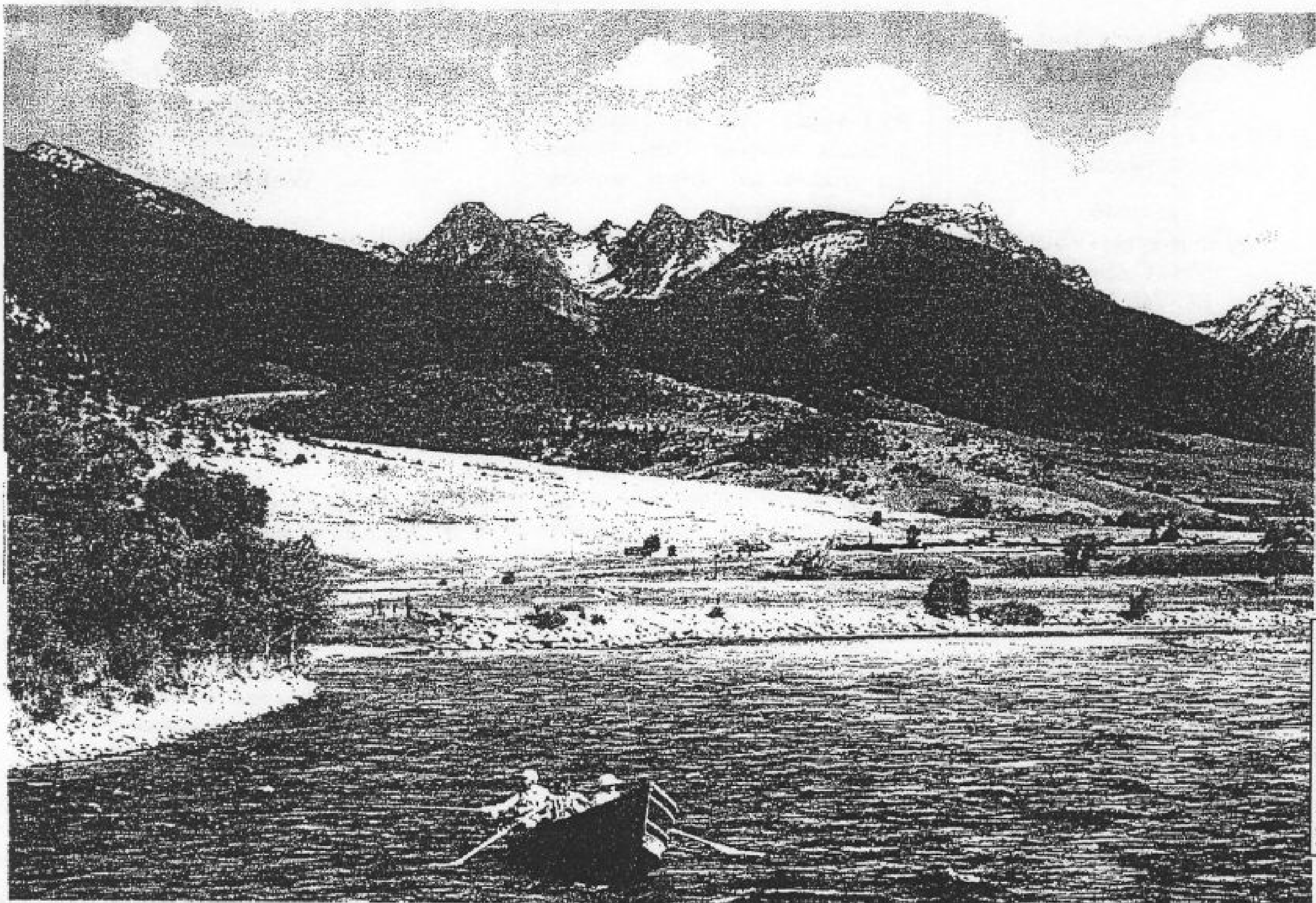
Still Alive With Sir Clive!

ZXir QLive Alive!

The Timex/Sinclair NorthAmerican User Groups Newsletter

Volume 9 No. 3 Autumn '99

MEMORY MAP



Yellow Stone River

ZXir QLive Alive! ©
Established 1991 The Timex/Sinclair NorthAmerican User Groups Newsletter

T/SNUG Information

We wish to support the following platforms : ZX-80/81, TS-1000, Spectrum, TS-2068, Z88 and QL. If you have any questions about any of these fine Sinclairs, contact the:

Chairman

Chief Motivator
Donald S. Lambert
738 Gunnar Ln.
Forsyth, IL 62535

Vice-Chairmen

Tape & JLO PD Library
D. G. Smith
415 Stone St.
Johnstown, PA 15906
814 535-6998

Z88 Library
Dave Bennett (HATSUG)
1275 Timber View Dr.
Mechanicsburg, PA 17055-9146
717 732-4374

QL Hacker's Journal
Timothy Swenson
2455 Medallion Dr.
Union City, CA 94587-1914
swensontc@geocities.com

TS-2068
Rod Humphreys (VSUG)
10984 Collins Pl.
Delta, BC V4C 7E6 Canada
604 583-2819

QL PD Library
John Donaldson (CATUG)
835 Foxwood Cir.
Geneva, IL 60134-1631
630 232-6147

AERCO & Z80 Emulator
Keith Watson
41634 Amberly Dr.
Mt. Clemens, MI 48038

BBS --==GATOR==--
Bob Swoger (CATUG)
613 Parkside Cir.
Streamwood, IL 60107-1647
630 837-7957 Work 847 576-8068

Any of the above can also be reached by E-Mail through the **Club BBS 847 632-5558**

ZXir QLive Alive!

Is the newsletter of T/SNUG, the Timex/Sinclair North American User Groups, providing news and software support to the T/S community in a **VOLUME** of four newsletters per year; beginning with the Spring (March) issue.

T/SNUG's main goal is to preserve and encourage the use of Sinclair computers by providing an open forum for the exchange of knowledge, building and maintaining of software libraries. Providing vendors, repair service and members with free ad space.

It is the user groups and individual subscribers, rather than the vendors, that provide the pecuniary support for this newsletter. Vendors and developers receive this newsletter free of charge, though contribution from vendors and user groups is gratefully accepted. Please support our vendors and service providers whenever possible.

If you have a problem or you have solved a problem, please share it with the rest of us. No problem will be considered unimportant.

Editor/Treasurer Publisher

You can keep T/SNUG alive by an annual contribution of $12 for one VOLUME made payable to Abed Kahale. Send check to:-

ABED KAHALE
3343 S FLAT ROCK CT
SIERRA VISTA AZ 85650-6874

520 378-3424

Back copies are available for $1.00 each postpaid.

Trea$ury Note$

As of September 1 , 1999, we have a balance of $840.22

Article Contributions

Send in your articles by disk, hardcopy or mail, e-mail and your inputs to:—

Abed Kahale
E-mail: AKahale@compuserve.com

GATOR'S
Twisted Pair

To better inform the Sinclair Community, three 24-hour a day BBSs are now provided to serve you. You are encouraged to exchange mail and use the files sections of these boards. Bulletins and ads are available to all.

Q-Box BBS 810 254-9878
Utica, Michigan
SOL BBS 520 882-0388
Tucson, Arizona
Club BBS 847 632-5558
Arlington Heights, Illinois

WebPages

http://users.aol.clubbbs/tsnug/
http://www.outlawnet.com/~jboatno4

If you know the Internet E-Mail address of a Sinclair user, but do not have access to Internet, simply address your E-Mail to GATOR Sinclair on the 24-hour **Club BBS** and include the name and E-Mail address of the user you wish to reach. Then check the Club BBS from time to time if you expect a reply.

We encourage you to exchange mail and contribute to the UPLOAD section. Call and ***register*** using your first, last name and phone number along with a password you won't forget. ***Write It Down!*** Do not try to do anything else at this time.

When you call-in the next time, you will have Level 5 security and be able to enjoy full user privileges. The BBS has smaller sections called conferences. Select "J " for "Join a Conference". Select "TIMEX" to get into the Sinclair Section. The mail you then read will only be from other T/S users. Use extension .ART for articles, .ADS for ads and .NWS for news when UPLOADing.

For help, contact the SYSOP, Bob Swoger, by leaving a message, mail, E-Mail or phone.
CENG108@email.mot.com

Input/Output

by Abed Kahale

Abed,

I have a **joystick**, a **Wickes roller ball**, and sketchpad etc. that I would be thrilled to donate in hopes that someone would enjoy playing with them. Some have not been fired up in a decade but were working then. Also I have both **cassettes** and **cartridges** (commercial programs) from Timex and Psion, etc. I will give my old 300 Baud **modems** and everything except my bare bones LarKen and Zebra systems with the 2 little printers. I will also give my two **Banana Gorillas** -- one a **serial** and one a **parallel**. I will pay shipping to our two **surplus** places, I am on verge of having to rent a storage chamber monthly to calm the clutter around here.

Joan Kealy

Thanks! Abed, for sending the Spring and Summer issue's of the ZXir Alive Newsletter. I did not get them for some reason. Must be our mail.

Sorry to hear about your wife. Also, sorry, that I didn't get back to you about the ZXir newsletter's. My IBM was down last week. I'm having to re-install stuff on my machine and it's going to take me at least a month to get everything back up and running right!!

Thanks again for the newsletter!!

Dane Stegman

Dear Mr. Kahale, I have been studying code programming for the Timex Sinclair TS 1000. I have had some modest success, and I'm very interested in being in touch with other Z80 users. Thank you for any info you can toss my way. Respectfully yours,

Stephen Waldman

brogine@hotmail.com

I know there is one person who is good at the Z80 code.

Keith Watson

Keith_watson@juno.com

There is a very active group in Germany for the ZX-81 but, one has to know German.

Hello Abed:

My name is **Richard Burt**, and I am a long time Timex Sinclair user, and former dealer in this area. I belonged to several TS groups in Canada, all of which are gone now, and I still long for info of the TS brotherhood. I still use my Sinclair computers although not as much as I used to, just the 2068, and the QL now.

If you would email me back the information reguarding membership, and the mailing address, it would be much appreciated. Thanks in advance. Rick

Sender: ajb@intranet.ca

-==GATOR==--

bobswoger@juno.com

CENG108@email.mot.com

bobs@comm.mot.com

--------- Begin forwarded message ----------

From: William McBrine <wmcbrine@clark.net>

From: "Richard Dodds" <dicky@popup.freeserve.co.uk>

-----Original Message-----

From: owner-2068@unixville.com [mailto:owner-2068@unixville.com]

On Behalf of William McBrine

To: TS-2068 users

Subject: Two-liner

Of all the programs I've written for my TS2068, probably my favorite is this trivial little thing:

```
 10 POKE 22528+RND*767,RND*63
 20 GO TO 10
```

It was supposed to print squares of random color at random locations on the screen. (Why? I dunno; that's just the sort of thing I did back then.) But it didn't quite turn out that way. As more squares were plotted, a distinct pattern emerged.

This program exposes the "pseudo" aspect of the psuedo-random number generator. I also find it quite beautiful. The best part is that the location of each plot does appear random, and there's some margin of error, so that a particular cell may change colors a few times, while remaining within the overall pattern. Replacing "63" with "127" also works well.

If you reverse the order in which the two calls to RND are made, the effect is lost. –

William McBrine | http://www.clark.net/~wmcbrine/ wmcbrine@clark.net

Hi Mr. Kahale,

I gather Don doesn't do e-mail. I just wanted to say that I found the T/S web page. You might remember me from an article I wrote about 7 years ago for the newsletter on using the T/S 1000 with CompuServe. I still use my T/S 1000 with the Byte-Back I/O board to control my model train layout. I was hoping I could write Don directly, but gather he still likes snail mail. :-)

Glad to see his name at the top of the page. Don was very helpful to me, and I will never forget his kindness. Please say "hi" for me. Taker care,

Joe Rampolla. jprampolla@blazenet.net

Hi Abed,

Thanks for the reply! Yes I still have the article, but until I get a scanner I won't be able to e-mail it. Anyway, I doubt that CompuServe still handles its mail the way it did about 7 years ago.

I know Mr. Lambert didn't think he could handle an online service like CompuServe back when I was using the T/S 1000 with CompuServe. I just bought a new PC, an E-Machine, and I am new to Windows, not to mention Windows 98. The old way with an ASCII interface and CompuServe was easy and less confusing for me. Too much stuff on the screen, and the Internet is full of junk, and surfing can be a real waste of time, in my opinion. But at least I found the T/S web site!!!!

I look forward to e-mailing Mr. Lambert when he gets online. I remember he moved a couple years before I moved from Baltimore City, MD to here in Hanover, PA. Also look forward to keeping in touch with you! I would enjoy hearing about others who still use the old Byte-Back I/O board, the BB-1. I had a real challenge figuring a way, on my own, to unpack the input byte.

Take care, **Joe**.

(His article is in the Vol 2, No.2, Summer 1992)

Dear Redneck Son;

I'm writing this letter slow because I know you can't read fast.

We don't live where we did when you left home. Your dad read in the newspaper that most accidents happen within 20 miles from your home, so we moved.

I won't be able to send you the address because the last family that lived here took the house numbers when they moved so that they wouldn't have to change their address.

This place is really nice. It even has a washing machine. I'm not sure it works so well though: last week I put a load in and pulled the chain and haven't seen them since.

The weather isn't bad here. It only rained twice last week; the first time for three days and the second time for four days.

About that coat you wanted me to send you, your uncle Stanley said it would be too heavy to send in the mail with the brass buttons on, so we cut them off and put them in the pockets.

John locked his keys in the truck yesterday. We were really worried because it took him two hours to get me and your father out of the truck

Your sister had a baby this morning; but I haven't found out what it is yet so I don't know if your an aunt or an uncle. The baby looks just like your brother..........

Uncle Ted fell in a whiskey vat last week. Some men tried to pull him out, but he fought them off playfully and drowned. We had him cremated and he burned for three days.

Three of your friends went off a bridge in a pick-up truck. Ralph was driving. He rolled down the window and swam to safety. Your other two friends were in back. They drowned because they couldn't get the tailgate down.

There isn't much more news at this time. Nothing much has happened.

Love, Mom

P.S. I was going to send you some money but the envelope was already sealed.

During the course of World War II, many people gained fame in one way or another. One man was Butch O'Hare. He was a fighter pilot assigned to an aircraft carrier in the Pacific. One time his entire squadron was assigned to fly a particular mission. After he was airborne, he looked at his fuel gauge and realized that someone had forgotten to top off his fuel tank. Because of this, he would not have enough fuel to complete his mission and get back to his ship. His flight leader told him to leave formation and return. As he was returning to the mothership, he could see a squadron of Japanese Zeroes heading toward the fleet to attack. And with all the fighter planes gone, the fleet was almost defenseless. His was the only opportunity to distract and divert them.

Single-handedly, he dove into the formation of Japanese planes and attacked them. The American fighter planes were rigged with cameras, so that as they flew and fought, pictures were taken so pilots could learn more about the terrain, enemy maneuvers, etc.

Butch dove at them and shot until all his ammunition was gone, then he would dive and try to clip off a wing or tail or anything that would make the enemy planes unfit to fly. He did anything he could to keep them from reaching the American ships. Finally, the Japanese squadron took off in another direction, and Butch O'Hare and his fighter, both badly shot up, limped back to the carrier. He told his story, but not until the film from the camera on his plane was developed, did they realize the extent he really went to, to protect his fleet. He was recognized as a hero and given one of the nation's highest military honors. And, as you may know, O'Hare Airport in Chicago was named after him.

Prior to this time, in Chicago, there was a man called Easy Eddie. He was working for a man you've all heard about, Al Capone. Al Capone wasn't famous for anything heroic, but he was notorious for the murders he'd committed and the illegal thing's he'd done. Easy Eddie was Al Capone's lawyer and he was very good. In fact, because of his skill, he was able to keep Al Capone out of jail. To show his appreciation, Al Capone paid him very well.

He not only earned big money, he would get extra things, like a residence that filled an entire Chicago city block. The house was fenced, and he had live-in help and all of the conveniences of the day. Easy Eddie had a son. He loved his son and gave him all the best things while he was growing up; clothes, cars, and a good education. And, because he loved his son he tried to teach him right from wrong. But one thing he couldn't give his son was a good name, and a good example. Easy Eddie decided that this was much more important than all the riches he had given him. So, he went to the authorities in order to rectify the wrong he had done. In order to tell the truth, it meant he must testify against Al Capone, and he knew that Al Capone would do his best to have him killed. But he wanted most of all to try to be an example and to do the best he could to give back to his son, a good name. So, he testified.

Within the year, he was shot and killed on a lonely street in Chicago.

These sound like two unrelated stories, but Butch O'Hare was Easy Eddie's son.

For Cubs Fans

20 major events that have occurred since the Chicago Cubs last laid claim to a world series championship:

1. Radio was invented; Cubs fans got to hear their team lose.
2. TV was invented; Cubs fans got to see their team lose.
3. Baseball added 14 teams; Cubs fans get to see and hear their team lose to more clubs.
4. George Burns celebrated his 10th, 20th, 30th, 40th, 50th, 60th, 70th, 80th, 90th and 100th birthdays.
5. Haley's comet passed Earth twice.

6. Harry Caray was born....and died. Incredible, but true.
7. The NBA, NHL and NFL were formed, and Chicago teams won championships in each league.
8. Man landed on the moon, as have several home runs given up by Cubs pitchers.
9. Sixteen U.S. presidents were elected.
10. There were 11 amendments added to the Constitution.
11. Prohibition was created and repealed.
12. The Titanic was built, set sail, sank, was discovered and became the subject of major motion pictures, the latest giving Cubs fans hope that something that finishes on the bottom can come out on top.
13. Wrigley Field was built and becomes the oldest park in the National League.
14. Flag poles were erected on Wrigley Field roof to hold all of the team's future World Series pennants. Those flag poles have since rusted and been taken down.
15. A combination of 40 Summer and Winter Olympics have been held.
16. Thirteen baseball players have won the Triple Crown; several thanked Cubs pitchers.
17. Bell-bottoms came in style, went out of style and came back in style; disco did the same.
18. The Chicago White Sox, Cleveland Indians, Boston Red Sox and Florida Marlins have all won the World Series.
19. The Cubs played 14,153 regular-season games; they lost the majority of them.
20. Alaska, Arizona, Hawaii, Oklahoma and New Mexico were admitted to the Union.

Sinclair E-Mail List

Albrecht, Alvin	aralbrec@concentric.net
Anderson, Paul	p.anderson@prodigy.net
Anson, Gerald	jerrya@aztec.asu.edu
Barker Robin	Robin@di-ren.demon.co.uk
Bennett, Dave	dbennett10@desupernet.net
Boatwright, Jack	jboatno4@outlawnet.com
Boehm, Al	boehm@ziplink.net
Boehm, Bill	boehm@plh.af.mil
Burt, Richard	ajb@intranet.ca
C. A. T. S.	mf0002@epfl2.epflbalto.org
Catotti, Christopher	kd4ace@compuserve.com
Chambers, George	gfchamb@pathcom.com
Collins, Bill	bcollins@home.ifx.net
Cottrell, Les	jacottre@gte.net
Cruz-Figueroa, Jaime	cruzfiguer@aol.com
Dansby, Andrew	adansby@atlantic.net
Davis, Frank	fdavis@iquest.net
Dunnet, Ron	ron@qubbesoft.freeserve.co.uk
England, William	wengland@iname.com
Feng, Al	alfeng@juno.com
Fink, Mike	domino.cubes@excelsior.net
Fink, Mike	domino.cubes@pointblank.com
Firshman, Tony	tony@firshman.demon.co.uk
Florit, Louis	florit@wormhole.unixville.com
Franke, John	j.m.franke@larc.nasa.gov
Ganger, Gary	gangerg@dma.org
Gillespie, Doug	aa431@cleveland.freenet.edu
Girnius, William	girnius_w@bls.gov
Gowen, Rod	jl911@kanga.ins.cwru.Edu
Harbit, Ken	krh03@cvip.fresno.com
Henderlight, Mike	mikehend@microsoft.com
Henn, Fred	oranur@juno.com
Humphreys, Rod	rodh@lightspeed.bc.ca
Hunkins, James	jdhunki@ibm.net
Impellizerri, John	jimpellizerri@compuserve.com
Jaap, Matthias	matthias_Jáap@hhs.hh.schule.de
Jonas, Mike	mjonas@bbn.com
Jones, Dilwyn	dilwyn.jones@dj.softnet.co.uk
Jones, Terry	tjones@iname.com
Kaczor, Jon	75363.1127@compuserve.com
Kahale, Abed	akahale@compuserve.com
Kealy, Harriet Joan	hjkealy@admin.hilconet.com
Kenny, Larry	larken@storm.ca
Kingsley, Ed	edk4@aol.com
König, Urs	urs.koenig@agrodata.ch
KurtK7	kurtk7@aol.com
Kwitkowski, Phillip	kwit47@aol.com
Lancaster, Garry	dharkhig@delphi.com
Lanciault, Francois	francois.lanciault@energies.alstom.ca
Lassov, David	emanon@azstarnet.com
LaVerne, Melvin	mlaverne@usit.net
Lebowitz, Dave	dkl@dpliv.com
Lessenberry, Gary	gl743@aol.com
Liebert-Adelt, Peter	p.liebert@t-online.de
Malloy, Bob	74776.2342@compuserve.com
McBrine, William	wmcbrine@clark.net
McKelvey, William	mckelveyw@delphi.com
Merz, Jochen	jmerz@t-online.de
Merz, Jochen	jochenmerz@j-m-s.com
Miller, Seymour	seymil@delphi.com
Muth, Bob	bobkeeper1@aol.com
Norton, Gary	gnorton@world.std.com
Parrish, Gil	gil.parrish@abanet.org
Pashtoon, Nazir	nazir.pashtoon@ingram.micro.com
Paul Holmgren	paulholm@indy.net
Payne, Josh	joshpayne@bigfoot.com
Pazmino, John	john.pazmino@moondog.com
Perry, Russ Jr	slapdash@enteract.com
Rampolla, Joe	jprampolla@blazenet.net
Rigter, Wilf	rigter@cafe.net
Rish John	74601.1535@compuserve.com
Shepard, Jay	jshepard@netins.net
Simon, Thomas	73177.333@compuserve.com
Skapinski, Tom	tskapins@juno.com
Smith, Dennis	denny.smith@juno.com
Solly, David	ac355@freenet.carleton.ca
Stegman, Dan	danesteg@juno.com
Swenson, Tim	swensontc@geocities.com
Swenson, Tim	swensont@sirclive.csd.sgi.com
Swentko, Wally	wswentko@maroon.tc.umn.edu
Swoger, Robert	ceng108@email.mot.com
Taylor, Jeff	jetaylor@mdrobotics.ca
Thoresen, Jeff	74200.257@compuserve.com
Waldman, Stephen	brogine@hotmail.com
Walterman, Don	walterm@ix.netcom.com
Watson, Keith	Keith_watson@juno.com
Wood, Roy	qbranch@qbranch.demon.co.uk

From The Chairman's Disk

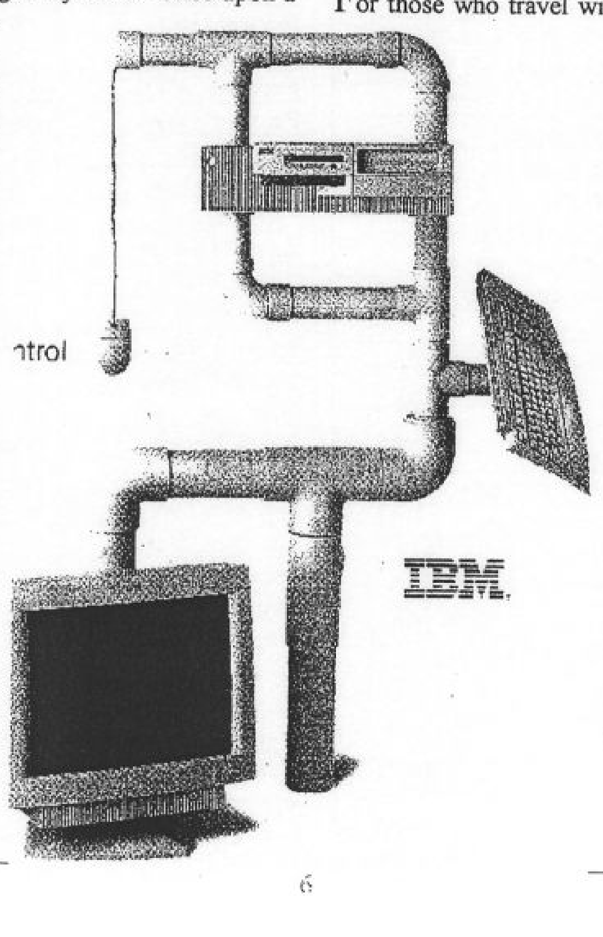
Donald S. Lambert

This is a hard one to write I am putting my old friend my trusty TS-2068, in a box and shipping it to Jack Boatwright soon after I type this in. Why must I do this? It is because we are moving and have to cut out a lot of extra stuff that is not longer needed - part of which is what my wife says. I will still have the Z88 and I plan to try to get active with that and of course I am thinking about a PC

I bought my first TS-2068 in 1985 and from that point on deserted the TS-1000 I went through the woes of bad LOADs on the cassette mass storage till I built a LOAD AID and later learned to use a 8Ω to a 1000Ω audio transformer to LOAD questionable cassettes and when that failed I learned to use a Radio Shack miniamplifier to LOAD those that failed with the transformer. Then in 1988 I went to the LarKen disk drive and since then I used the cassette system as little as possible. After I moved to Indiana I acquired an Oliger disk interface and soon had an Oliger/LarKen interface which I used till I sent the last of my equipment to Jack Boatwright.

I have sent many boxes of my accumulation of hardware, software and books, newsletters and magazines to Jack and now after I finish this I will be boxing up my last shipment to Jack I have set this up as my contribution to ZQA! for the FALL issue since I have no idea of how much time I will have available then to write or to get a computer set up and running (or learning to use it). I guess I have learned to no longer say never. Once upon a time I said I would never leave the TS-1000 but then along came the TS-2068. Then I said I would never go to disk drives and I did. And I said that I would never leave the TS-2068 and here I am shipping my last one to Jack. I guess that I had better learn to say, "I don't plan to ...." and that would be far closer to the truth.

But I have learned a lot about computers from working with the TS-2068 (and the TS-1000) which is what I was after when I got into using them. have learned that the field is moving so fast that it is almost impossible to keep up with the latest developments. If and when I buy a PC I know that it will be obsolete before I get it out of the box, but at my age I will get all the latest that I can afford and plan to have it my last big computer.



It is not easy to part with my 2068! It is like an old friend that is moving far away and I will never see again. As was packing the stuff that I had to send Jack if I stopped to think about it I would never have gotten the first box packed. I just could not stop to sort and pack stuff in categories more than I did. I do hope that Jack wasn't too greatly upset about that

The disk case next to the keyboard on the computer desk has only three disks in it that have files on them, they are a disk of letters I have wrote, a disk of Oliger Utilities and a disk of personal finance. The later I have printed out all the important stuff (the latest file of each category) to take with me to keep me acquainted with where our money is and how much we have. Or how much we will have left after the move.

I knew nothing about disk drives when I got into that field with the LarKen interface. I had to learn how to trouble shoot one particular type of disk drive (Tandon TM 100-2A); to reset track zero and speed I had to learn about the terminator resistor on the last drive on a ribbon cable and to set the proper jumpers to get it to operate. Now I will be getting into hard drives with capacity of gigabytes. How much is a gigabyte? I just Can't visualize how much I am used to 5 K bytes to a track and 400K bytes to a disk so one gig is about 2,500 400 K disks. BUT! then the programs on a PC are so big. Get as big a hard drive as you can afford is what everyone tells me

For those who travel with a Z88 I have found that a soft cassette box will hold nine AA batteries if you twist off the little deals inside that keeps the reels from turning, of course the batteries are just fat enough to keep the lid from completely closing but a heavy rubber band will cure that. Two batteries will lay full length of the case and the other seven will lay in there crosswise to them makes a neat package for travel.

I have a parallel printer connecting cord for both the Z88 and the Laser PC3 which will (or should) allow me to print out what I write on either. BUT! I do not expect instant use! I expect some down time learning the proper settings to get the printer to work.

So this is all for now and I hope to give more information on the Z88 in the next issue. 0/0

Donald S. Lambert
738 Gunnar Ln.
Forsyth, IL 62535

QL Hacker's Journal

Supporting All QL Programmers #31 June 1999

The QL Hacker's Journal (QHJ) is published by Tim Swenson as a service to the QL Community. The QHJ is freely distributable. Past issues are available on disk, via e-mail, or via the Anon-FTP server, garbo.uwasa.fi. The QHJ is always on the look out for article submissions.

QL Hacker's Journal
c/o Tim Swenson
2455 Medallion Dr.
Union City, CA 94587
swensont@geocities.comswensont@mail.sns.com
http://www.geocities.com/SiliconValley/Pines/5865
/index.html

Editors' Forum

This issue is far later that I would like. Planning for the West Coast Sinclair Show (for which this issue is being prepared) took a fair amount of time. I've also been fiddling around the house getting it ready for guests coming to the show. I've also been distracted by some projects from work. I'm looking forward to running Linux on my Q40 and doing some "Professional" stuff with it.

Well, after about a year, the Qliberator Source Book project has reached a point that I have enough material to release. Since Qliberator was only a small part of the main document, I changed the title to the SuperBasic Source Book. The focus was on Qliberator, Programming Toolkits, and Programming Tools. There is no sections on the SuperBasic language itself. The emphasis was on what is necessary to produce compiled professional code. I plan to add more information to the Source Book as I find the time. A number of sections came from the QHJ, but were expanded with more information. The whole document (along with many others) can be found on my web page:

www.geocites.com/SiliconValley/Pines/5865/

Last issue covered two different languages, Perl and AWK. This started me thinking about other languages that are available on the QL and who, if any, uses them. A number of people have ported different languages to the QL, from the then popular XLISP, to the now popular Perl. Each language has it's own features and reasons for being. What I want to know is, who uses these languages? Has anybody done anything useful with these languages? Do you have a favorite language that you like to program in? If you use a language on the QL other than Assembly, C, SuperBasic, or Perl (we just touched it), let me know. Tell me what language you use, what you use it for, how it suits your needs, and provide an example of a useful tool you've created with the language.

Structured SuperBasic 2.6.1

One project keeping me away from working on the QHJ was updating Structured SuperBasic. I've made a few minor changes to it that allows it to much more useful. I've also added a new utility that makes it SSB production easier.

The two changes are:

1 - Added a second command line argument of Starting Line Number. This was made so that SSB would be used with the Unix utility 'make'. Make, which comes with the C68 distribution, is a tool designed to only compile those sections of code that have changed. With SSB, if you have a program comprised of 5 program files, and you only change 1 one them, you have to run SSB on the whole lot (if you've used #include statements). Using 'make', you leave out the #include statements and let make run the SSB filter for you. If you have only changed 1 file, the only that file is converted from _SSB to _BAS, then all of the _BAS files are added together into one file (using 'cat', another Unix utility).

I'm not saying that we all should be using 'make', but I wanted the ability to use it built into SSB.

2 - When using the command line, if SSB fails with an error, the _BAS file is deleted. This was need for the SSBGO, the utility mentioned below.

3 - Better error reporting.

The new utility is SSBGO. SSBGO automates running SSB and Qliberator. SSBGO was designed to be used with MicroEmacs. From MicroEmacs, I would saved a file, then use the execute-program command. I would enter:

```
  ssbgo flp1_file &
```

(the & is to EXEC it and not EXEC_W it)

SSBGO will then run SSB on 'flp1_file_ssb' (_ssb is the default extension). Once it has file_bas, it then load file_bas in to SuperBasic, SAVEs the file to a temporary file, checks it for the keyword MISTake and exists if one exists. If not, it executes the LIBERATE command, which is the command line interface to Qliberator. Qliberator then fires up, compiles the program, and finishes. So I have gone from having SSB source to a compiled program in just a few minutes, with only one command. SSBGO can be use with other editors, you just need to CTRL-C out of the editor, EXEC SSBGO, and let it run.

SSB261_ZIP is available on my web page.

FileConfig

Another program that I have been working on is FileConfig. The short explanation is that FileConfig is an automated version of BasConfig, by O. Fink (and modified by Norman Dunbar and Dilwyn Jones). With BasConfig, you can't edit an existing Config Block, only create a new one. With FileConfig, you store the definition of a Config Block in a text file. If you need to edit a Config Block, edit the definition file, run FileConfig, and you have a new Config Block.

FileConfig is designed for programmers and has very minimal error checking. It does not verify the data put into the Config Blocks, which means that you could attempt to put the character A into a Byte item. If you attempt this, the results will be "undefined", which is a nice way of saying, "you are on your own." FileConfig is available on my web site.

Microemacs Macros

Thierry Godefroy has ported over the latest version of MicroEmacs 4.00 to the QL. He has added Pointer Environment support for MicroEmacs, including menu items for all the commands. This has made MicroEmacs much more appealing and much easier to use. Just before this port, I had been playing with a little with configuring MicroEmacs and tinkering with macros. I have tried using macros in the past, but I had not quite figured out how to use them. With all the commands now available from pull-down menus, it is very easy to execute a macro from a file.

Now that I know how to execute macros from a file, the next thing was to figure out what would be useful to write as a macro.

I've looked over some macros that I found on the MicroEmacs web page. These macros helped create HTML files. These macros would query the user for any information it needed when creating HTML constructs. This means that an application could be written in macros, querying the user for certain data, and generate an end product. Given the math functions built into MicroEmacs, one could write short little calculating programs, just like we did, years ago, on the ZX-81.

The macro I wrote to show how this works is a simple mail-merge like application. The user creates a document, with fields marked out where they want information to go. Here is a short example:

```
  @fname@ @lname@
  @street@
  @city@
  Hello @fname@,
  How are you today? How is your wife
@wife@?
  Signed,
```

Here there are fields marked for first name, last name, street, city, and wife's' name. Since this is a text editor, I used an at sign (@) at the beginning and end of each field to make it distinct from the rest of the test. The macro will first query the user for the information and then it will go through the text file, replacing the marked fields with the user provided data. A macro like this can be useful if you are writing a Christmas letter that you to make a little more personal, but still save time in writing. The macro is faster than editing the document yourself, or even running the same search-and-replace queries.

The command

```
   set %variable @"String"
```

tells MicroEmacs to query the user for input, showing "String" on the command line, and store the data in the variable %variable. Without the at sign (@), the string "String" would be stored in %variable.

```
  ;This macro will query the user for
some items
  ; and then replace them with marked
fields in the
  ; text file.
  goto-line 1
  set %fname @"First Name : "
  set %lname @"Last Name : "
  set %addr @"Street Address : "
 set %city @"City : "
 set %wife @"Wife's First Name : "
  write-message "Replacing Text ..."
  replace-string "@fname@" %fname
  ; Need to goto to the beginning of the
  ; file because the search starts from
  ; where the cursor is to the end of
file.
  beginning-of-file
  replace-string "@lname@" %lname
  beginning-of-file
  replace-string "@street@" %addr
  beginning-of-file
  replace-string "@city@" %city
  beginning-of-file
  replace-string "@wife@" %wife
  beginning-of-file
  write-message "Done ..."
```

When Thierry introduced spell checking with MicroEmacs 4.00, it only allowed spell checking of a single word, already marked. I thought it would be a good idea to write a macro that would talk through a file and spell check them all. Thierry has since mentioned that he is looking to expand the spell checking capability to be more user friendly. But, still the idea of writing a macro to walk through a file, word by word, seemed like a good challenge. Below is the macro.

```
store-procedure get-word
  set $kill ""
  !force next-word
  set-mark
  !force end-of-word
  copy-region
  set %word $kill
;   write-message &cat "The Word is : "
%word
!endm
end-of-file
set %end $curline
beginning-of-file
!while &less $curline %end
  get-word
  write-message &cat "Word is : " %word
!endwhile
```

Program Internationalization

About 10 years ago I attended some vendor training on how to program and extend their particular office automation suite. One of the things that I took away from the training was how they designed their system to adapt to many languages. Recently the method came to mind. As I was thinking about the possibilities of using this method in my own programs, I pondered over it's limitations and what other methods might be used.

After some thought I have considered three different approaches to allowing a program to support multiple languages:.

- Text File
- One file per language
- All languages stored in executable
- One executable per language

Before covering the different approaches, the main thing that each approach hinges on is the storage of all output messages in an array. Instead of having a line like this:

```
   PRINT #3,"File Not Found"
```

You would have something like this:

```
   PRINT #3,messay_array$(35)
```

Since the output messages are not hard coded, the array can be changed to suit the language. For every possible output message you would have to put an entry in the array. Granted this will make reading and maintaining the source code more difficult, but it does make supporting different languages so easy.

Now the difference in each approach is how to store the different arrays for each language. Each method has some pluses and minuses and each have to taken into consideration for each programs needs.

The first method mentioned, Text File, is the method that I learned in the training class. The developers created a text file, in each language, of all of the possible messages. Each file would be given a different name. The program would expect a certain file name. The current language would be renamed to that file name. When the program was executed, the program would read the file and load messages from the file into the array.

The problem with this method is the overhead of reading in the file. If you have a program that may be executed many times in a single session, the speed of the program will suffer from reading in the language file each time. If you are writing an application that once executed will run for a while, such as a word processor or spreadsheet. One advantage of having that messages in a text file is that new languages can be added to the program, with no change to the executable.

One way of getting around the overhead of reading in a text file is to store all of the messages of all the languages in the executable and have a command line option or environment variable determine which language is chosen. With SuperBasic the different messages would be stored in DATA statements. When the determination of which language is made, the program would read select which DATA statements to read into the array.

There are two disadvantages to this approach. First, the space needed for each languages may add significantly to the size of the executable. Secondly, if a new language is needed to be added, then the program has to be recompiled.

Another approach is to create the program the same way as the previous approach, but use conditional compilation to create an executable for each language. This will cut down on the amount of space needed for the message data, but it does mean that a different executable would have to be distributed for each language. If you included each language executable on the distribution media, then this approach may work.

Given the wide distribution of QL users and the many different languages, having a localized version of an application may make an application more accepted in the community. The difficult part will be in translating the messages into different languages. If the Text File approach is used, then users could translate the messages and distribute the new language file to other users. I believe something like this has been done with various dictionary files.

TS-2068 ROM ByPass Board Features and Options II

by the late William Pedersen WDJUP

1. The PULL rear connector is repeated so other accessories can be used, including a bank switching controller and back-plane.
2. Use two 28 pin sockets for "HOME ROM" and two for "EXROM". That way you can use RAM chips before burning EPROM's. You can plug in a direct replacement for 6264 RAM, but not 43256.
3. When using EPROM, a 27128 chip can be used for "HOME ROM", leaving one socket empty.
4. When using ROM for "EXROM", use 43256 chips for reasons of economy. (32K*8bits)
5. "EXROM" memory from 0 to 32K is assumed to be 43256 "ROM" or 27256 EPROM. (See Pig. 2a,c)
6. Write-protect switches are used when RAM takes the place of ROM. One covers "HOME ROM", the other CHUNKS #0 thru #3 of "EXROM".
7. Other switches allow RAM to be initialized (written) before being configured as "ROM". This allows HOME ROM and EXROM to be copied to external RAM.
8. A jumper is provided to eliminate battery backup once EPROM has been installed in place of write-protected "HOME" ROM.

Because bypass operations need BE to operate, and external banks have higher priority, the board must monitor BE' to avoid conflict. The circuit shown breaks into the BE signal line to make this possible.

The average user wants better operation, more memory, more gadgets and software which makes use of them. who can blame them, but not everyone has the technical knowledge and skill to do it on their own.

The design presented here is hereby declared PUBLIC DOMAIN, with the blessings of The WIDJUP co. Suppliers are free to manufacture, but it is strongly suggested that The WIDJUP co. be consulted when doing 80.

PROCEDURE	S1 SWITCH SETTINGS						NOTES
	a	b	c	d	e	f	
NORMAL OPERATION:							ROM/EPROM EPROM
"HOME":							0-8,8-16K; 0-16K
Using EPROM	ON	ON	OFF	(Fig 1)			(PGM High)
Using RAM as "ROM"	ON	ON	OFF	(Fig 1)			0-16K Read Only
"EXROM":							0-32,32-64K;0-32K
Using EPROM & RAM	(Fig 2c)			---	---	---	32-64K RD/WR
Using RAM as "ROM"	(Fig 2a)			ON	ON	ON	0-32K Read Only
EXROM Removed	(Fig 2b)			---	ON	OFF	0-32K Read Only
COPYING INTERNAL ROM:							
HOME to "HOME" RAM:	OFF	OFF	ON	(Fig 1)			Write Only 0-16K
EXROM to "EXROM" RAM:	(Fig 2a)			OFF	OFF	OFF	Write Only 0-8K
EXROM to EPROM ROM:	(Fig 2c)			---	---	---	Done Externally
EXROM Removed	(Fig 2b)			---	OFF	ON	Before removal
MAKING "ROM" CHANGES:							
"HOME":	ON	ON	ON	(Fig 1)			Read/Write 0-16K
"EXROM":	(Fig 2a)			ON	ON	OFF	Read/Write 0-32K
EPROM:	(Fig 2c)			---	---	---	Done Exterally
EXROM Removed	(Fig 2b)			---	ON	ON	Read/Write 0-32K

This Bypass Design Stands Alone.

Nevertheless, RGB video, analog signals, external power sources, bus re-powering and attachment of back-planes are features best located on the same board. Provisions for adding features would satisfy those not yet ready for slot expansion.

It is quite easy to copy internal ROM to the bypass board.

```
10 REM Fig 1: HOME ROM
20 FOR N = 0 TO 16383
30 POKE N, PEEK N
40 NEXT N
-----------------------------------
10 REM Fig 2a,b: EXROM
20 SAVE "EXROM" CODE 0, 8196
30 LOAD "EXROM" CODE 0, 8196
```

Once an updated operating system is available, the most convenient way to operate is with a combination of Figure 1 (with a single 27128) and Figure 2c. The switches and battery backup can be eliminated. All you need are the proper EPROMs and board. The TS2068 will still operate without the board attached. If you like to change operating systems, like using SPECTRUM or CP/M, So you COULD have a set of EPROMs for each. This requires no special knowledge or procedures. There is, however, another way.

If you like to try your hand at making operating system changes by yourself, you can.

If you prefer to simply load one of several operating systems provided by others, you can.

The combination of Figure 1 and Figure 2a can hold any operating system including the original. Using a fairly simple procedure which can be prompted by a loaded program, any operating system can be "booted" into the TS2068 as software, just like other computers do.

The switches on the board are hardware in the figures, but they could easily be operated by I/O, making booting entirely automatic.

Once a system has been booted, you don't have to repeat it every time you turn your computer off and on. Who said the TS2068 is an orphan?

It is probably best to explain a little bit about how these circuits work without getting too technical.

If you ever tried writing to ROM, you know nothing happens. All you can do is read from it. These boards have STATIC RAM covering the same address space as ROM. That CAN be written to. you just have to be careful not to read anything back or confusion reigns. This is solved by a switch which prevents reading from RAM.

It is interesting that a program can be running in ROM at the same time it is being "written to". There is no interference. In fact, the ROM program can make a copy of itself in RAM, down to the last detail.

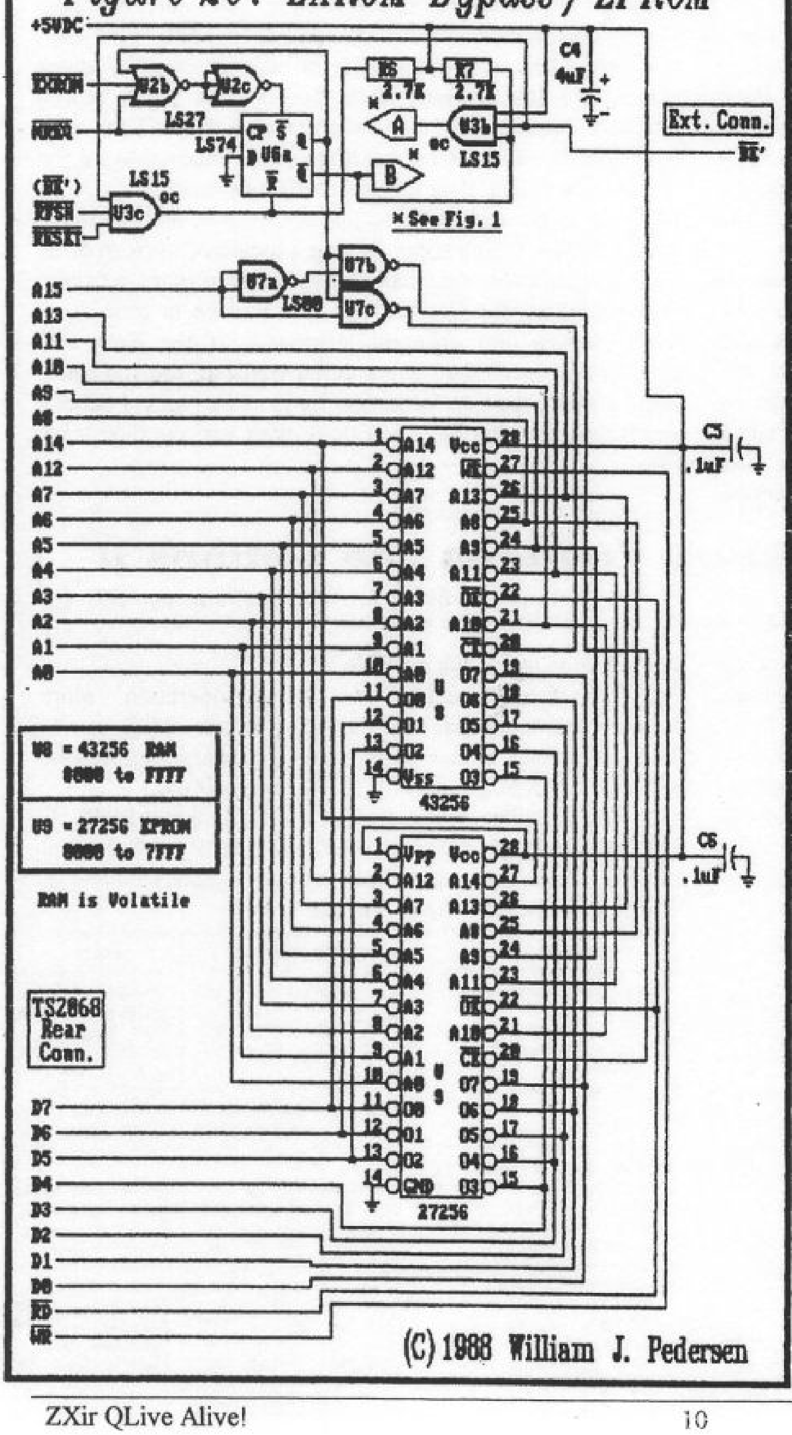
Ah hah! Now we can switch from ROM to RAM, switch the RAM to read-only, and you can't see anything different---so why do it?

You wouldn't, of course. Why copy ROM code when you can do better? The only reasons you might are to check out the hardware and to get a head start on making your own code modifications.

The rest of the circuitry makes sure that not only will your present programs run without changes, but that features TIMEX left out WILL. It will if you want, that is. Should you want FORTRAN, Pascal, FORTH, or something else instead, it is possible. The TS2068 hardware is a marvel of design. It lends itself to bank-switching for large memory capacity. It has no inherent problems with time-sharing. It is fast. It has no problem working with an expansion back-plane, DMA, nor other bus masters.

It Can Even Run IBM & Clone Accessories!

Figure 2c: EXROM Bypass / EPROM

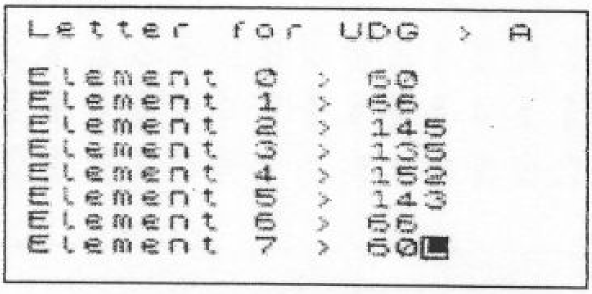


(C) 1988 William J. Pedersen

Function for Creating User Defined Graphics (UDGs) in HiSoft Pascal

Article and Program by David Solly

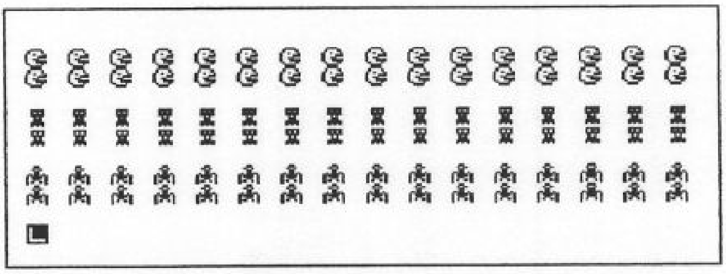
Unlike Sinclair BASIC, HiSoft Pascal does not have any reserved words like **BIN** to allow the user to create User Defined Graphics (UDGs). However, it is fairly simple to write your own function for creating UDGs within Pascal. The program listed below shows how I created the function **LOADUDG()**. The function has been written in such a way that it allows UDGs to be defined from elements supplied by the user through the keyboard, (Demo #1), or through hard coding within the program, (Demo #2). How this is achieved is explained in the program annotations. It only remains to be said that once the source code for LOADUDG() has been typed and debugged, the function may be saved to a library and called whenever required. Use and enjoy!



```
Letter for UDG > A

Element 0 > 60
Element 1 > 66
Element 2 > 145
Element 3 > 135
Element 4 > 152
Element 5 > 143
Element 6 > 66
Element 7 > 60
```

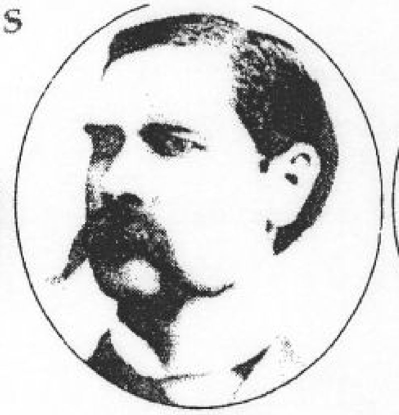
Program Listing

```
PROGRAM UDGDEFINE;
{
  This is a HiSoft Pascal Program.
  This program must be compiled in 31
column mode.
  Program by David Solly
  Written:  July 12, 1998
  Purpose:
    To demonstrate how user defined
graphics (UDGs) can
    be defined within a HiSoft Pascal
Program.
}
VAR
  L, GGC, GGC2, GGC3 : CHAR;
  {Graphic to define, Demo Graphics 1, 2
& 3}
  G0, G1, G2, G3, G4, G5, G6, G7, I :
INTEGER;
  {Graphic elements 0 to 7, Loop
counter}
  {These variables allow UDGs to be
definded by elements
    supplied by the useser through the
keyboard}

FUNCTION LOADUDG(
  LC : CHAR;
  LG0, LG1, LG2, LG3, LG4, LG5, LG6, LG7
: INTEGER) : CHAR;

VAR
  GC : CHAR;      {To hold character of
the UDG when found}
  LOC : INTEGER; {To hold the UDG start
address}

BEGIN

   {Find the UDG to define
   UDGs start at CHR(144), i.e. 79
characters after 'A' and are
   usually referred to as Graphic A,
Graphic B .. Graphic U.}

  GC := CHR(ORD(LC) + 79);

  {Find how many characters it is into
the alphabet}
  LOC := ORD(LC) - ORD('A');

  {Calculate the address of the 1st
element in the
   UDG buffer

   Note:  The start of the UDG buffer is
65638d but Hex notation
   must be used here to avoid a 2's
compliment error.}

  LOC := (LOC * 8) + #FF58;

  {Poke 1st element of the UDG into its
memory location
   then add 1 to LOC for each additional
element}
  POKE(LOC, LG0);
  LOC := LOC + 1;
  POKE(LOC, LG1);
  LOC := LOC + 1;
  POKE(LOC, LG2);
  LOC := LOC + 1;
  POKE(LOC, LG3);
  LOC := LOC + 1;
  POKE(LOC, LG4);
  LOC := LOC + 1;
  POKE(LOC, LG5);
  LOC := LOC + 1;
  POKE(LOC, LG6);
  LOC := LOC + 1;
  POKE(LOC, LG7);
  LOADUDG := GC;
END;

BEGIN {Main Program}
  {DEMO #1: Define UDG from data entered
through the keyboard}
  PAGE;
  {Fetch the letter for UDG definition}
  WRITE('UDG to define > ');
  READLN; {Required in HS Pascal}
  READ(L);
  {Fetch the 8 elements for the UDG}
  WRITELN;
  WRITE('Element 0 > ');
  READLN; {Required in HS Pascal}
  READ(G0);
  WRITE('Element 1 > ');
  READLN;
  READ(G1);
  WRITE('Element 2 > ');
  READLN;
  READ(G2);
  WRITE('Element 3 > ');
  READLN;
  READ(G3);
  WRITE('Element 4 > ');
  READLN;
  READ(G4);
  WRITE('Element 5 > ');
  READLN;
  READ(G5);
  WRITE('Element 6 > ');
  READLN;
  READ(G6);
  WRITE('Element 7 > ');
  READLN;
  READ(G7);
  {Define the UDG}
  GGC := LOADUDG(L, G0, G1, G2, G3, G4,
G5, G6, G7);
  {Write out 2 lines of the new UDG}
  PAGE;
  FOR I := 1 TO 64 DO
    WRITE(GGC);
  WRITELN;
  WRITELN;
  {DEMO #2:  Examples of direct UDG
definition}
  {Space Alien #1}
  GGC2 := LOADUDG('B', 24, 60, 90, 126,
24, 36, 66, 0);
  {Write out 2 lines of the new UDG}
  FOR I := 1 TO 64 DO
    WRITE(GGC2);
  WRITELN;
  WRITELN;
  {Space Alien #2}
  GGC3 := LOADUDG('C', 24, 36, 24, 90,
189, 153, 165, 129);
  {Write out 2 lines of the new UDG}
  FOR I := 1 TO 64 DO
    WRITE(GGC3);
  WRITELN;
  WRITELN;
END.
```



Remembering 30-seconds of gunfire and death 117 years ago ...

Are they guilty?



Wyatt Earp

Doc Holliday

BILL HESS
Herald/Review

TOMBSTONE — It's part of Western lore. More than a century ago, eight men engaged in a 30-second gun battle in a corral.

Three died in the hail of bullets, and the accusations of who did what and why are still flying today.

One side says the lawmen murdered their opponents. The backers of the Earp brothers and Doc Holliday claimed the fight was fair and the deaths of the "cowboys" was justifiable homicide.

Was it murder or self-defense is still the unanswered question of the events at the OK Corral on Oct. 26, 1881.

For more on the mock trial, see page 16A.

There was no trial for Wyatt Earp, his brothers Virgil and Morgan and John Henry "Doc" Holliday for the deaths of Billy Clanton and the McLaury brothers, Frank and Tom. Until now, Wyatt Earp and Doc Holliday — both long dead — will face justice in a Phoenix courtroom Thursday.

The two long-dead men are being "sued" for causing the wrongful death of the McLaury brothers in a civil suit filed by the dead men's mother. What will be used against the men are accounts of the action based on testimony of witnesses and the accused themselves gleaned from a hearing, or inquest, conducted for almost a month by Tombstone Magistrate Wells Spicer in November of 1881.

Spicer concluded that no jury would find Wyatt Earp or Holliday guilty of murder and he ordered the two men released from jail. Wyatt Earp and Holliday had been jailed without bond after their arrest on Nov. 4, 1881.

Friday, June 18, 1999 **Page 16A**

Part - 6 Cont.

```
    5E4C F3      DI
    5E4D C9      RET
```

This code decrypts the headerless file we have just loaded in. The decrypter is CALLed to, and it's the same one we had before (with the RRD). So, in actual fact, we can forget about the BASIC loader; just load in the headerless file normally and run our own decrypter. For now, just put a breakpoint at #5E4D and JP to #5E3F. The RET is to #9C40

```
      9C40 21 52 9C       LD HL,#9C52
      9C43 01 90 01       LD BC,#0190
      9C46 16 A5      LD D,#A5
      9C48 7E         LD A,(HL)
      9C49 AA         XOR D
      9C4A 77         LD (HL),A
      9C4B 23         INC HL
      9C4C 0B         DEC BC
      9C4D 78         LD A,B
      9C4E B1         OR C
      9C4F C2 48 9C       JP NZ,9C48
```

This is a decrypter, and you can crack it in one of two ways. Firstly, you can copy if to somewhere else, put a breakpoint on the end, and run it from there, or you can replace the JP NZ,9C48 with JR NZ,9C48. Both do the same thing, but the JR NZ only uses two bytes. This means we can put a RET at #9C51 and CALL the decrypter. So change #9C4F to #20, #9C50 to #F7 and #9C51 to #C9, then put a CALL #9C40 and a breakpoint somewhere convenient (such as #5B00) and run the decrypter from there. When decrypted, the code continues at #9C52.

```
      9C52 21 63 9C       LD HL,#9C63
      9C55 11 45 FE       LD DE,#FE45
      9C58 01 90 01       LD BC,#0190
      9C5B ED B0      LDIR
      9C5D 31 84 FD       LD SP,#FD84
      9C60 C3 45 FE       JP #FE45
```

This moves the code we've just decrypted to #FE45, sets the stack pointer to #FD84, and JPs to #FE45. Put a breakpoint at #9C60 and JP to #9C52.

Now at #FE45, we come to the actual loading system itself.

```
      FE45 3E 84      LD A,#84
      FE47 11 00 18       LD DE,#1800
      FE4A DD 21 00 40    LD IX,#4000
      FE4E CD AB FE  CALL #FEAB
```

This code loads in a headerless file with start #4000 and length #1800 (which is the display file for the screen). Nothing too unusual about that.....

```
      FE51 11 00 04       LD DE,#0400
      FE54 DD 21 FF 5B    LD IX,#5BFF
      FE58 CD 2A FF       CALL #FF2A
```

....except that as soon as it's done that, it loads in a block with start #5BFF and length #0400 straight away. This block is "sandwiched" right next to the other block on the tape. This particular block loads "backwards".

```
      FE5B 11 E1 01       LD DE,#01E1
      FE5E DD 21 1F FE    LD IX,#FE1F
      FE62 CD FA FA  CALL #FEFA
```

And here's another block of the same kind, except it's loaded forwards this time. What's worse is that it's going to overwrite the code we're looking at now, so it must be a "modification" to the loading system (similar to the Mikro-Gen loader). To get round this, we would have to copy the code somewhere else, stick a breakpoint on the end, and run it from there. But remember that the loading system was copied from address #9C63, so there is, in actual fact, a copy of the code anyway. You want to put a breakpoint at #9C83 (the instruction after loading these three blocks), and JP to #9C63. Then start the tape and load in the next headerless block. The loading screen will appear, and the game will load for about four seconds, then control will return to the disassembler. Now look at the code at #FE65.

```
      FE65 11 1D 9F       LD DE,#9F1D
      FE68 DD 21 1C FA    LD IX,#FA1C
      FE6C CD 2A FF  CALL #FF2A
```

This code loads the main game backwards. This will overwrite your disassembler, so you will have to put the NEW routine at #FE6F (but write down all the bytes you are replacing, because you'll need to restore all the original code later). Then you will have to go back and load the first block, because there isn't a header for the main game block. Change #FE4D to #01, #FE57 to #10 and #FE61 to #01 - this will make the computer try to load the three blocks into the ROM. Then rewind the tape back to the start of this headerless block, JP #FE45, and start the tape. Upond loading the whole block, the computer will reset. Load in your disassembler, and replace the code from the NEW routine to the values they should be. Now you can tackle the final part of the loading system.

```
      FE6F 11 E4 12       LD DE,#12E4
      FE72 DD 21 FF FF    LD IX,#FFFF
      FE76 C3 70 FF       JP #FF70
      FF70 3E 00      LD A,0
      FF72 D3 FE      OUT (#FE),A
      FF74 CD 1F FE       CALL #FE1F
      FF77 21 43 FE       LD HL,#FE43
      FF7A BE         CP (HL)
      FF7B CA 89 FF       JP Z,#FF89
      FF7E 21 48 EE       LD HL,#EE48
      FF81 01 FF FF       LD BC,#FFFF
      FF84 11 49 EE       LD DE,#EE49
      FF87 ED B0      LDIR
```

The routine at #FE1F adds up all the memory in the screen to get a value in the D register. This is then compared with the value of the byte at #FE43. If there is no match, all the memory is blanked out, so the value in the D register must be the same as the byte at #FE43. You should find that the byte is #E6. You need to know this for later on.

```
      FF89 21 A3 FF       LD HL,#FFA3
      FF8C 01 45 00       LD BC,#0045
      FF8F 7A         LD A,D
      FF90 AE         XOR (HL)
      FF92 23         INC HL
      FF93 0B         DEC BC
      FF94 78         LD A,B
      FF95 B1         OR C
      FF96 C2 8F FF       JP NZ,#FF8F
```

This is all we can disassemble for the moment, because the code from #FF89 to #FF98 decrypts the final part of the loader. Change the byte at #FF87 to #16, and the byte at #FF88 to #E6 (this is LD D,#E6, which is used in the decrypter), put a breakpoint at #FF99, and JP to #FF87. Then continue the disassembly.

```
FF99 CD 31 FE       CALL #FE31
FF9C 21 44 FE       LD HL,#FE44
FF9F BE      CP (HL)
FFA0 C2 7E FF       JP NZ,#FF7E
```

This code checks the main game, coming out with a result in the E register. However, this value is never used, so you can ignore this whole routine.

Following on.....

```
FFA3 21 C0 5D       LD HL,#5DC0
FFA6 01 30 75       LD BC,#7530
FFA9 CD D4 FF   CALL #FFD4
FFAC 21 C0 5D       LD HL,#5DC0
FFAF 01 30 75       LD BC,#7530
FFB2 CD DE FF   CALL #FFDE
FFB5 21 1C FA       LD HL,#FA1C
FFB8 11 1C FF       LD DE,#FF1C
FFBB 01 1D 9F       LD BC,#9F1D
FFBE ED B8      LDIR
FFC0 21 10 A7       LD HL,#A710
FFC3 22 36 5C       LD (#5C36),HL
FFC6 01 10 DF       LD BC,#DF10
FFC9 AF      XOR A
FFCA ED 42      SBC HL,BC
FFCC 31 FF FF       LD SP,FFFF
FFCF ED 56      IM1
FFD1 C3 6F 00       JP #006F
006F E9      JP (HL)
```

This routine runs the game Decrypters, moves the game into the right place, sets the stack and the interrupts, and puts the start address for the game in the HL register. Change the #006F at #FFD2 to somewhere where you can put a NEW routine (such as #5B00), because the disassembler will be overwritten. Then, change the value at #FFA1 to 16, and the value at #FFA2 to #E6 (because one decrypter uses the value in the D register), and JP to #FFA2. When that's done, you can reload your disassembler, and hack the game using a forwards and backwards trace (but you won't be able to run the code because some of it's missing!)

Now we'll write a complete hack for the game. You have to be a bit careful about where you put your hack in memory, because a lot of memory is overloaded. The first free address we can put the code at is #FA1D.

```
FA1D DD 21 40 9C  LD IX,#9C40
FA21 11 90 01       LD DE,#0190
FA24 3E 07      LD A,#07
FA26 37      SCF
FA27 CD 56 05       CALL #0556
FA2A 30 F1 JR NC,#FA1D
```

This loads in the first headerless block using the values set up in the BASIC loader.

```
FA2C 06 FF      LD B,#FF
FA2E 21 40 9C       LD HL,#9C40
FA31 7E      LD A,(HL)
FA32 ED 6       RRD
FA34 23      INC HL
FA35 10 FA      DJNZ #FA31
FA37 06 FF      LD B,#FF
FA39 7E         LD A,(HL)
FA3A ED 67      RRD
FA3C 23      INC HL
FA3D 10 FA      DJNZ, #FA39
```

This decrypts the headerless block.

```
FA3F 21 20 F7       LD HL,#F720
FA42 22 4F 9C       LD (#9C4F),HL
FA45 3E C9      LD A,#C9
FA46 32 51 9C       LD (#9C51),A
FA49 CD 40 9C       CALL #9C40
```

This changes the JP NZ at #9C4F to a JR NZ and a RET, then calls the decrypter.

```
FA4C 3E C3      LD A,#C3
FA4E 32 83 9C       LD (#9C83),A
FA51 21 5A FA       LD HL,#FA5A
FA54 22 84 9C       LD (#9C84),HL
FA57 C3 63 9C       JP #9C63
```

This puts a JP back to our hack at #9C83, and jumps to #9C63 to load the first part of the headerless block.

```
FA5A 3E C9      LD A,#C9
FA5C 32 6F FE       LD (#FE6F),A
FA5F CD 65 FE       CALL #FE65
```

This puts a RET after the code to load the rest of the game, then CALLs that loading procedure.

```
FA62 21 E6 16       LD HL,#16E6
FA65 22 87 FF       LD (#FF87),HL
FA68 3E C9      LD A,#C9
FA6A 32 99 F        LD (#FF99),A
FA6D CD 87 FF       CALL #FF87
```

This patches in the LD D,#E6, puts a RET at the end of the decrypter, and CALLs it.

```
FA70 21 7E FA       LD HL,#FA7E
FA73 11 00 40       LD DE,#4000
FA76 01 20 00       LD BC,#0020
FA79 ED B0      LDIR
FA7B C3 00 40       JP #4000
```

This code moves our hack into the screen memory (so it isn't affected by the LDIR which overwrites it in the next bit of code), and jumps to it there.

```
FA7E 21 0B 40       LD HL,#400B
FA81 22 D2 FF       LD (#FFD2),HL
FA84 16 E6      LD D,#E6
FA86 C3 A3 FF       JP #FFA3
```

This code replaces the JP #006F with a JP back to our hack (which is in the screen memory by this time), and JPs to #FFA3. We have to include the LD D,#E6 again, because the value of DE was corrupted by the LDIR.

```
FA89 AF      XOR A
FA8A 32 C6 CD       LD (#CDC6),A
FA8D E9      JP (HL)
```

This sets the infinite lives POKE, and does a JP (HL) to start the game.

Phew! I hope you managed to get all that, because it's really hard to do without a Multiface. If you can do it, then you've definitely got the hang of things, so keep it up!

SEARCH LOADER

This loading system appears on every game ever written by Steve Marsden (who wrote the original loading system), as well as a few others. You can recognize them by their fancy front end, which consists of a countdown timer, accompanied by animated graphics and/or instructions, which appear as the game loads. The only game I've actually got at the moment that's got a Search Loader on it is Technician Ted, so I'm going to have to hack that.

So, *Hack the BASIC loader, and let's see what it's got to offer....

```
Chip Fact LINE 0 LEN 736
        0 RANDOMIZE USR 24341
```

The rest of the BASIC is a load of garbage which consists of the machine code for the game. It is stored in a similar way to that in the Mikro Gen loader. 24341 is 5F15 hex, so disassemble this address.

```
     5F15 F3      DI
     5F16 21 00 40        LD HL,#4000
     5F19 11 01 40        LD DE,#4001
     5F1C 01 FF 17        LD BC,#17FF
     5F1F 36 00      LD (HL),0
     5F21 ED B0      LDIR
```

This code blanks out the screen.

```
     5F23 CD 32 5E     CALL #5E32
```

The routine at #5E32 sets up the attributes for the screen i.e.: red banners at the top and bottom, black background with varying ink colors in the middle.

```
     5F26 C3 38 5F        JP #5F38
     5F38 21 AB 5F        LD HL,#5FAB
     5F3B 01 59 A0        LD BC,#A059
     5F3E 31 00 5C        LD SP,#5C00
     5F41 3A A8 5F        LD A,(#5FA8)
     5F44 57      LD D,A
     5F45 1E 0B      LD E,#0B
     5F47 7A      LD A,D
     5F48 87      ADD A,A
     5F49 87      ADD A,A
     5F4A 87      ADD A,A
     5F4B 87      ADD A,A
     5F4C 82      ADD A,D
     5F4D 83      ADD A,E
     5F4E 57      LD D,A
     5F4F 77      LD (HL),A
     5F50 23      INC HL
     5F51 0B      DEC BC
     5F52 78      LD A,B
     5F53 B1      OR C
     5F54 20 F1      JR NZ,#5F47
```

This routine fills all of the memory above #5FAB with unexecutable code. It is, however, extremely important code, as we shall see later on.

```
     5F56 CD 93 5F        CALL #5F93
```

This routine at #5F93 just messes around with the garbage a bit more.

```
     5F59 CD 80 5D        CALL #5D80
```

The routine at #5D80 scrolls in the title messages for the game, accompnied by annoying clicks.

```
     5F5C 3A 66 80        LD A,(#8066)
     5F5F 6F      LD L,A
     5F60 3A E6 60        LD A,(#60E6)
     5F63 67      LD H,A
     5F64 E5      PUSH HL
     5F65 3A 4F FC        LD A,(#FC4F)
     5F68 5F      LD E,A
     5F69 3A 0F 60        LD A,(#600F)
     5F6C 57      LD D,A
     5F6D DD E1      POP IX
     5F6F 37      SCF
     5F70 3E FF      LD A,#FF
     5F72 14      INC D
     5F73 08      EX AF,AF'
     5F74 15      DEC D
     5F75 3A 66 63        LD A,(#6366)
     5F78 6F      LD L,A
     5F79 3A E6 63        LD A,(#63E6)
     5F7C 67      LD H,A
     5F7D E5      PUSH HL
     5F7E DB FE      OUT (#FE),A
     5F80 1F         RRA
     5F81 E6 20      AND #20
     5F83 F6 01      OR #01
     5F85 4F      LD C,A
     5F86 BF      CP A
     5F87 F5      PUSH AF
     5F88 3A 87 65        LD A,(#6587)
     5F8B 6F      LD L,A
     5F8C 3A 85 6         LD A,(#6485)
     5F8F 6          LD H,A
     5F90 F1      POP AF
     5F91 E5      PUSH HL
     5F92 C9      RET
```

This code takes values out of the garbage and puts them in certain registers. It then imitates the start of the ROM loading routine, and puts some values on the stack. At #5F92, the values of the registers are: Hl= #056B, DE=#03C3, IX=#8000, and the values on the stack are first #056B, then #8000. So, this code will load a headerless file with start #8000 and length #0363, then will JP straight to #8000. We can do away with the BASIC loader altogether in the final hack by mimicking the headerless loader. This is done using the following program.

```
     5B00 F3      DI
     5B01 31 00 5C        LD SP,#5C00
     5B04 DD 21 00 80     LD IX,#8000
     5B08 11 C3 03        LD DE,#03C3
     5B0B 3E FF      LD A,#FF
     5B0D 3          SCF
     5B0E 14      INC D
     5B0F 08      EX AF,AF'
     5B10 15      DEC D
     5B11 AF      XOR A
     5B12 DB FE      OUT (#FE),A
     5B14 1F      RRA
     5B15 E6 20      AND #20
     5B17 F6 01      OR #01
     5B19 4F      LD C,A
     5B1A CD 6B 05        CALL #056B
     5B1D
```

This routine is slightly different than the one in the loader for two reasons. Firstly, I've put values into the registers directly, rather than have their values taken from bytes in memory. Secondly, you aren't allowed by law to rip off someone else's code; if you directly copied a loading system into a hack, you could be sued. In fact, someone was, once! You're probably all right copying a five byte decrypter from Powerload across, because there really isn't any other code which can do the job in the same way.

In general, I would say don't copy code into your hack unless you have to. If you do, change it if you can so it does the same job in a different way. Copying 40 bytes of code directly out of a loading system is definitely out, and most magazines wouldn't print the routine anyway.

There are a few commands we haven't met in the routine. EX AF', AF' concerns the swapping of registers. In the Z80, in actual fact, there are two different sets of each register, although only one set can ever be used at once. Think of it like a TV set, although both registers (A and A' in this case) are there, you can only see one at a time. EX AF,AF' exchanges both the A register and the contents of the flags. Don't worry any more about swapping registers for now.

RRA rotates all the bits in the A register to the right. Actually, it doesn't quite do this, but we don't need to know about it.

If you're not using a Multiface, the garbage routine will have overwritten the disassembler, so reload it in anywhere below #8000. Then run the routine and restart the tape from where you left off. A small part of the headerless block will load in, and control will return to the disassembler. Have a look at address #8000.

```
8000 D2 00 00      JP NC,#0000
```

This resets the computer if the previous headerless block didn't load properly.

```
8003 3E 08      LD A,#08
8005 D3 FE      OUT (#FE),A
```

This makes the border black, and sends a signal to the cassette recorder.

```
8007 D9      EXX
```

EXX is a "general exchange" instruction, and changes the registers B,C,D,E,H and L for their alternate sets.

```
800E 0E 00      LD C,#00
800A D9         EXX
800B 26 00      LD H,#00
800D 06 80      LD B,#80
800F DD 21 1C 8C  LD IX,#8C1C
8013 16 05      LD D,#05
8015 CD 41 83   CALL #8341
8018 D2 00 00   JP NC,#0000
801B 06 B1      LD B,#B1
801D 15         DEC D
801E 20 F5      JR NZ,#8015
```

This code loads in five bytes of tape (the routine at #8341 loads in information off tape into the address pointed to by the IX register), starting at #8C1C. Therefore, this tape routine will start loading code at #8C1C.

```
8020 11 D8 72      LD DE,#72D8
8023 D9            EXX
8024 0C            INC C
8025 D9            EXX
8026 C3 7D 83      JP #837D
837D 2E 01         LD L,#01
837F CD 41 83      CALL #8341
8382 D2 00 00      JP NC,#0000
8385 3E CB         LD A,#CB
8387 B8            CP B
8388 17            RLA
8389 D9            EXX
838A 47            LD B,A
838B E6 01         AND #01
838D 3D            INC A
838F 11 6B 80      LD DE,#806B
8392 1A            LD A,(DE)
8393 A7            AND A
8394 C4 77 82      CALL NZ,#8277
```

This loads in a set number of bytes from tape, and then prints some sprites on screen (the routine at #8277). This produces all the men walking forwards and backwards while the game loads.

```
8397 CB 18      RRB
8399 D9         EXX
839A CB 15      RL L
839C 06 B1      LD B,#B1
839E 30 DF      JR NC,#837F
83A0 7C         LD A,H
83A1 AD         XOR L
83A2 67         LD H,A
83A3 7A         LD A,D
83A4 B3         OR E
83A5 20 CE      JR NZ,#8375
```

This updates the computer ready to do the next loading and animation sequence.

```
83A7 7C         LD A,H
83A8 A7         AND A
83A9 C2 00 00   JP NZ,#0000
83AC 11 2F EC   LD DE,#EC2F
83AF 06 EB      LD B,#EB
83B1 CD 41 83   CALL #8431
83B4 D2 00 00   JP NC,#0000
83B7 3E EA      LD A,#EA
83B9 B8         CO #0B
83BA D2 00 00   JP NC,#0000
83BD 42         LD B,D
83BE 15         DEC D
83BF 1D         DEC E
83C0 C2 B1 83   JP NZ,#83B1
```

This loads in some more bytes from tape. When these have finished, the computer will continue execution at address #83C3. The code beyond this address does nothing except fiddle about with registers, so there might as well be nothing there. The first bit of useful code will appear at #8C1C, but this hasn't been loaded yet. Instead, put a breakpoint at #83C3, move the headerless loading routine so the CALL #056B is at #7FFD, and run the code from there. Rewind the tape a bit and reload in all of the main headerless block. When finished, you'll find the following code at #8C1C.

```
8C1C 3E 5C      LD A,#5C
8C1E 21 00 40   LD HL,#4000
8C21 54         LD D,H
8C22 5D         LD E,L
8C23 EB         EX DE,HL
8C24 4E         LD C,(HL)
8C25 23         INC HL
8C26 46         LD B,(HL)
8C27 23         INC HL
8C28 EB         EX DE,HL
8C29 09         ADD HL,BC
8C2A BA         CP D
8C2B 20 F6 JR   NZ,8C23
8C2D 11 92 5C   LD DE,#5C92
8C30 EE 5C      XOR 5C
8C32 28 EF      JR Z,#8C23
8C34 E5         PUSH HL
```

This routine adds up every single byte in memory to get a value in HL, which is PUSHed onto the stack. This value is important, because it is used in decrypting. This is what the garbage was all used for - to get the right value.

Unclassified Ads

Place your ads here, it is free!

Mail/E-mail to: A. KAHALE 3343 S FLAT ROCK CT SIERRA VISTA AZ 85650-6874

Please inform and/or **update the Editor of any changes in your ad/s**

SPECTRUM for your 2068

If you are a LarKen LK-DOS owner and would like to run SPECTRUM programs on your system, we will supply a V2 EPROM, socket and 74HCT32 for $12 which includes shipping and handling. The installation instructions are in your LarKen manual. We shall not be responsible for your install job. AERCO owners need only the EPROM for $10 forwarded to LarKen.

Bob Swoger Address on page 2

747 Flight Simulator

So you like to fly, the 747 Flight Simulator for SPECTRUM by Derek Ashton of DACC. Requires a SPECTRUM equipped 2068. Supplied on LarKen SSDD or DSDD LarKen disk for $10 which goes to Derek now working at Motorola with Bob.

Bob Swoger Address on page 2

Home Electronics Service

We have been a part of the Sinclair scene since 1982, repairing **ZX Spectrums** for Sinclair Research in England.
We provide Sales, Service, and Software for the

QL, Spectrum, ZX-81 and Z88.

www.members.tripod.com/hes_computing/hes1.html
E-Mail 74601.1535@compuserve.com
Hours of Operation is Monday - Friday 1300 hrs. to 2100 hrs. central time zone.
Phone 210 661-4376

John R. Rish
Home Electronics Service
5222 Kazen Dr.
San Antonio TX 78219 USA

The John Oliger Co.

11601 Widbey Dr.
Cumberland IN 46229

The John Oliger Floppy Disk System
FOR THE TS-2068
Disk Works
EXPANSION BOARD
2068 User Cartridge
DISK BOARDS "A" & "B"
2068 Parallel Printer Port
2068 EPROM Programmer
2068/SPECTRUM Joystick Port
DFh Mapped Universal I/O Port board
User Manual only : $5.00 (Read before you buy)

Service For America's
Favorite Home Computers and Their Accessories

SINCLAIR

TIMEX ADAMATARI IBM OSBORNE TI COMMODORE TRS-80

BUY SELL TRADE UPGRADE

Repair Charge Examples

TS-1000, ZX-81, 1016 RAMPack, Memotech, ZEBRA Talker, MIRACLE Centronics, RAM Centronics.
$5.00 each + parts & shipping.

TS-2020, 2040, PC-8300, ZX-80, TI-99, Z-SIO, Byte-Back, AERCO 2068 Centronics, BASICare, LarKen RAMDisk
$10.00 each + parts & shipping.

TS-2068, Spectrum, A&J MicrD, Miracle 512K, LarKen 1000 & 2068 FDI, Kempston FDI, Cumana FDI, CST FDI.
$15.00 each + parts & shipping. July 1, 1996

Reasonable flat rate plus parts and shipping.
Write or call for prices, SASE appreciated

COMPUTER CLASSICS

RT 1, Box 117
Cabool MO 65689
Phone 417 469-4571 417 467-4571

PLATYPUS SOFTWARE

QLAMBer	$20
QLuMSi	$20
SeekQL	$10
Upgrades	$5

Al Feng
914 Rio Vista Cir SW
Albuquerque NM 87105
(505) 843-8414

Make David an Offer
ZX-81/TS-1000 TS-2068 Hardware Kits
Real Time Clock I/O Controller RS-232
Centronics I/F 16K & 64K RAM 300 BAUD
Modem A-D Converter(assembled)

BYTE-BACK INC
536 Long Ter
Leesville SC 29070

ZX-TEXT - Word Processor
ZX-CALC - Spread Sheet
Business Software
Cycle Accounting Financial Report Generator
ZX-CALENDAR - Time Management
ZX-81 TS-1000 TS-1500 TS-2068
Albert F. Rodriguez
A.F.R. Software®
1605 Pennsylvania Ave 204
Miami Beach FL 33139
305 531-6464

WOOD & WIND

ARCHIVE Based QL Software

QLerk - A complete financial program for the QL

QLerk software (v3.21) with tutorial	$29
QLerk manual	$29
QLerk software & manual	$50

DBEasy - A menu based database system

DBEasy software (v1.6)	$24
DBEasy upgrade from V1.5	$7

DBProgs - A toolkit of ARCHIVE procedures

DBProgs software (v1.8)	$18
DBProgs upgrade from V1.7	$7

DBTutor - A general purpose learning program

DBTutor software(v1.5)	$12

PC DBEasy - Just like QL DBEasy but, you must have PC ARCHIVE to use It.

PC DBEasy software (v1.3)	$12

Bill Cable
Wood & Wind Computing
RR3 BOX 92
Cornish NH 03745 USA
Phone (603) 675-2218

QBox USA

810 254-9878
24 hours a day
300 to 28,800 bps

Supporting all Sinclair and Timex users Fidonet Echomail arefor Sinclair computers. Lots of new files for you to download such asTS-2068 emulator for those who use a PC give us a call and let us know what you want to see.Message and file areas. QL International, Quanta, QL Hacker's Journal, Spectrum/2068, ZX-81/TS-1000, Z88, NetMail, emulators, pointer, FDFormat for QXL/QDOS, etc.

Sysop John J Impellizzeri
Co-Sysop Don Walterman
Utica, Michigan, USA

Domino Cubes

Z88

Hardware & Software
352 7th Ave. 15th Flr.
New York, NY 10001
Phone 212 631-7563
Fax 212 947-5069 Voice mail pager 917 490-8407
Domino.cubes@excelsior.net

LISTing Newsletter

The Long Island Sinclair/Timex Users Group

Robert Malloy, Treasurer
412 Pacific St.
Massapequa Park, NY 11762

NESQLUG NEWS

New England Sinclair QL Users Group
Ed Kingsley, Editor
16 Highland Avenue
Saugus MA 01906
(781) 233-3671 EdK4@aol.com

QL Hacker's Journal

Supporting All QL Programmers

Timothy Swenson, *Editor*
2455 Medallion Dr.
Union City, CA 94587-1914
swensontc@geocities.com
http://www.geocities.com/SilconValley/Pines/5865/

CATS Newsletter

The Capital Area T/S Users Group
Barry Washington
7044 Cindy Ln
Annandale VA 22003
301 589-7407
BBS 301 588-0579
Internet mf0002@epfl2.epflbalto.org

Alec Carswell

Motivation

16 Montgomery Ave
Beith Ayrshire KA15 1EL
United Kingdom

Sinclair Resources

Jack Dohany (Developer - 2068)
627 Vera Ave
Redwood City CA 94061

John McMichael (Developer - Graphics)
1710 Palmer Dr
Laramie WY 82070

Keith Watson (AERCO & Z80 Emulator)
41634 Amberly Dr.
Mt. Clemens, MI 48038

Steamboat Lake, CO

Rod Gowen (RMG)
14784 S Quail Grove Cir
Oregon City OR 97045-8843

Send them a LSASE and ask for information about their current products and/or services.

Surplus T/S Inventory

JOHN J SHEPARD III
281 130th ST
OGDEN IA 50212
< jshepard@netins.net >
Mostly QL & TS-2068

JACK BOATWRIGHT
67325 FRYREAR RD
BEND OR 97701
< jboatno4@outlawnet.com >
Mostly ZX-81/TS-1000 & TS-2068

QL TS-2068 ZX-81
Books Magazines
Software