QSS™
Optimized Systems Software, Inc.
1221B Kentwood Ave., San Jose, CA 95129 ■ (408) 446-3099

THE OSS NEWSLETTER

October, 1984

The OSS Newsletter is published by Optimized Systems Software, Inc. (OSS, Inc.), as a service to its customers. News of new and modified OSS products is featured. Reports of problems, errors, fixes, and enhancements with and for OSS products are given in some detail.

The staff at OSS welcomes submissions for this newsletter. We also take seriously all comments and criticisms of not only the newsletter but of all OSS products. If you have a problem you need resolved, we invite you to write or call us.

Well, this newsletter is something different for OSS: for the first time, we are actually editing and printing our newsletter using our Atari computers.

It may seem strange that a company which is as much into Atari computers as we are would choose to continue to use its venerable (or should that be antiquated) Cromemco and its simple screen editor rather than move up to the advanced features of an Atari-based word processor. The truth is, though, there is a lot to be said for an 80-column display and a 10 megabyte hard disk for storage. Yet even 10 megabyte disks get full after a while, and we had to find an alternative for editing and printing our smaller documents--such as this newsletter.

Truthfully, another factor which kept us away from word processing on the Atari for so long was the lack of a word processor with the features which we wanted. True to our usual form, since we couldn't buy the word processor we wanted, we decided to produce our own. Thus it is that we have the privilege to present to you, for the first time, the visible results of our latest product: A discussion of THE WRITER'S TOOL starts on the next page.

NEWS FLASH!!

Even as we wrote this newsletter, **THE WRITER'S TOOL** was being improved! We now feature: (1) A sophisticated mail/merge capability without the need to buy anything additional. (2) Compatibility between mail/merge and Synapse's **SYNFILE**. Coming soon (as a free update): (3) A menu-driven printer setup program (in addition to having over a dozen printers already set up). (4) Include pictures from Atari's Graphic Tablet, Koala Pad, MicroPainter, etc., right in your text.

THE WRITER'S TOOL

The Writer's Tool is, quite simply, what we believe to be the finest word processor available for Atari computers. Our advertisements, which you will see soon in several magazines, call it the "most natural" word processor. What does that mean to you, as a user?

How about an example? Suppose you have used the editor in The Writer's Tool to prepare a few paragraphs (much like the ones used in this newsletter). Now maybe you haven't used all the nifty features, such as underline, boldface, double wide, etc. But, still, you want to print out a nice looking, evenly spaced document. Only The Writer's Tool gives you three levels of control over the printed format.

First, if you select the **PRINT** menu (simply push the **OPTION** button, see a small main menu, and push P), you will be presented by a set of system print-out defaults for such things as page length, top and bottom margins, left and right margins, etc. For a quick yet good-looking printed page, simply push P again and in a few seconds (depending on how fast your printer is) you will have the printed page.

Suppose, after looking at the printed result, you decide the document would look neater with wider left and right margins. No problem. Again, from the **PRINT** menu, you can easily ask to change the format of the document (a handful of keystrokes are all that are necessary, and the menu helps remind you which to use). Another push of the P key and another page is printed.

Now, not surprisingly, the formats established by the **PRINT** menu apply to the entire document. If you need finer control than that (suppose, for example, you want to indent a particular paragraph for emphasis), you simply imbed format commands in your document. The best part, though, is that the format commands used in the document are the same as those available in the **PRINT** menu!

Rather than detail some of the features of The Writer's Tool which are apparent if you simply examine how this newsletter is printed, we would like to discuss two of the least obvious yet most important features: the manual and the support.

The manual is, in our not overly humble opinion, the best of its kind. It comes in one of our by now infamous yellow binders and is actually two manuals in one. The tutorial is 89 pages of well organized, well presented material. Anyone willing to work through the tutorial will seldom need more than an occasional peek at the reference card (included, of course). But, if you are used to editors and word processors, you may wish to get down to the nitty-gritty by reading the 56 page reference manual. For example, you might look there if you wanted to find out which features of your particular printer are usable from The Writer's Tool.

Naturally, there are a few useful appendices and (wonder of wonders) a very well thought-out index. To summarize, then, we believe that even a relative novice will find himself or herself comfortably using The Writer's Tool after just a couple of hours of reading and practice. And only a very little bit more effort is needed to take advantage of almost all of the features of this powerful word processor.

You undoubtedly noted that we alluded to support for several printers. It's true: The Writer's Tool understands several different makes and models of printers and supports the best features of each. This newsletter, for example, is being printed on a ProWriter (from C. Itoh). This places 5 different font sizes and 5 print modifiers at our disposal. Some simpler printers may have fewer features available, but some inexpensive printers have even more. For example, The Writer's Tool was used to edit and print its own manuals. An inexpensive Comrex CR-II daisywheel printer produced such beautiful print quality that many have thought the manuals were professionally typeset.

Well, why belabor the point. The suggested retail price of The Writer's Tool is $129.95 (but look for the special deal offered herein). If that seems steep, we think you should compare it to both AtariWriter (at $99, but with printer drivers adding $30 more) and to much more expensive products available for other computers. And trust OSS to give you some value for your money: because we package The Writer's Tool partly on disk and partly in an OSS SuperCartridge, you gain valuable memory: There's no real limit on the size of a document, and you can edit up to two more text pages at a time (as compared to AtariWriter).

Should you rush out and buy The Writer's Tool? We offer you some mildly prejudiced advice on that subject. Certainly, if you do not yet have a word processor--or if you have one of the very simple, introductory ones, such as Bank Street Writer--then we think you should give our product very serious consideration. If you do a lot of writing, and flexibility is important to you, then we would also suggest a careful perusal of The Writer's Tool. But, to be as fair as we can be, we must admit that if you do little writing yet already have one of the better word processors (e.g., AtariWriter or Letter Perfect) which works with your printer, then it may not yet be time to trade up.

Finally, let us tell you what The Writer's Tool is not. It is not WordStar nor is it comparable to that product or its like. We simply have neither the disk space nor speed to be able to support the intensive HELP menus which products for more expensive machines can offer (the same holds true of 80-column editing, of course). Also, as of this writing we do not have a spelling checker (though from reports just in it appears that the APX spelling checker and/or Spell Wizard work fine with Writer's Tool files). So our new word processor may not be all things for all people, but (as this newsletter demonstrates) an Atari computer equipped with The Writer's Tool is capable of some pretty impressive quality. And it is available now.

THE NEW OSS TOOLKITS

By now, you have probably heard about the new series of "TOOLKITS" which we at OSS have introduced. As this is written, we are already shipping **THE MAC/65 TOOLKIT** and **THE ACTION! TOOLKIT**. We hope to be shipping **THE BASIC XL TOOLKIT** by October 1st, but it could slip to the 15th (documentation, as usual, is holding it up).

But just what is a **TOOLKIT**? We thought you'd never ask! (But you'll have to go to the next page for the answer.)

What IS a ToolKit? (continued)

A **TOOLKIT** is--ta da!--a set of programming tools designed to help you take advantage of one of our OSS programming languages. The "tools" may consist of subroutines, demonstration programs, examples of programming techniques, etc. The content of a particular **TOOLKIT** disk vary, depending on what language it will be used with.

Before we go into just a little detail about the individual **TOOLKITS**, we should make a comment or two about for whom they are designed. Certainly, the newcomer to one of these OSS languages could use help getting started. With the exception of BASIC XL (and, now, THE WRITER'S TOOL), our manuals are definitely not tutorials. Any of our **TOOLKITS** will help introduce its language since each contains full source code for some functional, well commented programs.

But to justify a name like **TOOLKIT**, we should offer more than just a beginner's tutorial. And we think we do. Consider this: with few exceptions, the software routines you will find in a **TOOLKIT** are ones which any intermediate to advanced programmer could produce, given time and experience. Why, then, should such a programmer pay OSS his/her hard earned cash instead?

The answer lies simply in how much you value your time: We would guess that it would take at **least** a day or two of work to produce a debugged version of even a single set of **TOOLKIT** routines (and each kit includes several sets of routines). Using round numbers, that's 16 hours for a maximum of $40 worth of software. Do you really feel your time is worth only $2.50 per hour? (And, since most users will find many things to use in a single **TOOLKIT**, the actual figure may be between 50 cents and a dollar an hour.)

So much for the economic justification. "What about the actual contents of these **TOOLKITS**?" you ask. "How do I know there are any routines of interest to me, even so?" Glad you asked.

THE MAC/65 TOOLKIT

As is appropriate for use with a macro-assembler, this **TOOLKIT** includes both subroutines and macros to access them. Along with a couple of programs which demonstrate usage, there are three separate libraries divided into the seven categories which we will describe briefly here:

- **Input/Output** -- In the spirit of the I/O macros listed in the MAC/65 manual, but with some useful extensions.
- **Integer Math** -- Double byte (16-bit) math calculations made easier.
- **Graphics** -- Essentially the equivalent of BASIC's various graphics statements.
- **Program Control** -- Looping, conditional branching ("IF...THEN"), and error trapping.
- **Miscellaneous Aids** -- Peeks, pokes, moves, etc., in several flavors.
- **Player/Missile Graphics** -- More help here than we can describe! Movement during vertical blank, horizontal and vertical movement, collision detection, and more.
- **Smooth Scrolling** -- Scroll in any graphics mode, any size memory map, horizontally and vertically with speed control, easy joystick interface. More? Of course!

THE ACTION! TOOLKIT

In many ways, this is simply an upgrade on a previous OSS product, the Action! Programmer's Aid Disk (usually just called the "PAD"). In fact, there are only a handful of differences between the two. We recommend that owners of the PAD peruse the following list of brief descriptors before deciding if it is worth upgrading. For the rest of you, what are you waiting for? It's hard to believe that this package isn't worth twice the price to anyone even halfway serious about learning and using Action!

Memory Allocation -- Very useful for building linked lists, dynamic arrays, etc.

Input/Output -- Action!'s built-in I/O is very good. These routines make it nearly great.

Player/Missile Graphics -- Makes control of many PMG features almost easy!

Real Numbers -- Action! still won't directly support real (floating point) numbers, and we don't recommend using it for many purposes as a result, but the PROCedures and FUNCtions in this library allow a reasonable (if limited) use of real numbers in your programs.

QuickSort -- Sort arrays of BYTEs, CARDinals, INTegers, and even strings using the QuickSort algorithm (one of the fastest methods known).

Turtle Graphics -- Maybe these should be called Hare graphics. They work like Logo's turtle graphics, but so fast you may want to put some artificial delays in.

Miscellany -- Circle drawing, easy joystick reading, console key handling, character testing and manipulation, and more.

Games and Demonstrations -- Some demos are strictly business, to show how to use library routines. Others are fascinating displays of programming virtuosity. The games? Gem and Snails' Trails are fun for two or more players. Warp Attack, for one player, would have sold for $39 all by itself a year or two ago.

SPECIAL OFFER -- Owners of a Programmers' Aid Disk who decide the update is worthwhile may purchase The Action! ToolKit for $15 (shipping included, but plus 6% tax in California) only upon return of the original PAD disk with a check or money order.

THE BASIC XL TOOLKIT

Again, to some degree this package is a resurrection of an older OSS product, the BASIC A+ demonstration disk. However, we believe the THE BASIC XL TOOLKIT will be valuable to many people.

First and foremost, this TOOLKIT includes a self-relocatable version of the BASIC XL RunTime package. This means that you can write programs in BASIC XL and give them to your friends, user groups, and (surprise!) even sell them commercially at no additional cost! All you have to do is agree to the terms of a very reasonable license agreement and send us a signed copy.

The fact that RunTime BASIC XL is self-relocatable means that you can use it with virtually any DOS, with the 850 RS-232 drivers, any memory configuration of 32K bytes or more, etc. So use it and enjoy.

Also, for the first time, there will be some extended BASIC XL statements. You may or may not have noticed that we have claimed that the BASIC XL cartridge is extensible in RAM. Well, the claim is true, and for the first time we will be showing some extended statements. What will they be? As this is written, we are working on LOCAL variables, but the final decision as to exactly which statements will appear is going to be made at the last moment. So it will be a surprise, but we promise a pleasant one.

Aside from all this, **THE BASIC XL TOOLKIT** primarily consists of programs written in BASIC XL. A brief description of each of several of them follows:

Freud -- This is an oldie-but-goodie. Let Dr. Freud psycho-analyze you. Strictly for laughs.

Pico Adventure -- "Pico" as in smaller than "micro". There are only 16 rooms in this adventure, but all the structure is there for you to add to, modify, or whatever. A good basic framework for writing your own adventures.

LEM -- The classic Lunar Lander program...except that this one is strictly graphic. An impressive yet small-scale demo of what you can do with BASIC XL's player/missile graphics, etc.

Square Root -- Simply a demonstration program. But nice because it shows you how mathematical algorithms are implemented. Uses the same method that the version built into BASIC uses.

Black Book -- A very simple data base program. Keeps track of just phone numbers and names. Its main purpose is to show you how to implement random access and/or keyed access files in BASIC XL. A good learning tool which could even be modified (by you) into a useful mailing list program, etc.

And more -- So much stuff on this disk that it may take both sides of the disk! You can't afford to be without it.

THE PRINT TOOL

Do you have a favorite screen or text editor? For example, have you considered using your **ACTION!** cartridge to write articles, books, or whatever? Have you been put off by the fact that the ACTION! editor does no print formatting, handles only a limited amount of text, and is (in short) a great screen editor but not at all a word processor? Would you be interested in changing all that for $39?

The Print Tool is by far and away the most powerful print formatter available for Atari computers. It is over 95% compatible with **RUNOFF** programs found on much, much larger DEC computers. Imagine being able to easily create a **book-length** document with multiple footnotes, a table of contents, an outline, an index, footers, headers, and more!

With all this, why do we sell it for only $39? Because you have to supply the screen editor (e.g., by using your ACTION! cartridge) and because, although you can edit for The Print Tool with almost any word processor, it uses complex, non-compatible printer formatting commands. (YET! If you might be interested in a version compatible with your word processor, please write us!)

Summary: If you need the ultimate in text formatting on an Atari, you won't find a better deal anywhere than **The Print Tool.**

P.S.: Ask about our Action! editor modification which gives you limited word wrap capabilities!

How to be RICH AND FAMOU$$$

Well, maybe a little bit famous at least. You may have been noted from other announcements in this newsletter that we at OSS are making an effort to encourage more widespread use of our products. For example, we have decided to eliminate charges for the RunTime version of BASIC XL.

In a related move, we have decided to allow all Atari-related magazines that publish disk or cassette versions (including **ROM, ANALOG, ANTIC,** and others if they ask) to distribute runnable programs written in either ACTION! or BASIC XL.

What does that mean to you? Now you can write programs in ACTION! or BASIC XL, submit them to your favorite magazine, and (presuming your article is accepted) see your program both printed and distributed in runnable form to thousands of people. And why should you write such an article? Well, the magazines usually pay a nominal per page rate for articles and programs, and we at OSS are going to sweeten that rate just a bit:

If you write an article which either uses an OSS product or shows how to do so, OSS will pay you an amount equal to what the magazine pays you.

Restrictions: (1) Your article must be accepted and paid for by a recognized national magazine (any magazine with 20,000 paid circulation automatically qualifies--ask about others). (2) The article cannot be construed to be a review. We do not want to be accused of influencing the outcomes of reviews. (3) We pay only for the first printing of an article, not for any reprints. Exception: an article which appears unpaid in a user group publication which is later paid for by a national magazine. (4) We match payments only for the publication of the article, not for any royalties, etc., which result from sale of programs based in whole or in part on an article. (5) No one may earn more than $500 per year under this plan. (6) We reserve the right to make other future restrictions if, in our sole opinion, they become necessary.

Also, since our license agreements already allow disk publication of ACTION! and BASIC XL programs by user groups (RunTime code only, remember!), those of you not ready to write for national magazines may write for your local users' group newsletter. And, even though most newsletters don't pay for articles, we will! A minimum of **$25.** Again, though, there are some rules:

(1) through (6), same as above, except that there is no restriction on the circulation of the newsletter. (7) The newsletter must be a bonafide production of a recognized user's group. Generally, the newsletter must be at least four pages in length and have articles, etc., by at least three different people. (8) The article must be a minimum of a page (8 by 11 inches) in length, or equivalent. (9) No more than two payments per year will be made to any one individual. (10) The article and any accompanying program(s) must be in the public domain. OSS must have the right to reprint either the article or program or both in its own newsletter. If OSS chooses to so reprint, a second payment (also of a minimum of $25) will be made. (These reprint payments are not subject to rule 9.)

You must enter to win! You have to send a copy of the published article to us before we can make payment.

(More about **$$$** on next page.)

Exceptions to any and all of the above rules are at the sole discretion of OSS. Similarly, awards greater than $25 for articles in user group publications will be made only when, in our sole judgment, the quality of an article is outstanding enough to warrant it.

Finally, in addition to all the above, OSS will annually make a cash award of **$200** for the best printed article in either a commercial or user publication. The decision as to what constitutes "best" shall be (once again, what else) at the sole discretion of OSS. The "contest" for this award will run from July through June (publication dates) and the award will be made by August 31st.

Is that enough incentive? We certainly hope so. Start writing!

YET ANOTHER DOS XL?

Yeah, why not. Truthfully, the whole reason for bringing out this "new" version of DOS XL is to consolidate several products into one. DOS XL (and this is version 2.30p) now includes all the following:

Mosaic RamDisk Support -- Use a pair of Mosaic 64K RAM cards to emulate a disk drive for fast copies, etc. (Atari 800 only)

Axlon RamDisk Support -- Use the Axlon 128K RamDisk with DOS XL. (Atari 800 only)

BIT-3 Support -- Allows you to use the BIT-3 80-column card with BASIC XL and MAC/65 SuperCartridges. (Your Action! programs may use the BIT-3 card, but the Action! editor will not support it.) (Also Atari 800 only)

BUG/65 -- Actually, we've been selling a package called "BUG/65 with DOS XL" for some time now. So now we have made a radical change: We're selling "DOS XL with BUG/65". Big deal? Well, maybe not, especially if you aren't writing assembly language programs, but it won't hurt. And BUG/65 is completely relocatable and uses no zero page space, so you can even use it with BASIC to write and/or debug small subroutines. See instructions for using BUG/65 with BASIC in another part of this newsletter.

SPECIAL NOTE: Because there is not enough room on a single density disk for all these goodies, the RamDisk support (both versions) is shipped only on the double density side of the DOS XL disk. If you do not have a double density drive, we will supply a separate single density disk with the needed files for only an additional $5 media and shipping charge. Also, if you have a Mosaic system, you may obtain a special, already-configured Mosaic RamDisk DOS XL directly from Mosaic. Contact Mosaic for pricing and delivery.

Finally: If you are reading this, the chances are excellent that you already own DOS XL or OS/A+. This latest version of DOS XL will probably only be of interest to you if you own version 2.20 or earlier. It is available as an update, of course (see price sheet).

THE REST OF THIS NEWSLETTER

The rest of this newsletter is devoted to a discussion of bugs in, foibles of, improvements to, tricks with, and who knows what else about various OSS products. We welcome contributions to this section.

The products are discussed one at a time, so you can skip the sections which don't apply to you. But you might read them anyway, just to see what level of support each has.

BUG AND BASIC

Among debug programs available for Atari computers, BUG/65 has a few unique features. First and foremost is its intrinsic relocatability: you can place it almost anyplace in memory (or let it place itself at LOMEM automatically). Second but just as significant is the fact that it uses absolutely no zero page memory. The combination of these two features makes it ideal for a variety of uses. We will describe one: writing and debugging BASIC USR() subroutines with BUG/65.

The procedure we are about to describe is not for the faint of heart, those hesitant about experimenting with their machine. One slip could easily wipe out all programs in memory, so be sure and back yourself up carefully and often. We describe the process step by step:

1. Remove all cartridges and boot your DOS XL disk with BUG/65 on it (hold down the OPTION button if using an 800 XL).
2. From the CP's D1: prompt, type BUG65, thus loading and running BUG/65.
3. Note the contents of LOMEM (location $2E7); write them down. Use either the BUG/65 W# command or the SAVE command from CP to save a new, non-relocatable version of BUG/65 to disk. Remember, the start address to save is $200 higher than what was in LOMEM and the end address is $2000 higher. (Thus, if $2E7 contained $2100, a likely value, you could use the CP command "SAVE BUG2100.COM 2300 4100" to obtain a new, custom version of BUG/65.)
4. Turn off the power and re-boot with BASIC (or BASIC XL) present. From CP, give a command to load and run your modified BUG/65 (e.g., "BUG2100" if you used our example above).
5. Once in BUG/65, change the contents of LOMEM to be <u>at least</u> $2000 higher than the value you noted earlier. (We recommend changing LOMEM upwards by $2400 at first. You can refine this value later.) This effectively protects BUG's memory from being used by BASIC.
6. Use BUG to load or mini-assemble your subroutine. Presuming that you wish to single-step, etc., through the routine, you should place a breakpoint at the beginning of the routine and exit to BASIC. If using Atari BASIC, you can do this with a command such as "G A000 @4100" (if your routine was at $4100).
7. Now, when you call that routine from BASIC (via a USR usage), BUG/65 will get control at the breakpoint location.

You will probably have to experiment quite a bit to discover how best to make BUG/65 work for you, but it certainly beats POKEing in routines from BASIC!

A "FIX" FOR BUG/65

If you have any version of BUG/65 other than the latest one (the one now supplied with DOS XL--it has a manual with a yellow cover), this info is for you:

As you are probably aware, BUG/65 supports only one breakpoint (via the "G" command). Sometimes, though, you would like to set another breakpoint. Or you might simply like it better if BUG would handle code which jumped to zeroed memory (zeroed memory appears to your 6502 CPU as a bunch of BRK instructions). BUG/65, though, is too smart for its own good.

The OSS All-purpose Center Pages

This sheet of paper (4 pages) contains these general comments, an order form, a market survey/ contest entry, & contest rules.

The sheet should be pulled out and used as an entry form in our contests and/or for orders.

Conte$t$!!

We like to encourage customer involvement, so we are going to run a few contests in this newsletter. This is your opportunity to get some of our new products without putting out any of your hard earned ca$h. The details about these contests appear on the reverse side of this page.

Fast Service Order Form

If you have a local dealer who has given you good service with our products, please continue to patronize him!

However, we recognize that many dealers do not stock OSS products, so for your convenience we have included an order form which you may simply fill out and send in. Notice that there are a couple of special deals offered here. Save money by ordering before December 15, 1984.

If you are in a hurry, you will be pleased to know that we are now guaranteeing to ship any order within one week from when we receive it. If we cannot ship within that week, we will give you a 5% discount and pay shipping costs via air. Exceptions: orders placed on hold awaiting clearance of payment and orders for the BASIC XL ToolKit or THE PRINT TOOL.

IMPORTANT: Even if you are not ordering, you must fill in your name and address on the order form if you wish to enter the market survey contest!

An OSS User Survey and Conte$t

First of all, and easiest to enter, is our OSS user market survey. If you are like us, you probably hate filling out survey forms. After all, why should you provide us with all that free information?

Well, we have a surprise. We are going to give away three prizes in a random drawing from all returned, completed survey forms.

FIRST PRIZE -- $100 in OSS merchandise of your choice.
SECOND PRIZE -- $50 in OSS merchandise of your choice.
THIRD PRIZE -- $25 in OSS merchandise of your choice.

True, these aren't exactly fortunes to either you or us, but they seem appropriate for 5 minutes of your time and a 20 cent stamp. Besides, we will publish some of the results from this survey, so you can find out what your fellow Wizards of OSS (groan!) are thinking.

The usual restrictions on public contests have to apply: No purchase necessary to win. One entry per person. Additional entry forms available on request (SASE, please). Survey form must be filled out completely except where otherwise noted. Decision of judges is final. Odds of winning determined by number of entries. Employees of OSS ineligible. All entries must be received by November 20, 1984.
BONUS: If we receive more than 2000 responses, we'll award extra prizes!

Programming Contest

This contest is for all of you who want to show off your programming talents. We will award at least three prizes of $100 worth of OSS merchandise each. The rules are fairly simple:

1. Your program must be written using Action!, BASIC XL, or MAC/65. The best program (as defined below) in each language will receive a prize.
2. Your program may not exceed 1K bytes (1024 bytes) in size. Size is as SAVEd from BASIC XL or object code size for ACTION! and MAC/65; routines in ACTION! cartridge's built-in libraries do not count against you.
3. Because of the size limitation imposed, we expect that most programs submitted will be games. If, in the sole opinion of the judges, a non-game program of exceptional merit is entered, a separate and additional prize (equal to the other awards) may be made.
4. All programs submitted must be the original work of the author(s) and must be placed in the public domain. Neither the author nor OSS will claim copyright or any other rights.
5. Entries will be judged on user appeal (50%), originality (10%), meeting size restriction (10%), and use of the language (30%). The latter implies, for example, that assembly language routines in BASIC XL or ACTION! will be looked on with disfavor.
6. Programs may not have been previously published anywhere. OSS reserves the right to publish any program submitted (winner or not) in its newsletter or other public domain media, in which case the author will receive the standard non-commercial publishing award of at least $25. Authors may submit programs to other publications after conclusion of contest.
7. All entries must be postmarked not later than December 31, 1984, and received not later than January 7, 1984.
8. Miscellaneous: No purchase necessary. No special form needed to enter. Enter as often as you wish. Programs must be submitted on disk or tape, which will not be returned unless a self-addressed mailer and postage is enclosed. Decision of judges is final. In case of ties, duplicate prizes will be awarded. Employees of and authors published commercially by OSS are not eligible. Winners will have to sign a form agreeing to the conditions imposed above.

ORDER FORM

[Remember: fill out this top portion if you want to enter our market survey contest.]

Name: _____

Address: _____

_____

City: _____

Zip/postal code: _____ Country, etc.: _____

Phone number (optional, but it may help): _____

==========

To place an order, please fill in the rest of this form.

==========

How will you pay for this order? [ ] Check or Money Order Enclosed
[ ] COD (via UPS only) or charge via [ ] VISA [ ] MasterCard

If charge, card #: _____ Expires: _____
How should we ship the order? [ ]Best Way [ ]UPS [ ]UPS 2-Day Air
[ ]US Mail [ ]US Air Mail [ ]Other: _____

[Just check the items you want]

Special Prices--until December 15, 1984, only:

The Writer's Tool	$99	[ ]
BASIC XL with ToolKit	$125	[ ]

Not special prices, but Fast Delivery (unless marked with *):

OSS SuperCartridges

BASIC XL	$99	[ ]
ACTION!	$99	[ ]
MAC/65	$99	[ ]

Disk Products

* BASIC XL ToolKit	$39	[ ]
ACTION! ToolKit	$39	[ ]
MAC/65 ToolKit	$39	[ ]
* The Print Tool	$39	[ ]
DOS XL with BUG/65	$39	[ ]
C/65	$80	[ ]
ACTION! RunTime Package	$30	[ ]

Manuals Only, $15 Each
($10 applies toward purchase of full product)

BASIC XL Manual..[ ] ACTION! Manual..[ ] MAC/65 Manual..[ ]

Updates on Selected Products
(Inquire about other updates available.)

OS/A+ or DOS XL to latest DOS XL ..... $20 [ ]
ACTION! Programmer's Aid Disk to ToolKit ..... $15 [ ]

==========

Total of Order: $_____

6% Tax (CA residents only): $_____

Estimated Shipping Charges: $_____
(free in U.S. if check/m.o. enclosed)

Total Enclosed or to be Charged: $_____

OSS MARKET SURVEY

Remember, fill this out and return it by November 15, 1984, to be eligible to win one of three prizes (see last page for details).

1. Why did you first buy one of our products?
   [ ]Advertising [ ]Magazine review [ ]Newsletter review [ ]Saw demo
   [ ]Recommended by friend [ ]Recommended by dealer [ ]Came with disk
   [ ]Other (specify):

2. What was your first OSS product?
   [ ]BASIC XL or A+ [ ]ACTION! [ ]MAC/65 [ ]C/65 [ ]DOS XL or OS/A+
   [ ]any ToolKit [ ]Writer's Tool [ ]other:

3. Please rate that first product in terms of value for the money:
   (poor) 1 2 3 4 5 6 7 8 9 10 (excellent)

4. Since then, which other OSS products have you purchased (if any)?
   [ ]BASIC XL or A+ [ ]ACTION! [ ]MAC/65 [ ]C/65 [ ]DOS XL or OS/A+
   [ ]any ToolKit [ ]Writer's Tool [ ]other:

5. Tell us about your hardware, please.
   My computer [ ]400 [ ]800 [ ]1200xl [ ]600xl [ ]800xl
   Has [ ]16K [ ]24K [ ]32K [ ]40K [ ]48K [ ]52K [ ]64K [ ]) 64K of RAM
   With [ ]a disk drive [ ]a second disk drive [ ]3 or more drives
   And [ ]a printer [ ]850 [ ]other printer interface [ ]modem [ ] ramdisk
   [ ]80 column board [ ]omnimon/omniview, etc. [ ]speech synthesizer
   [ ]other (specify):

6. Brand(s) of disk drive(s): [ ]Percom [ ]Indus [ ]Trak [ ]Rana [ ]Amdek
   [ ]Atari 810 [ ]Atari 1050 [ ]SWP [ ]Other (specify):

7. Type/brand of printer: [ ]MX80 or compatible [ ]MX80 G+ or compatible
   [ ]Gemini 10x/15x [ ]FX80 or RX80 [ ]Prowriter [ ]NEC 8023 [ ]Other NEC
   [ ]Atari 825 [ ]Atari 1025 [ ]Atari 1027 [ ]Comwriter [ ]Okidata 82 or 92
   [ ]Centronics [ ]Axiom [ ]Inkjet [ ]Other daisy wheel [ ]Other, matrix
   Give brand & model if not complete above:

8. If you are considering another computer, what brand and model?
   [ ]Atari xl-series [ ]Atari, future machine [ ]Atari, other:
   [ ]Apple IIe [ ]Apple IIc [ ]Apple MacIntosh [ ]Apple, other:
   [ ]Commodore 64 [ ]Commodore plus 4 [ ]Commodore, other:
   [ ]IBM PCjr [ ]IBM PC or compatible [ ]IBM-XT or compatible
   [ ]CP/M compatible [ ]MSDOS compatible [ ]UNIX compatible
   Give brand & model if not complete above:

9. Where and how much do you use any computer?
   [ ]At home, _____ hours per week [ ]At office, _____ hours per week

10. What is your favorite piece of software, from any manufacturer (do not include games):

11. What task are you now doing by hand which you would use a computer for if you could find the right software/hardware?

12. Rate our newsletters for both content and interest, please.
    CONTENT: (poor) 1 2 3 4 5 6 7 8 9 10 (excellent)
    INTEREST: (none) 1 2 3 4 5 6 7 8 9 10 (oodles)

13. We are considering a policy of sending only one or two copies of this newsletter free for each product purchased. Would you be willing to pay a postage and printing fee to continue receiving it?
    [ ]NO [ ]Yes, up to $1/issue [ ] Yes, $2, if bigger with more articles

14. Who is your favorite <u>local</u> dealer? :

When BUG gets control as the result of a BRK instruction being executed, it actually checks to see if the BRK instruction is one which it placed (as a result of a "G" command). If not, it assumes that it is a user breakpoint, **to be handled by a user routine!** Guess what? Most of us don't go around adding user BRK handlers to Atari's OS, so the machine goes off into never-never-land.

Quick fix (again, not for the faint of heart): After loading BUG/65 from CP, perform the following actions:

1. Display the contents of location $2E7 (LOMEM).
2. Use the "H" command to add $17EE to those contents. (Example: if $2E7 contains 2200, use "H 2200 17EE" to find the sum.) Call this value ADDSUM.
3. Display the memory at ADDSUM (via the "D" command. You should see the following bytes (if not, stop now):
   39EE E9 02 4D ...
   ("39EE" is arbitrary--value of ADDSUM if LOMEM is at $2200.)
4. Use BUG/65 commands as follows:
   ```
   Z 600
   LDA #B0
   STA addsum (use actual address instead of name!)
   LDA #1E
   STA addsum+1 (again, use an actual address)
   BRK
   (hit the BREAK key to exit miniassembler)
   G 600
   ```

If you did everything correctly, BUG/65 should allow a "USER RUN" and then print a breakpoint message for the BRK at $60A. If it didn't work, you may have an already-fixed version of BUG/65, so ignore all this.

All About Action!

Well, maybe not all about Action!, but at least quite a bit. First, we would like to remind you (and tell those of you who don't know) that several versions of Action! exist. The latest version is 3.6 (the byte at location $B000 is the version number--use ?$B000 from the monitor to see it). We at OSS do keep a list of known bugs by version number, and you are welcome to a copy of it. Simply send a self-addressed, stamped envelope to us. On the outside of <u>your</u> envelope (the one you send <u>to</u> us) make the notation "**ACTION BUGS**".

In this issue of the OSS newsletter, we will report some new bugs (among other things). Where space permits (and/or where the bug is significant to many users), we will print a fix (if it exists). The notation "see bug sheet" means that for more information and a possible fix you should send for the list mentioned in the last paragraph.

Before getting to the bad stuff (the bugs), here are some goodies about Action! which we would like to pass on to you:

Busy Bulletin Board Brings Big Benefits

In order to speed the distribution of ACTION!-related information, Clinton Parker (author extrordinaire of ACTION!) has set up a bulletin board system. His company, Action Computer Services, is located in Rochester, New York, and you may try calling his bulletin board at

(716) 235-3394

Clinton welcomes comments, suggestions, and praise. Complaints and their perpetrators will be ignored. That is, of course, our joke. But please, if you must offer criticism, keep the remarks constructive. And remember that OSS is responsible for distribution, etc. Clinton is the author and is (we feel) generous in making himself so accessible.

Tips on Temps

The magazine article titled "Lights, Camera, Action!" (by Dave Plotkin) which appeared in the July issue of ANTIC featured a set of routines to facilitate writing Action!-based interrupt handlers.

The article gave the listings for two routines (more properly, two DEFINEs) named "SaveTemps" and "GetTemps". These routines are adequate only if no math beyond addition and subtraction is performed in the interrupt servicer. If you are seriously into writing such interrupt handling routines, you will find the improved version on latest BUG SHEET.

BUGS IN THE ACTION! CARTRIDGES

We list these bugs separately from those in the RunTime library and/or the PAD disk or ToolKit.

1. **OFFSETS** -- Using a TYPE declaration will generate a spurious error whenever the code offset (contents of location $B5) is non-zero.
   Affects: All versions of the cartridge to date. (Presumably only noticed if using RunTime disk, though.)
   Fix: Make all TYPE declarations before changing the code offset.

   Example:

```
; Beginning of program -- first declare TYPEs
TYPE IOCB = [BYTE Id, Devnum, Command, Status]
; Then, if desired, change offset
SET $B5 = $1000 ; example: a code offset of 4096
```

2. **OFFSETS** -- Using a code offset greater than $7FFF (i.e., a negative offset, if you consider it to be of type INT) causes the compiler to generate improper code.
   AFFECTS: All versions, especially when used with the RunTime disk.
   FIX: No direct fix, but the BUG SHEET now includes a relocater program (which is also usable with assembly language).

3. **ATARI DOS** -- Exiting to Atari DOS from Action! can cause a system crash if DUP.SYS is not present on the disk in drive 1.
   AFFECTS: All versions, but only when used with Atari DOS.
   FIX: Use DOS XL (or be careful when exiting to DOS).

4. **ARRAYS AND ELSEIF** -- We have just learned that there is a relatively obscure bug in Action! related to the use of ELSEIF. In particular, statements similar to the form

```
ELSEIF ia(c) = 0 THEN ...
```

   (where ia is an INTEGER ARRAY and c is a CARD) produce incorrect code.
   AFFECTS -- All versions
   FIX -- There is no direct fix at this time. The best way around the problem seems to be to code something like this:

```
i = ia(c) ; i is an INTEGER
...
ELSEIF i=0 THEN ...
```

   This works properly. For more details, see the BUG Sheet.

Bugs in the Action! RunTime Library

Newcomers to Action! may not be aware that the RunTime System Disk is available (see price sheet). With this disk, you can write programs for your friends, your user group, etc.

Basically, the disk replaces all the library routines normally found in the Action! SuperCartridge. We have found a few bugs in the original version(s) of the RunTime Library Disk. Fortunately, they are all easy to fix. (The RunTime library is independent of the cartridge, so bugs affect all versions.)

In the fixes given below, the portion to be changed (to implement the fix) is underlined. The rest of the line remains the same. To make the fixes, simply load the library file containing the affected PROCedure, edit, and save it back to disk.

1. Hex numbers are printed incorrectly by PrintH and the %H parameter of PrintF.
   FIX: Change second line of CCIO:
   PROC CCIO=*()
   [$A386$A0A$A0A$AA$A3A5$9D$342 ... etc.

2. PrintBDE can cause a spurious compile time error.
   FIX: Change first line of PrintBDE:
   PROC PrintBDE =*(BYTE d,n)[$A0$0]

3. A minor error exists in ChkErr.
   FIX: Change second line of ChkErr:
   PROC ChkErr=*(BYTE r,b,eC)[$1610$88C0$8F0
   $98$80C0$12F0 ...

4. If your program redefines a library procedure (e.g., one which declares its own version of PROC Graphics), it will compile with no errors using the cartridge only (because declared procedures take precedence over built-in ones). However, since the RunTime library uses this same precedence trick to include its own definitions of library procedures, your program may generate Error 6 (doubly defined name) if you do not delete the appropriate PROCedure (or FUNCtion) from the RunTime library before INCLUDEing it.
   FIX: Make a custom version of the RunTime library on a copy (please, only on a copy) of your RunTime disk.

Problems With PAD

We will list the problems (and solutions) regarding the Programmer's Aid Disk here in reasonably compact form. You may find more information by sending for the BUG Sheet.

1. BGET/BPUT have problems. The solution was published in the last newsletter. A more elegant solution (written entirely without code blocks) is available on the BUG Sheet.

2. PRINTF has a bug which was reported and fixed in the last newsletter. We reprint the fix here because it is small and easy. In the file PRINTF.ACT, use the Action! editor to find
   args ==+ s
   and change it to
   args ==+ 2

3. Because S: uses some memory just below the display list (undocumented), our method of finding the base address for Player/Missile Graphics needs a slight revision. Use the Action! editor with the file **PMG.ACT** to find

```
PM_BaseAdr=(HiMem-PM_MemSize(mode))&PM_AdrMask(mode)
```

and change it to

```
PM_BaseAdr=(HiMem-PM_MemSize(mode)-$80)&PM_AdrMask(mode)
```

4. If you use the **PMMove** procedure and specify a vertical movement of zero, the horizontal movement does not take place (it should!). To fix this, change the lines in **PMG.ACT** which read

```
IF deltay=0 THEN
  RETURN ; do nothing
FI
```

to the following:

```
IF deltay=0 THEN
  PMHpos(n)=x   RETURN ; do horizontal anyway
FI
```

5. The documentation for **PMG.ACT** states that you may read the contents of **PMHpos** to find the horizontal position of a player or missile. This is simply not true. **PMHpos** is a set of write-only hardware registers. (Note that in the Toolkit we have added a shadow array and changed the name of the hardware registers, so this works correctly. If you wish, you could consider doing something similar on your PAD.)

6. There are two discrepancies in PROCedure names in the **REAL.ACT** library as compared to the **REAL.DOC** documentation, as follow:

Name in .DOC	Name in .ACT
StrR	RealToStr
ValR	StrToReal

We suggest that you change the source code in **REAL.ACT** to reflect the names given in the documentation (rather than vice versa), since this makes the names appear compatibile with the library's other number-string conversion routines.

7. In that same area, the routine **RealToStr** (or should that be **StrR**?) needs to change the line which reads

```
ptr=LBuff
```

to the following:

```
ptr=InBuff
```

ToolKit Troubles

It's hard to believe that a product as new as the Action! Toolkit can already have bug reports. Sigh. Anyway, there are already two versions of the ToolKit. Version 1 has 31 free sectors (when you list its directory). Version 2 has fewer free sectors and the second line of the file **MUSIC.DEM** reads ";Version 2". The comments here are organized by affected version(s).

Version 1 -- The manual describes a routine called **Format** (in the **IO.ACT** library), but no such procedure exists on the disk. However, the routine is there--it's just called **Init** instead. You should change either the manual or the disk to agree.

Version 1 -- The program called **MUSIC.DEM** will not work as is on older 400/800 machines. This is because it uses a call to **Graphics(15)**, which is only available on XL machines. You may change the program to use **Graphics(8)** with no effect except that the true colors of mode 15 become artifact colors in mode 8 instead.

Versions 1 and 2 -- The manual indicates that the procedure **AllocInit** requires that you pass it the address of the first free byte of memory (because **Alloc** "dispenses" memory from the first free byte through the top of memory, as correctly described in the manual). However, since you must follow the procedure described in the introduction to **ALLOCATE.ACT** (that is, you must use the command

```
SET EndProg=*
```

after compiling), the parameter to **AllocInit** is not really needed and so has been eliminated. (**AllocInit** uses **EndProg** just as **Alloc** does.) If you pass a parameter to **AllocInit**, it will be ignored.

Mad MACs

MAC/65--especially the cartridge version--remains a relatively bug-free product. As with Action' though, we are maintaining a "BUG SHEET", and you are welcome to send for one. remember, on the outside of your envelope be sure to put the notation "MAC/65 BUGS". And you must enclose a self-addressed, stamped envelope.

(By the way, the reason for all this seeming foolishness with BUG SHEETs and self-stamped, well-dressed envelopes really is valid: If we went back to the beginning of all our products each newsletter, printing all the bug reports and tips, the newsletters would become 60 pages long and cost us more to mail than our annual sales. This way even someone buying one of our products from Discount Dan's Used Software Lot can get caught up on all the past info. If the BUG SHEETs get too big, we may have to start charging a nominal copying fee for them, but for now they are free.)

Improved I/O and Forward Fixes

In the I/O macros (either the file "IOMAC.LIB" for the disk version or the listing in the back of the cartridge version manual), we cautioned you that you must not make forward references to a buffer. For example:

```
PRINT 0,BUFFER
...
BUFFER .DS 40
```

will not work! Instead, we warn you to place the buffer definition before the usage of the macro, thus:

```
BUFFER .DS 40
...
PRINT 0,BUFFER
```

which works just fine, thank you.

The reason for this is that the macros check for a literal string by code such as

```
.IF %1<256 ; if not a buffer, assume string
```

Which works because MAC/65 returns the length of a string as its numeric value. For example, in the line

```
PRINT 0,"HI THERE!"
```

The value of %1 (parameter 1) is zero (as coded), but the value of %2 is nine (9) because the length of "HI THERE!" is 9.

The problem with placing the buffer definition after the macro usage is that MAC/65 gives any undefined label a value of zero (especially for macro evaluation purposes). Thus it sees

```
PRINT 0,BUFFER
```

(where BUFFER is defined after its usage here) as the same as

```
PRINT 0,"BUFFER"
```

(believe it or else!) on the first pass of the assembly! On the second pass, though, BUFFER has been defined, so it takes on its proper value and is not treated as a string. Presto! Phase errors!

What can you do about it? The general solution is to always define a label before using it in a macro (a good idea with 6502 code anyway, because of zero-page ambiguities). However, for the special case of the OPEN and PRINT macros as we supplied them to you, you may make a minor modification which will allow the buffer to be defined after the macro usage. To make this fix, we depend on the fact that the value of an undefined label is zero and test for the specific value of zero. Thus, after applying these patches,

```
PRINT 0,BUFFER
```

will work fine, but

```
PRINT 0,BUFFER+2
```

will not work, since MAC sees that as 0+2 giving 2 which (once again) looks like the length of a literal string.

Anyway, the changes to be made are shown below. Simply add the underlined portion to each line shown:

For OPEN:

```
1730   .IF %1<256 .AND %1<>0
```

For PRINT:

```
3260   .IF %2<128 .AND %2<>0
```

Fantastic Funky Feature Found!

It's true. We "found" a hidden feature in the cartridge version of MAC/65. And it can make editing easier.

When you use the FIND or REP edit commands, the format is given as follows (using REP as the example):

```
REP /<old>/<new>/ [lno1[,lno2]][(,A)(,Q)]
```

and the manual notes that the delimiter (shown as / here) may be any character except a space. (Actually, the manual only states this for FIND, but it's true for REP as well.) All that is true. But...

Because of the way MAC/65 works, any characters you place between the delimiters will be converted to upper case. Searching for comments in lower case letters is thus well nigh impossible. But (and you saw this coming, didn't you) is you use a quote (") as your delimiter, the search (and/or replace) string is not converted at all, and the search is for an exact match.

BASICXLLY BASIC

Excepting for the fact that many of you are coming up with more Atari BASIC programs which simply will not work with BASIC XL (usually because of reasons documented in the manual, but see below also), BASIC XL seems pretty stable. The few bugs documented at this time are not major ones and are easy to get around. We do not expect to release a revised version of BASIC XL for some time to come, but look for announcements of add-on capabilities (including, for example, **THE BASIC XL TOOLKIT**).

This issue of the newsletter, then, we have only two topics related to BASIC XL.

Exciting Enhancement

Actually, this isn't so much an enhancement as it is an exploration of an almost-undocumented capability of BASIC XL.

If you have used the enhanced I/O capabilities of BASIC XL at all, you have probably played with or at least noted the capabilities of **BGET** and **BPUT**. Specifically, you can write out an entire string quickly and efficiently via the following:

```
DIM ST$(1000)
BPUT #file, ADR(ST$), 1000
```

(where the file number and dimension of the string--1000 here--are arbitrary).

If you consider how BASIC XL (and, for that matter, Atari BASIC) allocate memory, you may realize that you can use this same capability with numeric arrays:

```
INPUT x,y,z
DIM ST$(x), ARRAY(y,z)
BPUT #file, ADR(ST$)+x, (y+1)*(z+1)*6
```

We used the INPUT statement to point out that the values of x, y, and z are completely arbitrary. Not at all arbitrary, though, is the fact that **ARRAY** is **DIM**ensioned directly after **ST$** is! Doing it this way ensures that BASIC XL will allocate the space for **ARRAY** directly following the space used by **ST$**. The result? The address of **ARRAY** is greater than the address of **ST$** by exactly the dimensioned size of **ST$**. So, even though we can not use the **ADR** function with an array, we can use this trick to get the address of an array. (Sorry, but there is no correspondingly easy way to get the address of a simple numeric variable.)

And what about the size of memory that we wrote to disk with **BPUT**? Remember that using a statement such as

```
DIM XX(1,1)
```

actually creates an array of two "rows" by two "columns". Specifically, we would have available

```
XX(0,0)  XX(0,1)  XX(1,0)  XX(1,1)
```

because the zero-th elements are available.

Accounting for the zero-th elements explains why we used (x+1) and (y+1). Why do we multiply that result by six? Simply because each (floating point) number in Atari BASIC and BASIC XL occupies 6 bytes of memory.

As if all this was not enough, there is yet another very interesting capability of **BGET** and **BPUT** related to string arrays.

Before explaining our next "trick", let's refresh your memory about how memory is allocated for string arrays. When you use a statement such as

```
DIM SA$(x,y)
```

BASIC XL allocates x strings of y characters each. Then, when you assign data to elements of the array (e.g., via

```
SA$( 3 ; ) = "testing"
```

which assigns a string to the third element of the array), BASIC must remember the length of each individual string within the array. [Sidelight: for ordinary strings, BASIC maintains this information within the "overhead" of the variable value table, as part of the fixed 8 bytes per variable.] It remembers these lengths by adding two bytes to the beginning of each element of the array. Thus our statement

```
DIM SA$(x,y)
```

results in an allocation of x * (y+2) bytes. Or, to use a specific example,

```
DIM X$(30,10)
```

would allocate 360 [30 * (10+2)] bytes.

At this point it seems easy, then, to write out the entire string array, using the same techniques we used for the numeric arrays, just above. Actually, though, there is an even easier way:

```
BPUT #file, ADR( SA$(1;) ) - 2, x * (y+2)
```

Isn't that cute? The trick is that using the function call **ADR(SA$(1;))** returns the address of the first byte of the first string in the array. But, since the length bytes for this string precede the string itself, the address of the start is simply two less! Voila.

What you may not have noticed, though, is yet another side benefit. Since **BPUT** writes out all those length counts along <u>with</u> the strings, using **BGET** will recall them all correctly again! Thus the caveat in the manual (about BGET not changing the length of a string) is not needed if you read and write string arrays in this manner. H ver, powerful enhancement to disk I/O if used properly. Try it sometime.

Compatibility Comments

As we hope you have discovered by now, there is a section of your BASIC XL manual devoted to the issue of compatibilit (and incompatibility) of BASIC XL with Atari BASIC. As noted in that section, there are some things which can never be compatible between the two BASICs and several things where you, as the programmer, have the choice of being compatible or not. We present here a correction to that section (for some manuals) and two more (sigh!) recently discovered problem areas.

SELECT button -- The manual states (page 133) that using the command **RUN "filename"** will automatically load the file and then RUN the program in **FAST** mode. To defeat the **FAST** mode, you hold down the **SELECT** button. **THIS IS COMPLETELY BACKWARDS** if you have a version 1.02 cartridge! Version 1.02 will not go into **FAST** mode when it encounters **RUN "<filename>** <u>unless</u> it sees the **SELECT** button held down.

Quoted DATA -- This one is pretty obscure, but some article in some magazine managed to find it: Because BASIC XL allows you to place your string **DATA** values in quotes (so that you can include commas in the data if desired), it must (of necessity) throw those quotes away when the value is **READ** into a string variable. If you have an Atari BASIC program which has string **DATA** which begins with a quote, BASIC XL will not be (and can not be) compatible.

Now, truthfully, we can't figure out why anybody would use quotes in Atari BASIC DATA. The magazine example program was one which had been converted from another BASIC which required quotes. The program had to go to lot of work to get rid of the quotes after READing! **DUMB!** However, we can (barely) imagine a need to start a string with a quote (perhaps special character data or a machine language subroutine in character form). We offer a suggestion: Put a junk character in front of each data item which can start with a quote and strip off that character. Example:

```
1000 DATA .some data
1010 DATA ."some data which starts with a quote
```

GRAPHICS does too much -- BASIC XL handles Player/Missile Graphics for you in such an automatic way that you are probably not even aware of where the buffer needed by PMG is placed (but we're gonna tell you where). When BASIC XL sees a **PMGRAPHICS 1** or **PMGRAPHICS 2**, it determines where the bottom of the GRAPHICS screen is. It then allocates the needed PMG buffer (1280 bytes for PMG.1, 640 Bytes for PMG.2) as close as possible below the screen.

Suppose, now, that you change GRAPHICS modes. Your screen memory may grow and end up being in the same locations that the PMG buffer uses. Or the screen memory may shrink and leave a lot of wasted space between the PMG buffer and the screen. In either case, BASIC XL solves possible problems by simply turning off PMGraphics whenever it encounters a **GRAPHICS** statement. This is neat, albeit not overly elegant, solution.

The problem arises when running programs written under Atari BASIC, where the programmer has gone out and allocated his/her own PMG buffer. If this Atari BASIC program then makes a **GRAPHICS** call, BASIC XL quite graciously turns off the PMGraphics for you! Oops.

If you are trying to RUN protected software under BASIC XL, there may be no solution. If, however, you are using a magazine-published program, we can suggest some fixes:

1. You can rearrange the program slightly so that the **GRAPHICS** statement always appears before the PMG setup.
2. You can change the **GRAPHICS** statement to an **OPEN** statement instead. Specifically, a statement of the form

```
GRAPHICS n
```

should be changed to

```
CLOSE #6 : OPEN #6,12+((n&$30)%$10),n&$0F,"S:"
```

C-SICK

C/65 is one of our more mature products (that simply means we haven't made any revisions in a long time). Some of our products--C/65 included--are not produced by OSS employees. We simply market them for the original authors, so we are often at the mercy of the author(s) when changes and fixes are needed. In the case of C/65, we have been expecting a revised version for at least six months now. We are still expecting it, hopefully soon.

In the meantime, we have prepared a BUG SHEET with several warnings, fixes, and ideas. You are welcome to send a self-addressed, stamped envelope with a request for a free copy of the **C/65 BUG SHEET**.