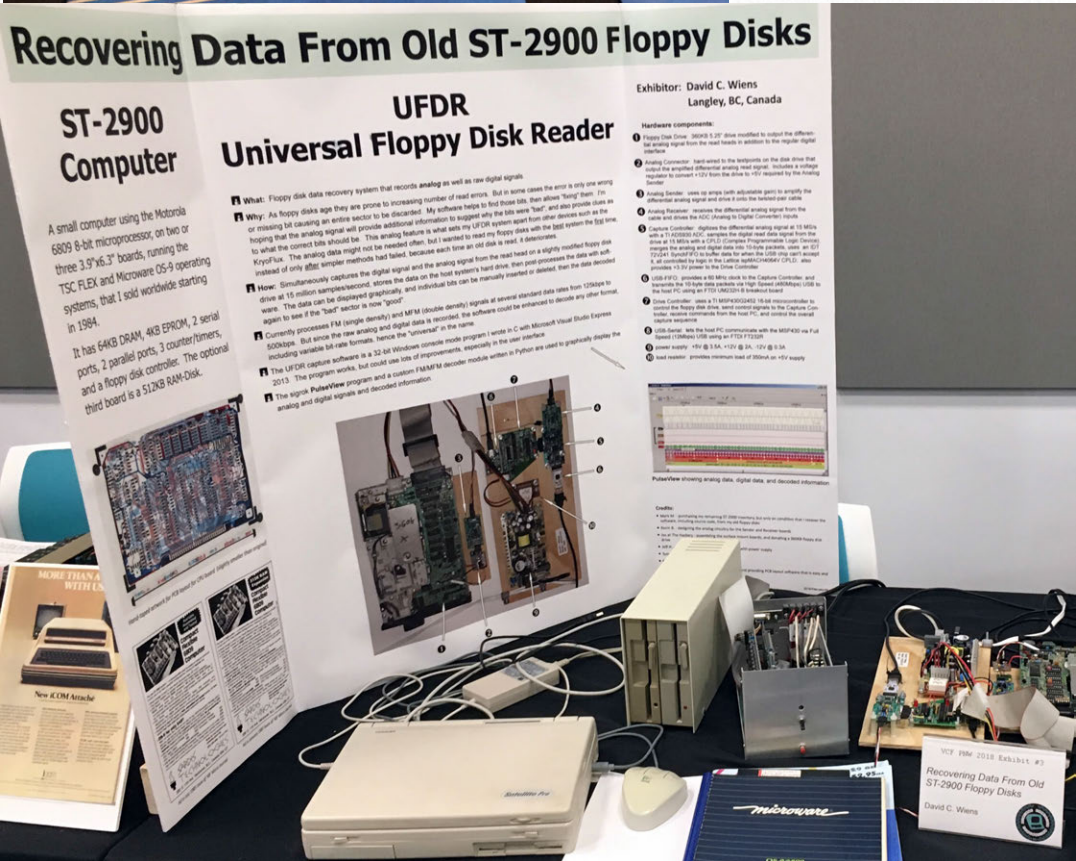


VCF Pacific Northwest in Photos

by Kevin Savetz









**HOW TO COMPUTERIZE YOUR HOME
USING YOUR APPLE II COMPUTER**

Produced by
Brian Wiser & Bill Martens

Available at: www.callapple.org/books

Complete with all programs on FREE floppy disk image

VCF Southeast gets its First Vendor

David Greelish becomes the first confirmed vendor for the 2018 rendition of VCFSE

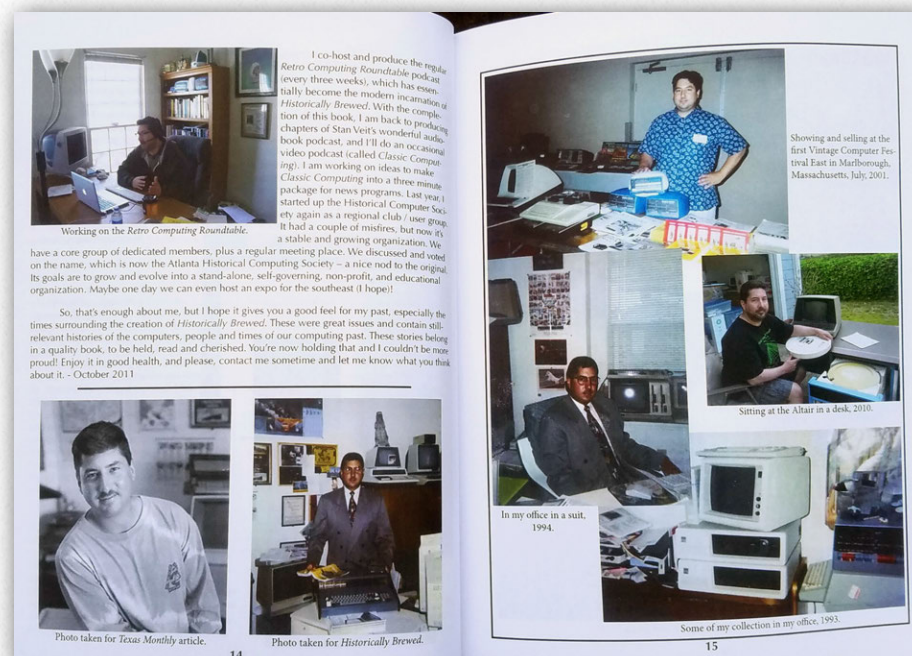


by A.P.P.L.E. Staff

Vintage Computer Festival Southeast announced that they had a confirmation of their first confirmed vendor for the forthcoming VCFSE. David Greelish, famous for his book, *Classic Computing: The complete Historically Brewed*, has confirmed that he will indeed be the first vendor going to the event. The event has been announced for April 21 in Roswell, Georgia.

In 1993, David Greelish created the Historical Computer Society, where he published a fanzine named, "Historically Brewed." The HCS was an online, international user group for fans of computer history, and "HB" was the newsletter communications tool. Ultimately, nine issues of "HB" were printed and distributed, with a tenth issue never completed. This last issue was to be called, "Classic Computing."

In 2011, he ran a successful Kickstarter campaign, garnering the funding for the project and publishing all ten issues into one book. The book also included his story of what brought about the magazine.



1994/06/top-ten-29/

http://www.classiccomputing.com/CCPodcasts/CC_Show/Entries/

[2010/10/11 Classic Computing, Ed Roberts interview 1995.html](#)



racks or even rooms is also a major part of this book.



obsolete computers as nothing other than trash.

Classic Computing website:

http://www.classiccomputing.com/CC/My_Book.html

vendor, visit their website:

festival-southeast/

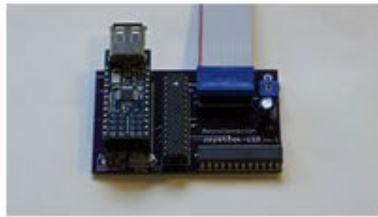
NOX ARCHAIST



Forged from Raw Bits by 6502 Workshop
www.6502workshop.com

RetroConnector

Accessories and Cables for Your Apple II Computer



RetroConnector Joystick Interface for Appl...

\$50.00



RetroConnector Joystick Shield

\$40.00



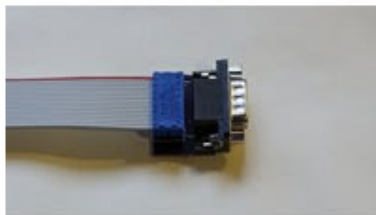
RetroConnector keyboard shield - Apple IIc...

\$45.00



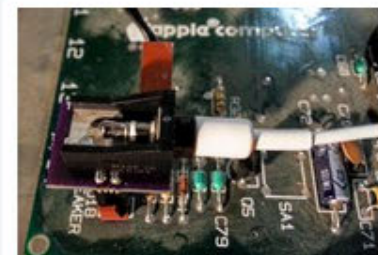
RetroConnector keyboard shield for Apple IIe

\$45.00



Joystick adapter for Apple IIc

\$10.00



Speaker Connector for Apple II

\$8.00



RetroConnector keyboard interface for Appl...

\$60.00



M0100 to USB Conversion

\$55.00



Apple IIe Card Y-Cable

\$45.00



Apple III Disk II Cable

\$6.00



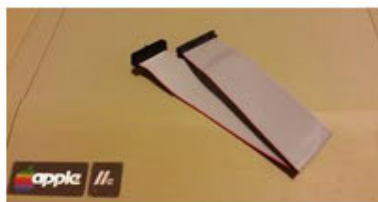
Uthernet II Light Pipe

\$30.00



4 connector TRRS 3.5mm patch cable (M/M) 3...

\$2.50



Apple IIe Keyboard Cable

\$6.00



Apple 9-pin mouse to USB adapter

\$25.00



DIP-16 cable, male to male

\$5.00

Ultimate apple2.com

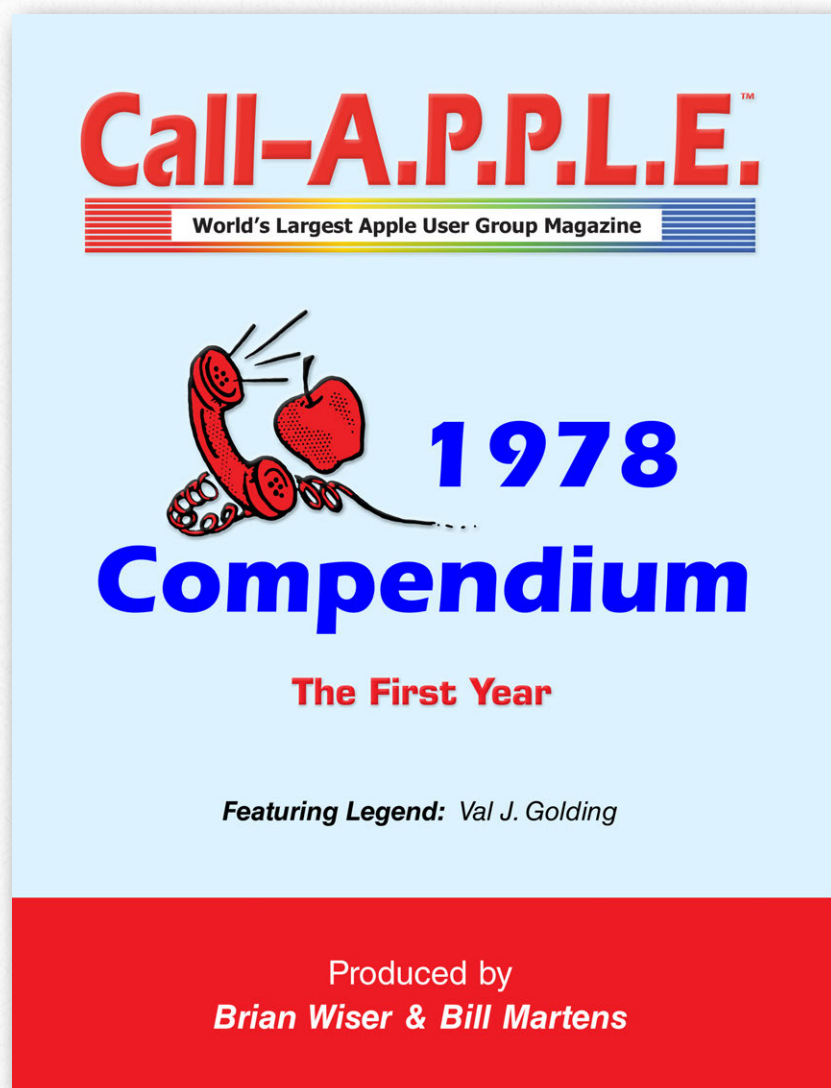


It's Either Ultimate...Or It's Not!

www.ultimateapple2.com

Call-A.P.P.L.E. Compendiums

The complete *Call-A.P.P.L.E.* magazine in its original glory – but better!



All issues from 1978 and 1979

Call-A.P.P.L.E. magazine

in Paperback and Hardback.

Get your copy through the A.P.P.L.E. Bookstore:

www.callapple.org/books

Freshly Squeezed Reviews: HomePod Brings Me Accessibility

by Frank Petrie

Just a quick one. Apple has long been in the forefront of making their software accessible. They've done fantastic work for the visually impaired. And to show how seriously they take this, there's an Accessibility Preference in System Preferences on the Mac OS. And under General > Settings > Accessibility in iOS 11 there are adjustments that benefit the hearing impaired.

Now, as I have mentioned before, I have MS and am wheelchair bound. I'm independent but some things require extra effort and forethought. One example is getting into bed. Because I still retain some tone in my legs, I can perform standing transfers. It requires a bit of gymnastics and engineering to get into my bed each night but I manage to accomplish it.

For pragmatic reasons, I'm considering purchasing Apple's HomePod. Not for it's musical abilities (I love listening to music through a set of Sennheiser cans and purchase uncompressed music every now and again) but for how it can help me with my independence day-to-day.

Let me set up my scenario. I live alone but can cook, do my laundry, bathe, drive, etc. But once I'm in bed, there I stay. If I have to get out of bed repeatedly (as little as two times, some nights) my legs become fatigued and become less responsive. Then what happens is I take the easy way out; lean against the wall in my power chair for the night.

HomePod would enable me to avoid these situations. Say I'm in bed and forgot to shut off a light or something of that nature. Instead of going through all the physical steps necessary to accomplish the task, I merely ask Siri to take care of the issue for me, allowing me to return to sleep. In this case, the hardware provides my accessibility.

Now, I know that I can accomplish this for a fraction of the cost with similar devices but I've been steeped in the Apple ecosystem for two and a half decades now. And I want to keep everything that way. I don't want to introduce other OSs.

Of course, I could accomplish simple things with my iPhone and HomeKit. But I'm intrigued by what else I may be able to do in the future when AirPlay 2 arrives on the scene. Will I not only be able to use it with my Apple TV but somehow with my regular HDTV? And you know that people are already thinking of work arounds for that! And if I can afford a second one, will FullRoom provide me with a better listening experience in my studio apartment?

Even though my iPhone rests in my Qi charger every night on my nightstand, I could listen from bed to audio podcasts or the day's news with better sound quality than my iPhone could provide.

Plus, truth be told, it gives me an excuse to buy another toy. Funny, I'll bet Apple never envisioned that the HomePod could be an important accessibility device. But there you are. That's my driving force to acquire one.

© 2018 Frank Petrie



Apple II Disk and File Utilities

ADTPro 2.0.2



ADTPro, which allows users to image Apple II and Apple III disks, using either a serial connection or an ethernet connection. The program also allows boot strapping of a machine via those same methodologies.

Download ADTPro from: <http://adtpro.com>

AppleCommander 1.3.5.14

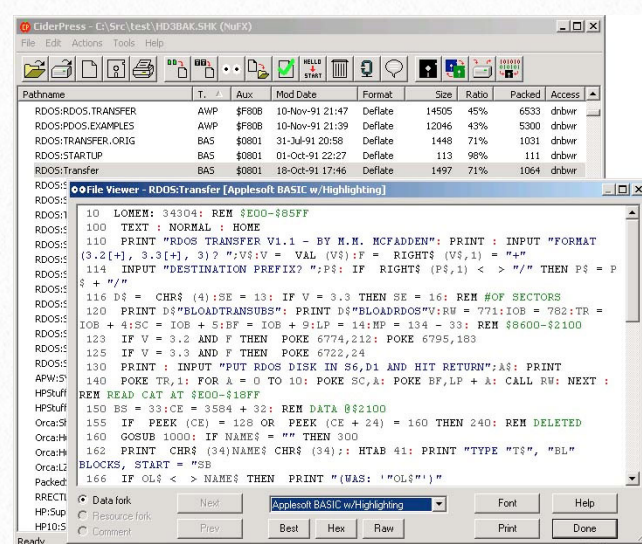


AppleCommander is a disk image file manipulation package which allows conversion of files between formats as well as the export of files. AppleCommander is available as a JAR file and runs on most major platforms that run Java.

Visit Dr. John B. Matthews site:

<https://sites.google.com/site/drjohnbmatthews/applecommander>

Ciderpress 4.0.3



Ciderpress is an Apple II disk image manipulation package which allows users to copy images and files between disks, both virtual as well as physical. Compact Flash media can also be written and updated with the Ciderpress package.

Key Features:

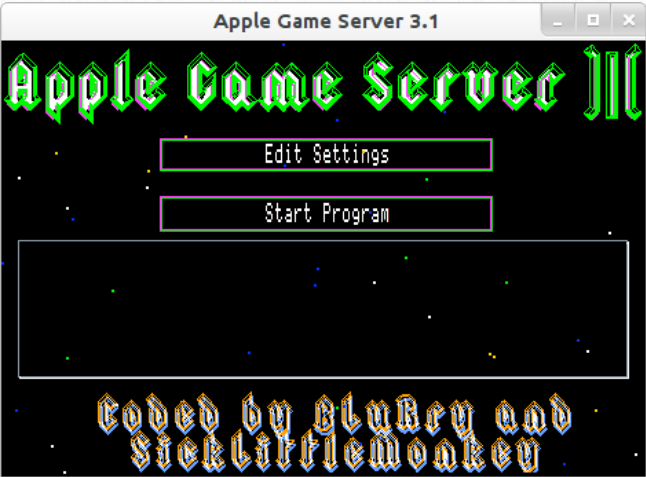
- Full support for ShrinkIt archives.
- Full support for all disk image formats and Apple II filesystems.
- Direct access to hard drives, removable media, and CF cards.
- Converters for text and graphics files.
- Disk image creation and conversion utilities.
- Some handy disk manipulation tools.

Version 4 is intended to run on Windows XP and later OSes.

The previous version of Ciderpress, version 3.0.1, which runs on Windows 98, ME and 2K, has been left on the server as the current official release version of the software will not work on these older operating systems.

Download Ciderpress from: <http://a2ciderpress.com>

Apple Game Server 3.1



Apple Game Server allows users to boot their Apple II computers without floppy disks. The only requirement for this is a functional serial connection to a computer running the Apple Game Server software.

Check out the Apple Game Server file repository:
<https://sourceforge.net/projects/a2gameserver>

A2Command 1.1

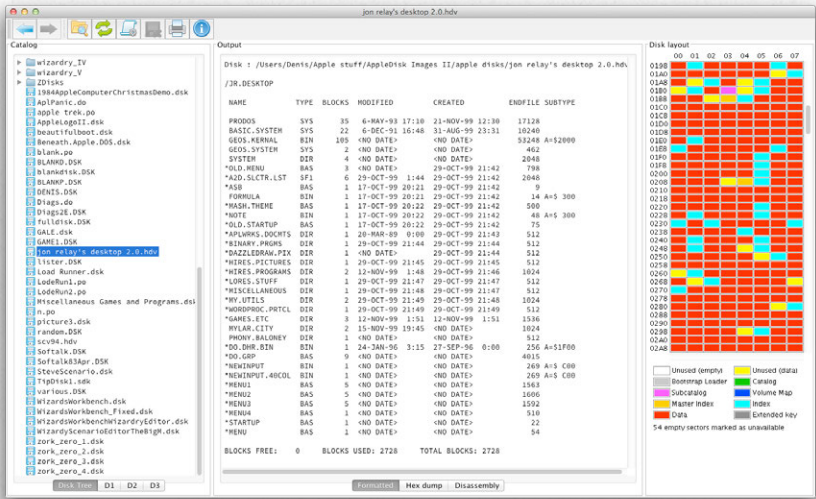


A2Command is a file manager which runs on any Apple IIe or newer Apple II series machine and many Apple IIe clones. The program allows you to manage files on the disk in the same manner as Norton Commander and includes the following features:

- Dual Panel Operation
- Text File Viewing
- Copy single files and batches of files
- Delete single files and batches of files
- Rename files
- Create, delete and navigate directories
- Write Disk Images to Physical Disks
- Write Physical Disks to Disk Images
- Copy Physical Disk to Physical Disk

Download A2Command from:
<https://github.com/A2Command/a2command>

Apple II Browser 1.0.24



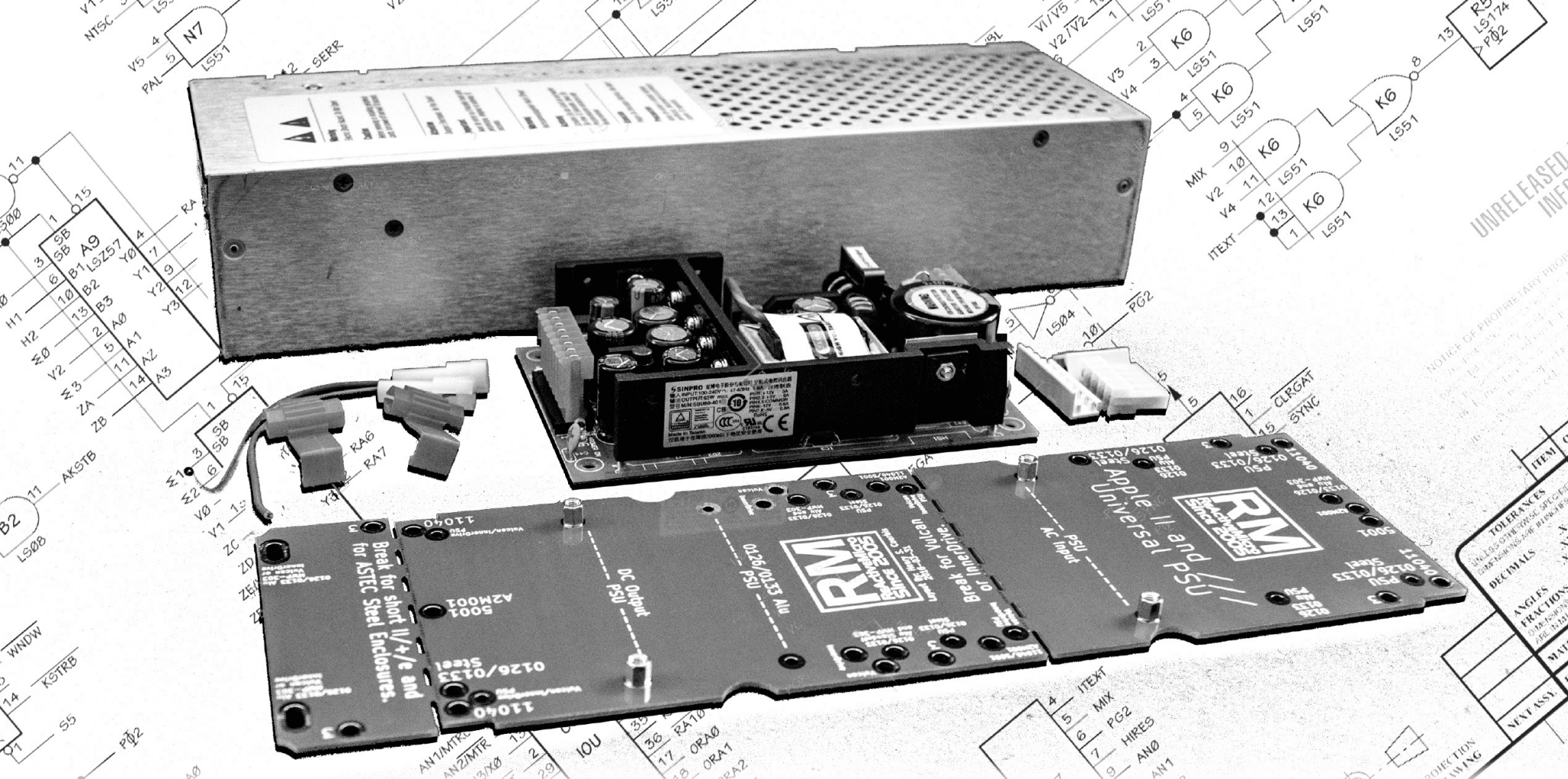
Denis Molony, the author of Apple II Disk Browser, has come up with a nifty tool for Apple II users who want to get to the guts of their floppy disk images in a hurry. Apple II Disk Browser is a tool which allows you to flip through several of your disk images at a time and see the actual contents of those images right down to the sector level.

It also include a very nice disassembler which makes viewing assembly language files all the more enjoyable and educational. The source code for Apple II Disk Browser has also been made available.

Download Apple II Disk Browser from:
<https://github.com/dmolony/diskbrowser/releases>



Diskbusters
computist.applearchives.com



The power supply. Redesigned.

The Universal PSU Kit is a power supply replacement designed with installation by the end user in mind. The primary function is to replace the Apple II power supply PCB, which is 30+ years old (40+ for II and II+), with a new, modern solution while reusing the original power supply enclosure. The kit is very simple, safe to install with basic household tools, and takes about 15 minutes from start to finish.

The Universal PSU Kit is currently available for the Apple II, II+, IIe, IIgs, and Apple III. It also supports power supply upgrades for the Vulcan and Inner Drives. The kit was conceived, created, and designed by ReActiveMicro. Comes standard with a 2 year warranty.

When Do YOU Need a New Power Supply? Good question! Although the answer should be obvious. Is it the original, 30+ year old PSU that came with the Apple? Then you need a new one. It's THAT simple. How much more do you want or expect from a piece of equipment that was only designed to last a few years at most? Do you really believe that Apple designed the power supply to last 30+ years? Most people either don't realize the potential dangers of continuing to use an old PSU or are of the mindset "If it ain't broke, don't fix it." Both can be equally damaging to your beloved Apple II.

Fits all standard Apple II and III Power Supply Enclosures (all parts included, user supplied household tools).

Main features:

- Modern PSU design.
- On-board power LED.
- Constant Voltage design.
- Meets or surpasses all Apple II and III power requirements.
- Fast Power Reset: No more "hung" systems or need for long off periods and waiting before turning back on.
- Interchangeable DC Output Cable for II/+e/gs/III. One PSU can support them all.
- Runs cool/low heat output: operates 50% cooler than the old Apple II PSU.
- Cleaner and more consistent power than the old Apple II PSU.
- Fused AC Over Current/Over Voltage/Surge protection.
- DC Over Voltage protection (active crowbar design) and auto restart.
- All units fully stress-tested/burned-in at 90% loading before shipping.
- Kit is "reversible". You can downgrade your enclosure back to its original state if desired in the future.

Specifications

- Input voltage: 90 to 264 VAC / 47-63 Hz
- High efficiency switching design
- Output: +5V @ 6000MA
+12V @ 3000MA
-12V @ 1000MA
-5V @ 1000MA
- Ripple/Noise: 1% max
- Full load operating temperature: 32° to 122°F (0° to 50°C)
- Standard 2-year warranty
- Replacement DC output cables available

"I recommend ReActiveMicro products wholeheartedly since 2005."

(Of course Henry has a Universal PSU installed)

Henry Courbis, creator of the Universal PSU Kit



REACTIVE
MICRO.COM

Web Store: store.reactivemicro.com

Sales Email: sales@ReActiveMicro.com

Support Email: support@ReActiveMicro.com

Phone: 800-ReActive (732-2848)

Golden Orchard

Apple II CD-ROM

CD-ROM is definitely a hot part of today's home computer industry, and the Apple IIGS market is no exception! With the coming of Sequential Systems' DiscQuest, there now is an extensive library of informational CDs usable on an Apple IIGS, including an encyclopedia. But there has been a shortage of compilation CDs (CDs that contain a vast assortment of freeware and shareware programs, information, and data), until now!

If you have a CD-ROM drive, take a look at all the useful files you will find on Golden Orchard. No longer wait for slow downloads, or mail order shareware! Golden Orchard puts over 600 megabytes of Apple II files at your fingertips. If you don't have a CD-ROM drive, perhaps now is the time to consider purchasing one. The cost of drives has plummeted, and the library of Apple II compatible CDs only continues to grow!

Now is the time to join in the CD-ROM revolution!

Golden Orchard Content Summary

Note: Shareware fees must still be paid to some of the programs' authors if you end up using the programs frequently.

Applications.....	81mb	Music.....	64mb	TrueType Fonts.....	28mb
Demos.....	10mb	MIDI Songs	5mb	Deprotects & Cheats	6mb
Games	27mb	MODs	22mb	Icons	1mb
Graphics & Sound Demos ..	7mb	SoundSmith Songs	16mb	CDAs.....	2mb
Graphics Utilities	5mb	SynthLab Songs.....	12mb	NDAs	2mb
Sound & Music Programs ...	10mb	Sounds	13mb	Inits	1mb
System Utilities	4mb	Apple Software	28mb	Finder Extensions	1mb
Telecommunications	5mb	System Software		Patches & Updates	1mb
Disk Images	145mb	HyperCard IIGS		Text Files	25mb
FTA Software	22mb	AppleWorks.....	10mb	Programming	124mb
Graphics	51mb	BASIC Programs	1mb	From Apple	31mb
3200 Color Pictures	8mb	Stacks	28mb	Data Compression.....	2mb
Animations.....	10mb	HyperCard IIGS	6mb	Assembly Source.....	8mb
APF Pictures	5mb	HyperStudio	22mb	C Source	10mb
GIF Pictures	12mb	Bitmap Fonts	2mb	Utilities	18mb

This is just a partial listing! Golden Orchard is the only CD available that contains over 600 megabytes of files useful to Apple II owners! This CD will pay for itself many times over!

<https://digisoft.callapple.org>

Article Submission Guidelines

Do you have content which you would like to share with the A.P.P.L.E. community?

Call-A.P.P.L.E. is always looking for new and interesting ideas and articles related to Apple, Linux/Unix, technology of general interest, technical or editorial, modern or retro.

All submissions must be original works of the person submitting and free of distribution restrictions. Please email your proposal to:

editor@callapple.org.

By submitting materials to us, you agree to give A.P.P.L.E. the royalty-free right to publish and reuse your submission with your name in any form in any media. We reserve the right to edit, change, or not use anything submitted to us.



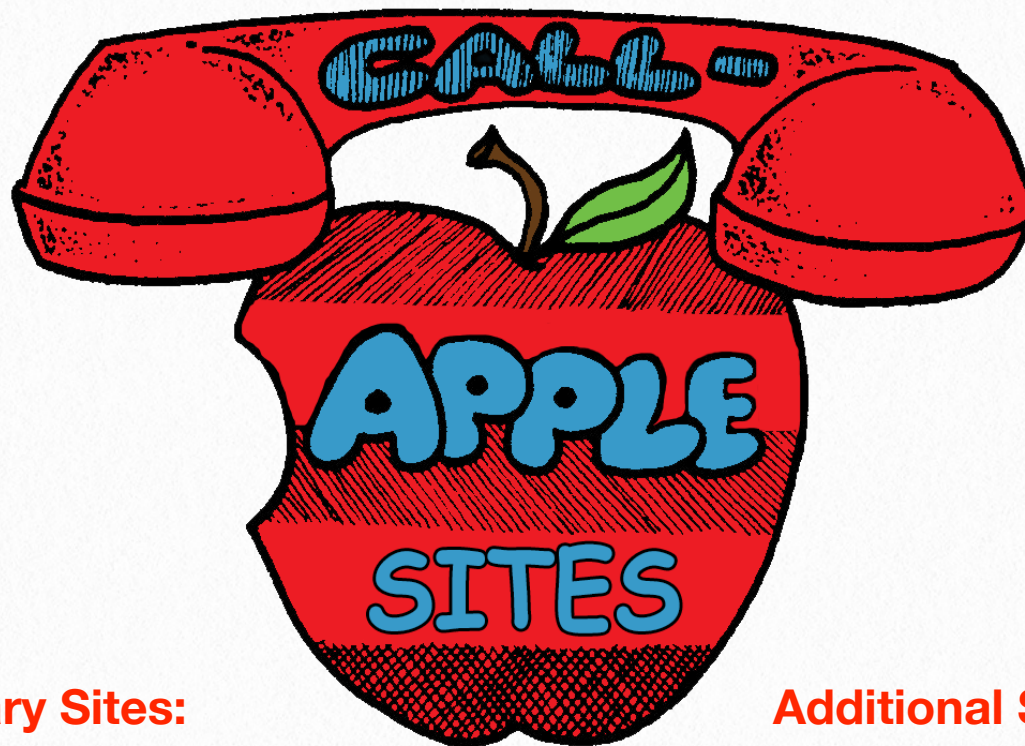
Join A.P.P.L.E. Today!

*Access every legacy PDF issue of
Call-A.P.P.L.E. magazine!*

With your membership, the new, current issues of the magazine are included along with access to many back issues, articles, books, software and discounts for products.

Only \$27.95 Per Year!

callapple.org/members



Primary Sites:

**Apple Pugetsound Program
Library Exchange (A.P.P.L.E.)**
<https://www.callapple.org>

A.P.P.L.E. – New Books
<https://www.callapple.org/books>

Structris
<https://www.structris.com>

Apple Archives
<https://www.applearchives.com>

Virtual Apple][
<https://www.virtualapple.org>

Gamezyte
<https://www.gamezyte.com>

Applied Engineering
<https://ae.applearchives.com>

Beagle Bros Official Repository
<https://beagle.applearchives.com>

Minnesota Educational Computing Consortium
<https://www.mecc.co>

WOZ Speaks
<https://wozspeaks.callapple.org>

GBBS Pro – The Official Archive
<https://gbbs.applearchives.com>

Additional Sites:

Take-1 Movie Site
<https://take1.applearchives.com>

Willamette Apple Connection
<https://wac.callapple.org>

Digisoft Innovations
<https://digisoft.callapple.org>

Australian Apple Review
<https://aar.applearchives.com>

Computist Project
<https://computist.applearchives.com>

A2-FS1 Flight Simulator Site
<https://fs1.applearchives.com>

A2-FS2 Flight Simulator Site
<https://fs2.applearchives.com>

A2-PB1 Night Mission Pinball Site
<https://pb1.applearchives.com>

Ancestorworks Repository
<https://anw.applearchives.com>

The Apple /// Resource
<https://apple3.applearchives.com>

Apple II Hackers
<https://hackers.applearchives.com>

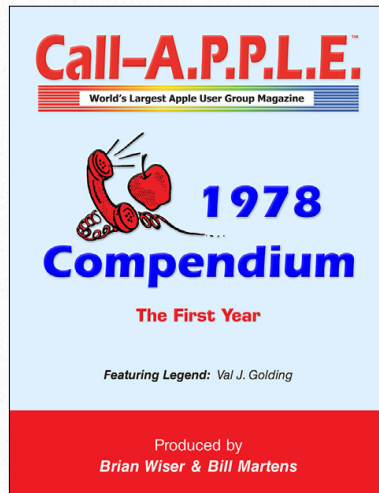
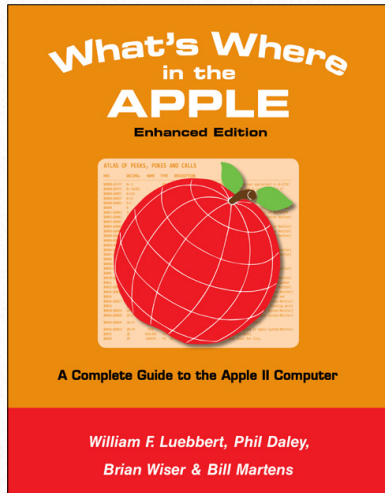
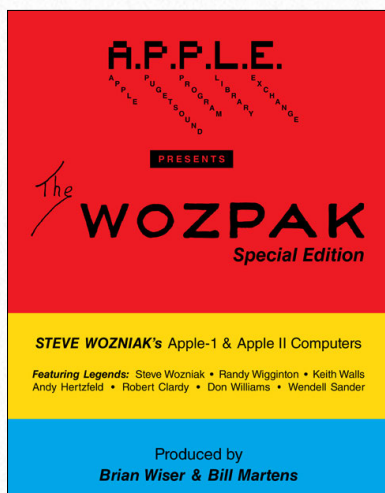
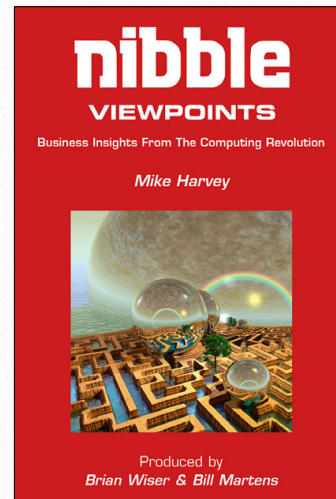
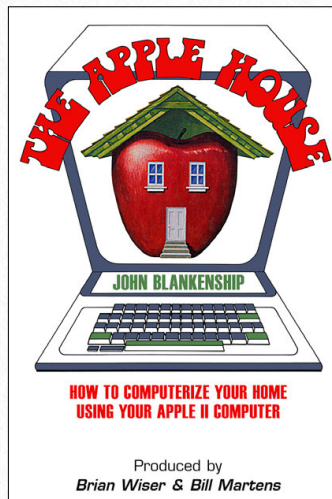
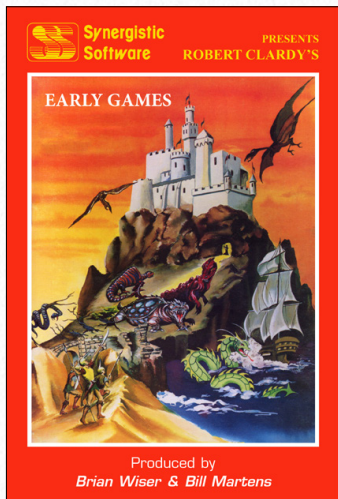
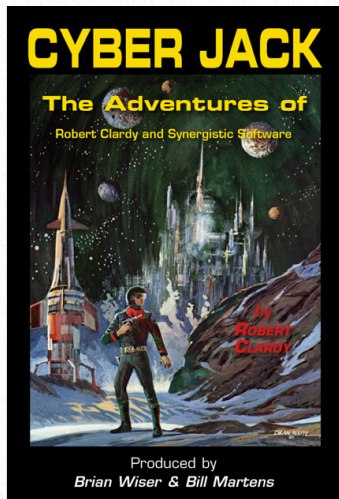
Terry Allen's Apple II Page
<https://apple2.callapple.org>



Call-A.P.P.L.E.™

World's Largest Apple User Group – Since 1978

Join our User Group – Read our Magazine



^ ----- CallAPPLE.org/books ----- ^



Retro & Current NEWS
callapple.org

mecc.co

VIRTUAL apple II
virtualapple.org



beagle.applearchives.com



Apple Archives

The Best Vintage Apple & Mac Websites

applearchives.com



APPLIED ENGINEERING

ae.applearchives.com

^ ----- [Our Web Sites](http://www.callapple.org) ----- ^



www.callapple.org