DAL-ACE

Dallas Atari Computer Enthusiasts

Volume 10, Issue 6 June, 1989

Table of Contents

MEETING DATES: JUNE 10, JULY 8.

Thank you Angela

First of all, I want to say a special thank you to Angela Burns who recently resigned as newsletter editor. I'm sure you'd agree that she has done an excellent job producing a quality newsletter and keeping us informed.

Now, we need a new volunteer. On a temporary basis, Jeff Golden and I will co-edit the newsletter but I STRESS TEMPORARY!

So here's your chance to be creative. You can do it. Just volunteer.

Signed,

Anita Uhl
VP Communications

Highlights of May Meeting

For those of you who were unable to attend the May meeting, the following is a synopsis of what you missed:

8-bit SIG

Eb Foerster demoed a new library disk, (# 109), CITYEDIT. This disk is a disk sector editor with exceptional capability in the area of disassembly.

Scott St Clair did a fine job of flying his helicopter during a demo of the helicopter simulator game TOMAHAWK.

Jeff Golden gave a brief demo of Analog disk # 72 which includes a graphic conversion program that converts Print Shop, Newsroom, and Graphics 8 formats into each other. A-72 also includes a DOS command file processor and an assembler game called CLOWN.

Jeff demoed the new club Forth disk, (# 108). A discussion was held on a planned Forth class to be started this month on the BBS. More on this later in the newsletter.

Main Meeting

Morris Stevens informed the group about the death of Mr. Hardesty, the owner of Computer Skills, and of the closing and liquidation of the Computer Skills business in Bedford. Morris is assisting the widow of the owner, and much of the ST inventory was on sale in the vendor area. He expects to have additional inventory for sale at the June meeting.

Rene Tucker gave her usual sterling treasurer's report. Dalace is still alive and well and is growing stronger financially.

Anita Uhl asked for a volunteer to take over as newsletter editor. You have already read about the large number of people who jumped up to volunteer. Come on folks, editing the newsletter is one of the fun jobs the club has to offer.

After a brief question and answer session, the meeting adjourned for the big event of the day, the annual auction.

Used Equipment Auction

The auction, as usual, was a big success, thanks to our intrepid auctioneer, Marc Salas, and the huge inventory of equipment for sale. Items placed on sale included such goodies as an 8-bit MIDI interface, Computer Eyes, and a 1050 Happy disk enhancement package. ST and 8-bit software was there in abundance, including many never-opened packages. If you missed the auction, you missed a great opportunity to acquire that missing item that you always wanted, but could never afford. The prices were right.

The early report from our treasurer, Rene Tucker, is that the club netted more than $500 from the auction. Now! that should help our May financial report. The auction lasted well into the afternoon, so the remaining SIG meetings were canceled.

Award to Jeff Golden

Recently the Dal-Ace Board decided to acknowledge members who have contributed exceptionally of their services to the club. This will be done by a quarterly presentation of a certificate of recognition.

The first recipient of this award is Jeff Golden. At the May meeting Donny Arnold presented a certificate of recognition to Jeff.

Among his current contributions to the club, Jeff runs an 8-Bit Sig at the Infomart before the main meeting and also a monthly self-help class at his home.

Notes from DAL-ACE Board of Directors Meeting 5-20-89.

Attending: Rene Tucker
Randy Randolf
Marc Salas
Donny Arnold
Brenda Arnold
Terry Borchardt
Dave Gramm
Anita Uhl

Meeting was called to order by the President at 9:15AM.

Treasurer gave a financial statement.

MAL Gramm reported that the member survey is in the May issue of the newsletter.

Ad Manager reported that Computer Skills still owes the club for advertising--vote in favor of collecting the debt.

Auction was discussed: 60% of proceeds to go to the donor; 40% of proceeds to go to the DAL-ACE treasury.

Meeting was adjourned by the President at 9:40AM.

Hi Ralph!!

by Harold Lewis

Dear Ralph Salmeron,

I guess you could say it is about time I got on the board and said something to someone. I hope you can read this because, it is my first message to ever leave on any BBS.

I am using a word processor called "Fleet System II" to write this letter. I know I should be using "Tex-Pro" but, I have been using the Fleet System for a long time and feel more comfortable with it.

It has a lot of great features such as 40, 80, & 120 column printing, automatic word wrap, an 80,000 word spell checking system and a whole lot more.

First of all I would like to thank the people & officers of Dal-Ace for your help, friendship, & acceptance to the club. A special thanks to Terry Borchardt for introducing me to Dal-Ace. Also, a special thanks should go to Donny Arnold who worked with me to get the right Harold Lewis verified, sorry about that Donny.

The way I first heard about Dal-Ace was when I purchased my modem from Federated. The guy in computer sales gave me a large envelope and said "Compliments of Dal-Ace". I thanked him and hurried on home because, I couldn't wait to get the Modem (XM301) hooked up.

After getting the modem installed, sure enough, I wanted to call a BBS. That's when I picked up the large envelope and opened it up. There it was, a Dal-Ace news letter & a phone number to call my first BBS.

I keyed in the number on the computer, the phone began to ring, my heart beat picked-up a little, then an operator came on and said that number was no longer in service, "Rats", foiled again. So then I began to scan over the Dal-Ace news letter, "Hey", here's a phone number with the same exchange as mine, I'll give it a call.

Sure enough, it was Dal-Ace's own Terry Borchardt. From that point on I was stepping in the right direction. Terry just lived down the street from me & made it easy for us to get together. Since that time, Terry & I have become good friends.

So far, I have not missed one Dal-Ace meeting. Every time I go, I always learn something and meet a lot of nice people. "DAL-ACE IS GREAT", and has helped me to understand more about the Atari computer.

By the way Ralph, you were right about basicon & basicoff com's. It looked like it worked ok, because while in DOS 2.5 you could load basicoff and try to go back to basic and the computer would say "No Cartridge Present". After that you could load basicon.com, then go to basic & get the ready prompt. At that time I loaded a basic program and it did. After running the program I tried to go back to DOS and it happened. The computer locked up completely. I tried everything, even hit the reset key, but that didn't help either. So the only thing left to do was to turn off the computer and start over again.

I don't know what the problem was and worse of all I didn't know how to correct it.

This communication on a modem is a new experience for me. There is a whole new world out there. It was as if I opened a doorway in which you can gain a wealth of knowledge. After all, that was one of the main reasons for buying a modem. I didn't realize there was so much information out there.

I only wished Atari would have helped out a little more. You just can't go into any computer store and find Atari products any more.

I guess it's up to the Atari computer users to keep it going. Thanks to Dal-Ace who supports the Atari computer user & helps keep things alive for all of us.

I hope this letter hasn't been to boring for you all. But, I am not a writer at all. I must be going for time is getting short. See ya'll at the next Dal-Ace meeting which I believe is May the 20th.

Here's Signing Off
Your Friend,
Harold
*** Live Long and Prosper
*** BY!

(Ed. Note) Harold, you are a much better author than you are giving yourself credit for. We hope that you will continue to tell us about your experiences from the viewpoint of a new Atari user.

I know of one member who puts in a great deal of effort putting together those "Starter Kits", both in duplicating the disks and running all over town, and that person will certainly be glad to read your comments.

This is the first time I ever heard of Fleet System II, so believe it or not you are already educating the rest of us. Maybe you could write up an in-depth review on the program.

Thanks again on behalf of the club for all those words of appreciation.

NEW ST LIBRARIAN

Some of you may have noticed that Gary Loges was manning the ST Library at the May meeting. Gary, bless his soul, has offered to step into the spot left vacant when Angela had to cut back on her many club activities. Both Gary and Angela deserve a big hand for helping the club to support the ST.

THE WICO COMMAND CONTROL

by Ralph Tenny

Although it may not seem like it, the WICO Command Center is definitely a "productivity" peripheral for your Atari ST.

This device is a mouse-compatible trackball which has a place for the the mouse to be plugged in.

Why would you need the mouse with a functional trackball? You don't - the trackball has two buttons, different enough that you can tell them apart by feel; it will do everything a mouse will do.

However, dragging scroll bars single-handed is awkward and would require a lot of practice. But think about this: hold the trackball in your lap, working the buttons with your left hand and the trackball with your right.

For jobs done entirely via mouse, this is the most stress-free computer control I've seen. The "left" button on the Controller is large and in front of the other button, and very comfortable to use. The smaller button functions as the "right" mouse button for those functions requiring both.

With a lap-mounted trackball, it is easy and natural to run Easy Draw - the only software requiring dual-button operation I am sufficiently comfortable with to make a valid comparison.

As for myself, I normally use the mouse only for those things which don't have keystroke combinations and for those functions I haven't learned the keystrokes for. My desktop is messy and piled up, with stuff often cascading down onto the mouse pad, creating distractions.

The Controller is tall enough that clutter doesn't bother it, and it doesn't have to move. It occupies the normal mouse position, and the mouse is stationary on the left side of the keyboard. Just as with lap operation, the controller is essentially a two-handed operation for best efficiency. In either case, I found the learning curve to be very short.

The key functions on the mouse are reversed when it is plugged into the Controller, but this seems quite natural for my left hand. A normal menu operation (for example, with WordUp that I'm using to write this) goes like this:

1. Spin the ball toward the pull-down area and tap it to stop close enough to fine-tune the position.
2. Tap the mouse key to make the selection.

For click-and-drag operations, spin to the starting point as before, press and hold the mouse key and move the ball to drag. Note that double-clicking is easier with the mouse key than with either button on the Controller, because the springs are stronger on the Controller buttons.

In the first example, either single-handed Controller operation or two-handed operation seems equally natural, and are interchangeable for me. In the second case, two-handed operation (with the mouse or with lap-based controller) is very natural and faster.

Since I am almost totally ambidextrous, I have no problem with dual-hand coordinated operations. I have no idea how easy or hard it would be for a strongly-handed individual to learn similar operation. I suspect that left-handed persons would be more comfortable using the same mouse/trackball arrangement that I use, and that right-handed persons would find that left-handed trackball operation would be easier to learn.

The bottom line is that I could see some speed improvement from two-handed operation over mouse-only operation almost immediately. It only gets better with practice, and I think I would expect at least 15% or more improvement with opportunity for steady practice. At a list price of $49.95, I count the Wico Command Controller as a worth-while investment.

MODIFYING THE SEGA LIGHT GUN FOR ATARI USE

by Farmer Jeff

(Article originally in Dec 88 SCVACE Newsletter, and reprinted by ACORN)

Parts:
1 - SEGA Light Phasar
Federated $25.97
1 - Joystick Extension Cable
Radio Shack $3.99
Cat-no 270-1705
1 - SPDT Lever type switch GC
Electronics 35-844

Buy the SEGA Light Phasar at the best price you can find. Federated's price is given for comparison). Remove the six screws (one is under the SEGA Logo. Lift the right side of the plastic logo with a sharp pointed screwdriver or knife).

Open the gun and find the trigger mini-lever switch with two contacts. Lift the switch from its posts and look on the back, mine was marked Matsushita, one contact is marked "C" the other "NO" (normally open).

Mark both wires so you know which is which and use a soldering gun to remove the two wires from the switch.

Because the SEGA joystick connector is not compatible with Atari's, you will have to cut it off, strip back the wires and solder it onto an Atari Joystick connector.

Most joysticks don't have wires for unused signals, so cutting up an old joystick cable may not work. Specifically, an Atari Joystick does

not use the 5 volts, so there is no wire connected to pin 7. Radio Shack has a joystick extension cable that has all nine pins wired from male to female. Cut off the male connector, strip back the wires and connect as follows:

SEGA GUN ATARI
JOY-STICK PORT

Blue wire Pin 1 Stick Forward
Gray wire Pin 6 Trigger
Green wire Pin 7 +5 volts
Black wire Pin 8 Ground

Looking at the female connector:

```
Pins 5 4 3 2 1

   _______
   \0 0 0 0 0/
    \0 0 0 0/
     -------
pins 9 8 7 6
```

Since wires on the Radio Shack extension cable may be different colors, you may want to check them yourself to be safe, but on my cable they were as follows:

White wire - Pin 1 connect to Blue wire on SEGA Gun
Orange wire - Pin 6 connect to Gray wire on SEGA Gun
Red wire - Pin 7 connect to Green wire on SEGA Gun
Black wire - Pin 8 connect to Black wire on SEGA Gun

The SEGA gun's wiring is opposite of the Atari causing the gun to fire when the trigger is released. To change this so it fires when the trigger is pressed we must replace the normally open (NO) switch we removed from the SEGA with a normally closed (NC) switch. Take the switch with you to your favorite electronics stores (Radio Shack only carried one that was too small). I found one at Electric City that was the exact same switch but with all 3 contacts (C,NO,NC).

Solder the wire you marked "C" to the C contact on the new switch. Solder the wire marked "NO"to the Atari compatible NC contact (Normally closed). Put the gun back together, screw in the screws and you're ready to try it out.

The Atari light gun uses the light pen support built into the Atari computer. The light pen horizontal position is in memory location 564 and the vertical position at location 565. Try this simple program:

```
10 LPENH=564:LPENV=565
20 GRAPHICS 0
25 POKE 752,1:POKE 712,15
30 POSITION 0,0
40 FOR I=1 TO 4
45 ? "0123456789";:NEXT I
50 ? "HOR= ";PEEK(LPENH)
60 ? "VER=";PEEK(LPENV)
70 ? "TRIGGER=";STICK(0)
80 GOTO 30
```

Light pen horizontal readings range from 0 to 227. Point your gun at the left side of the screen. You will notice that the reading is about 88. Move the gun across the screen to the right, the reading will reach 227 then suddenly drop to 0 then increase again until about 30 at the right side. This offset is due to the delay between when a pixel is actually lit on the screen and when the information is relayed to the light gun sensor to the pokey chip, which latches an internal scan counter for the pen reading. Move the gun from top to bottom. The vertical readings will move from about 17 to 115. Press the trigger and the values will change from 14 to 15 when the trigger is pressed. You are now ready to try your new light gun on your favorite program such as Barnyard Blasters, or Bug Hunt.

Also there are public domain programs available from our club:

CHEAP SHOT - display several different targets to shoot at.

TARGET.COM - a simple + type target with very nice sound and graphics.

DRAW.GUN - a drawing program using the gun.

Enjoy your SEGA- ATARI light gun and remember to keep it out of the reach of children, or you'll never get any peace and quiet in the house!

SCVACE, ACORN and Farmer Jeff disclaim responsibility for any damage that might occur during improper implementation of this, or any, hardware modification project we may publish.

Editor: The same disclaimer applies to Dal-Ace.

DON'T FORGET TO TURN IN YOUR COMPLETED SURVEY!

Either mail it in to the P.O. Box on the back of your newsletter or give it to a Dal-Ace officer.

Wanted:

ST Monochrome monitor. Call 739-3116 and leave a message. Michael Duke

TURBO ST

Reprinted from JACG newsletter
Author - John H. Dean

The Atari ST comes equipped with TOS (The Operating System), and GEM (Graphics Environment Manager). If you have used the mouse and Desktop for more than a few minutes, you will know that GEM is intuitive, powerful, and easy to use. GEM is one of the principal reasons most users decide to buy an Atari ST.

But GEM was not written for the ST TOS. The routines in GEM that write characters to your screen, clear it, and scroll it, do not fully exploit the speed of the 68000 CPU. GEM isn't as fast as it could and should be. There are two ways to overcome this limitation. You can (1) add a special piece of hardware that takes over part of GEM's job, or (2) you can rewrite the slower parts of GEM with more efficient code. Atari's blitter chip represents method #1. Turbo ST represents method #2.

What this means is that Turbo ST can speed up most Atari programs that write text to your screen. For example, Data Manager ST displays new pages 81% faster, GFA BASIC 69% faster, and ST Writer 116% faster, when these programs are run with Turbo ST, according to the manual that accompanies the copyrighted program from Softrek. With few exceptions, Turbo ST does for programs running on your Atari 520 or 1040 what Atari's "blitter chip" does for programs running on a Mega ST -- but it usually does so even faster.

Turbo ST is installed as an accessory from your boot disk, either floppy or hard. You can then run your favorite programs just the way you always have -- there are no special instructions or complications. And once installed, should you ever wish to turn Turbo ST off, simply pull down the desk accessory menu on you Desktop, and click "REMOVE". Click on "INSTALL" to get it back.

The quickest and simplest way to test Turbo ST is to call up a list of files from the Desktop, and "Show as Text". Now, switch between "Sort by Name" and "Sort by Type" with Turbo ST switched OFF. Now do the same thing with Turbo ST switched on. The increase in speed using Turbo ST is strikingly obvious, especially when the list of files is long.

Turbo ST unfortunately does not speed up all programs. Since it achieves its increase in speed by enhancing the performance of GEM and TOS, it cannot accelerate programs which, like pc-ditto, replace GEM with their own operating systems. Nor can it speed up programs which spend a great deal of time on operations other than text display, such as those which perform intensive math operations. Similar limitations, of course, apply to Atari's hardware blitter as well. Neither does Turbo ST speed up most games, which typically bypass GEM and access the 68000 chip directly via assembly language.

There are a few limitations to using Turbo ST. It is recommended that ProCopy be used only when Turbo ST is turned off. Apparently, speeding up the sector write status display can throw off the internal timing of ProCopy when it is copying a disk. ST Writer with Turbo ST on a color monitor will produce a slightly compressed screen, since ST Writer directly changes the line spacing. And, when starting up GFA BASIC in low resolution, the hardware register that controls screen resolution is apparently changed directly. Since Turbo ST does not see this, the screen display is garbled. If GFA BASIC is started in medium or high resolution, it works fine.

The authors of Turbo ST point out that due to the way GDOS changes workstation handles, it is impossible to insure 100% reliable operation with GDOS installed. With G+PLUS from CodeHead Software (see article on G+PLUS) there should be 100% reliable operation.

Turbo ST is published by SofTrek, P.O. Box 5257, Winter Park, Fl 32793.

Programming is by Wayne Buckholdt, and the manual was written by Dick Biow.

8-BIT DISK LIBRARY

by Ralph Salmeron

Well, this month we've got a bag full of 8-Bit disks for the library.

We've got something for everybody, starting with DISK #107, Atari's latest, DOSXE. The boys from Sunnyvale finally have released their new dos for distribution by users groups so here it is.

In case you haven't heard by now, this is the long awaited dos written to support the XF551 double sided/double density disk drive. If you hung around for the 8-BIT WORKSHOP after the main meeting back in April, you were fortunate enough to catch a 30-min. introduction to it by Dave Gramm. If you didn't well, you'll just have to pick up the disk and explore it for yourself. Here's a couple of the new features you will find:

DOSXE is both a menu driven AND command line processor type dos.

It works with the XL/XE computers (sorry 400/800 owners) and 810, 1050 and compatible drives (Indus, Astra, etc.), in addition to the XF551.

It provides for multi-density support, sub-directories, date stamping, and high speed I/O transfer for the Xf551.

It supports DOS's 2.0/2.5, albeit, in a slightly less convenient matter).

The "lite" version of the docs are included on the back of the disk, but if you want the full version you'll have to write off to Atari at:

Atari Corp.
1196 Borregas Avenue
Sunnyvale, Ca.94806
(408)745-2000

Include $10 and you'll receive the complete spiral bound DOSXE docs, something which will probably be worthwhile if you really want to get your money's worth out of all the features in this DOS.

Next is DISK #108, a PD version of the Fig-FORTH language. This is a two-sided disk with a very good and very extensive tutorial built right in. Jeff Golden has offered to conduct a FORTH language class on the club BBS if enough interest was expressed by members...SOooo, if this sounds like something you guys (& gals) would like to get into let Jeff or myself know and we'll set it up.

Disk #109 is the very professionally done CITYEDIT, a disk/sector editor written by the people at the CITY BBS, that's just loaded with all the features you'll need to not only take an in-depth look inside your disks, but to repair and modify them too. ASCII/ATASCII/INTERNAL CODE, view, edit, write, disassemble, map a disk, etc. If you were at the May 8-Bit SIG then you saw Eb Foerster put this program thru the paces, and if Eb liked it, that's good enough for me. This is a must if you want to learn more about your disks.

Well that's it for this month, of course, the current ANALOG disk is available as usual and there's plenty of older disks that deserve a look too, later!

The Creed

The following has been reprinted and reprinted and reprinted from ACORN, The I/O Connector, Atari Interface Magazine, and the BASE users group. Thank you to all

In the beginning there were fingers. And then there was an Abaçus. And it was a start.

At this point in history, it was a long wait for silicon. We had to wait for a lot of small things like Ben Franklin frying himself on a kite, Marconi, Jobs, the WOZ and Bushnell.

And lo and behold, Bushnell discovered the Egg. And the Egg was called "PONG". And thus the Home Video Game was born. And this added to the power that had been laid down by the Great Hackers That Came Before.

And finally Home Computers were born. And it was good.

But there was a dark cloud.

Once in their new homes, the home computer gave birth to the user. And the users were alone in the wilderness. And they cried into the void. Finding no answer, they found themselves.

And User Groups were formed. And it was good.

And they banded together. Tribe unto tribe. Each seeking its own maker. Apple unto Apple, Atari unto Atari, Commodore unto Commodore, IBM and its clones, and Orphan unto Orphan.

A lucky few tribes found their makers. Most found nothing. Others were abandoned and know so. But still they came together. And it was good. The User Groups survived when the makers had forsaken them.

The User Groups came together in the dark and will be there until the door of obsolescence squeaks its hinges for the last time.

If you trust in the maker and he forsakes you, turn to your brethren. Seek ye out thy brethren for they shall help you find the light.

Yea, there are many soothsayers and false prophets that shall cause many doubts in your path and faith. Put your faith in your brethren for they have not designs on your soul (or wallet).

We banded together because we had no choice. We of the silicon tribes do not bow down to any self-serving God. For we will survive as we have always done, relying on each other and ourselves. Do not lie to us, for we talk to each other more than you will ever realize. Demi Gods come and go, trademarks last longer, but even they can fade. Above all, we demand respect, for we are not the children of the computer industry. We are its FUTURE!

FORTH LANGUAGE

by Jeff Golden

Quoting from the May issue of Computer Shopper:

"Because major development time and cost savings result from its ability to function both as a prototyping language and as an implementation language, Forth is considered a proprietary edge and trade secret by many software industry leaders."

The article goes on to state that many popular products, such as Ashton-Tate's RapidFile, Paperback's VP-Planner and IBM's PC-CAD are written in Forth.

There are a large number of other products on the market that are written in Forth, but again the origin of the code is kept secret by the software houses. Some of the telltale signs are proprietary disk file formats, availability on many systems, and lower than usual prices. Lower prices? Read on!

From my personal experience of many years in the programming business, (have any of you ever heard of SOAP), it never seemed to make a whole lot of difference what language you used. You still have to tell the computer to "MOVE A to B", and it takes a program statement to do it.

What I'm trying to say is that there was never any real significant difference between the languages, in terms of programmer productivity, be it Assembler, Cobol, Basic, C, PL/1, Pascal or what have you. That is until one day when I tried Forth.

Yes, you still have to move A to B, but it is the weirdest sensation to suddenly wake up and discover that your program is done, weeks before you expected to be done.

After having experienced it myself, I found that I was not alone. Others had noted the weird sensation as well, and had written about it in Forth Dimensions, the newsletter of the Forth Interest Group, (of fig-Forth fame).

The inquiring mind immediately asks, "How can this be"? "How, after all those other languages, can a language allow me to write a program in 1/4th of the expected time?

Programmers are notorious for underestimating how long it will take to write a program. Here is a language that reverses the situation. If the secret ever gets out it might put us all out of business.

Back in the early 70's Nicholas Wirth and his cohorts discovered the value of "Structured Code".

Structured code is a method whereby the code is written in small, relatively independent, modules, with one entrance point and one exit point.

The small modules are then strung together, either in a linear fashion, or, most often, in a hierarchical tree structure. The tree has an advantage in that the root, or main module, controls the flow, and if properly commented, serves as a road map to the program.

Now what does all this have to do with the speed of writing in Forth. Well, Forth is a highly tree-structured language. Its modules have one entrance point and one exit point, and are very very small, usually fewer than 16 lines of code.

While you can easily write garbage code in Forth, (as in any language), you are almost forced into following the rules of good structured code. Just what old Nick ordered.

Forth incidentally, was invented way back in 1969, before anyone ever heard of structured code which came into vogue in the 1973-75 timeframe. Maybe old Nick was looking over Charles' shoulder.

Charles Moore originally invented Forth after he became dissatisfied with the then available languages for the PDP-11 minicomputer that was used to drive the telescopes at the Kitt Peak National Observatory.

Forth is still being used today for stand-alone industrial control applications. That computer under the hood of your car may very well be driven by Forth.

Forth has one more ace up its sleeve that contributes to high programmer productivity. Forth is interactive, (like Basic), but since it is organized as independent modules in a tree structure, and since each module can executed from the console, it is very easy to thoroughly test the tiny modules one at a time. (Believe it or not you can actually write similar structured code in Basic, or any language, and end up with many of these advantages, but Forth kind of forces you into doing it right.).

With Forth, when something is wrong with one of those tiny modules, you only have a few lines to be concerned with, so error correction goes very fast, and once a tiny module has been proven to be functioning correctly, you can forget it, and move on to the next. The net result is that, before you know it, you have an error free running program.

Perhaps now you have some understanding of that "Oh-My-Gosh" feeling one gets when you suddenly realize that the tiny module you just tested was the root segment, and the program is done.

Moving on, a question that is quite often asked, involves what kind of applications can be written in Forth. The answer is anything.

Forth is an extendible language. Whenever the language does not support something that you want it to do, you simply write another one of those tiny modules and add it to the language.

Standard vanilla Forth is exceptionally good for writing text-based applications such as word proces-

sors, spread-sheets, data bases and adventure games.

Since the code is very compact, (even more than assembler code), it is exceptionally good for ROM based applications, where memory is scarce and error free code is important.

Throwing out the production ROMs, because of an error in the code, can be expensive.

Standard Forth applications are easily ported between various computers of all types. Forth is quite often the first third-party high-level language that is available for a new computer, and it is available for just about anything.

The ST users and the 8-bit users finally have something in common. We can talk about Forth and actually trade programs via modems. The club library has PD Forth versions for both machines.

The reason for the availability of Forth lies in the fact that any reasonably skilled programmer can actually write a Forth compiler from scratch. The language and the knowledge is in the public domain, and for the most part, Forth is written in itself, except for a very small kernal of machine-specific modules. Again, the modem plays a part in allowing the bulk of the Forth code to be transferred to a new machine.

When I first started toying with the idea of starting up a Forth class on the BBS, serious consideration was given to actually taking the class through the building of the system from scratch, and we may someday do just that, since I really want a Forth that is a bit more sophisticated than the current crop of PD Forths. However, for now we will stick with the public domain versions that we have available for both the ST and the 8-bit.

Now that you have been told all of the good things about Forth, let's talk about some of the things that many people feel are not so good.

Forth is a very strange language, unlike anything most of us have ever seen before. It really takes some "getting used to", and many people will throw up their hands, (if not their dinner), and walk away at the first sight.

Forth is different. If it were just another copy of a similar language, then there would be no advantage in using it. If you approach the language with an open mind, and accept what you find without question, and with the knowledge that some real good will come out of it, then we are home free.

Now that we are all warmed up to the subject,we might as well get the worst of it over first.

Perhaps the hardest part to come to grips with is the apparently backwards way that Forth statements are written. In high-school algebra, we learned that mathematical equations were written in the following fashion:

C = A + B

In Forth we say:

A B + C !

Would you believe that the first equation is the backwards one? Before we can do anything with C, we first have to find A, and before we can do any adding, we first have to find B, now we can add, but what do we do with the answer,? Oh yes, way back there in the beginning the program told us to store the answer in C.

Computers simply do not work that way, they only move forward, and a compiler that accepts the first statement really has to stand on its head to remember C.

The backward-ness of the 10th grade equation becomes even more apparent when it becomes necessary to add nested parentheses in order to compute a more complex equation.

In a Forth equation, the order of the elements determines the order of precedence, and there is absolutely no need for parentheses.

It's simple, elegant, and requires very little code to process. Little wonder that Forth when compared to Basic, or C, runs like a bat.

Another interesting point is that the terms A and B, in the above Forth statement, can actually be Forth words that invoke complex computations on their own. For instance, "A" might invoke a routine that checks the current temperature at several points and computes an average. That average plus some constant "B" could affect the adjustment of valve "C". Mind-boggling.

An advantage of Forth lies in the fact that parameters are passed to the subroutines on the top of the hardware stack. There is no need to generate separate lists of parameter addresses every time you call a sub-routine, therefore the compiler in creating a Forth program only needs to store the address of the subroutine in the program code. The subroutine already knows that its input will be on the stack.

A compiled Forth program then becomes simply a string of two-byte subroutine addresses with a few literals sprinkled in between. We told you it would generate very small programs.

(Note some ST Forths use strings of four-byte addresses).

When the program is running, a small highly optimized routine, called the interpreter, scans the compiled code and branches to each subroutine address in turn.

Each subroutine pulls its parms off the stack and when it completes, places the result(s) of its function back on the stack where it is available to the next subroutine.

The programmer has a big hand in all this since it is his job to place

the calls to the subroutines in such an order that the required parms are always on the top of the stack.

Along about now we need to discuss how to call a subroutine.

Subroutines in Forth are called "Word definitions", (usually shortened to "words"). Word actually refers to the name of the subroutine and can be a character string of from 1 to 30 characters that is ended by a space.

In a few early Forth versions, only the first three characters were significant. This out-dated limitation is still being carried over into today's standard Forth words, and contributes highly to the strange look of the language.

We recommend learning and sticking with the standard Forth words, but, for new user-defined words, we recommend using uppercase words of sufficient length to be meaningful.

Forth is case sensitive, (little a is not the same as a big A). While programming in Forth, be sure to keep your caps-lock key locked. It will save you a lot of frustration.

In the Forth statement that we used before: A B + C !

A, B and C are user-defined words, and would you believe that + and ! are standard Forth words. They all meet the requirement of at least one character followed by a space. The + character is called "plus", and the ! character is called "store". How does that grab you. (You learn something new every day, don't you.)

The operation of the above statement goes something like this: The A word routine is called and after doing its thing places a result on the stack. The B word routine does its thing and places its result on top of the result from the A word. The + word routine removes the two items from the stack, adds them together, and places the sum on the stack. The C word, (remember it was user-defined), places the address of the memory location where we want to store the result on the stack. And then we finish up with the ! word whose function in Forth is to pull the address and the sum off of the stack and "store" the sum away at the memory address.

Sound complicated? Not really. It all becomes second nature more quickly than you might imagine to the point were you no longer have to think about it.

Now how do we define a word. With another Forth word of course. Actually there are several "defining words" in Forth, but right now I want to talk about the most commonly used one, the "colon definition".

A colon definition is used to define an executable routine. It starts off with, (you guessed it), a colon followed by a space. Following the colon we type in the name of the new routine. And after that we string out the names of the Forth words that when combined will perform the new function that we are trying to define. The colon definition is ended by typing a semi-colon one space beyond the end of the last word in the definition.

Ex. : TWO+TWO 2 2 + . ;

As you can see doing it is a lot easier than describing how to do it in English. The function of the above definition is to add 2 and 2 together and then print the sum (4) on the console.

The name of our routine is TWO+TWO and once it is defined to the system, then every time we type TWO+TWO (return), we will print a 4 on the console. Not a very useful function, but it serves well as an illustration.

The 2s in the above definition are literals. When we type in a number, the system will place that value on the stack.

The next to the last word in the above definition is a period, (pronounced DOT). The function of dot is to print the top value on the stack on the console.

There is one more thing that I want to cover for now. I have included in the newsletter a list of the standard Forth-83 words. The club ST version of Forth follows the Forth-83 standard. The club 8-bit version follows the old fig-Forth standard. Where the two versions differ, the word is defined twice with the fig-forth version marked with an asterisk.

To use the list, you need to know something about "stack notation", the standard shorthand way of describing what a word definition expects to find on the stack and what it will leave on the stack when it is done.

Stack notation takes the form of a comment that every good Forth programmer should include in his source. The notation for the + word appears as follows:

(w1 w2 -- w3)

The items to the left of the dash represent the condition of the stack at the time the routine is entered.

The notation indicates that the + word will require two 16-bit signed values, w1 and w2 on the stack as input. The w2 value is the top of the stack at input.

The item(s) to the right of the dash are what the routine will leave on the stack after it completes its function. Again the rightmost item, (if more than one), will be the item at the top of the stack.

In the example, the + routine will "eat" items w1 and w2 from the stack and will leave the sum w3 on the stack. w1 and w2 are no longer on the stack after the + word does its thing. Forth words will usually eat their input, but not always.

I'm now out of space. Get back to you soon with more details, and be sure to tune in to the Forth class on the BBS.

FORTH-83 HANDY REFERENCE

Information provided by FORTH INTEREST GROUP, P.O. BOX 8231, SAN JOSE, CA. 95155

GLOSSARY

Unless noted otherwise, all numbers are 16-bits.
Pronunciations where needed are included within quotes.

Stack abbrev	Number type
flag	boolean flag, 0 = false
?	truth value, either 0 or non-zero
char	character
x	any 16-bit number
n	signed number (-32,768-32767)
u	unsigned number (0-65,535)
w	wrap-around circular number (either signed or unsigned)
+n	positive number (0-32767)
addr	address (same as u, unsigned number)
apf	address of the parameter field
comp-addr	compilation address
d	signed double (32-bit) number
xd,ud,wd,+d	specifies type of double number

Attributes key:		(in parentheses following definitions)
C	Compile only	Must be used in a colon definition.
I	Immediate	Special compiler case that executes at compile time. Many immediate words will generate "run-time" components".
M	Multiprogramming	In a multi-tasking system, may relinquish control to other tasks.
	Impact	
U	User Variable	16-bit variable

Stack notation:

(inputs -- outputs) the inputs are shown to the left of the dash. The rightmost input value is the top of the stack at input time. Outputs are shown to the right of the dash. Again the rightmost output is the top of the stack at output time..

FORTH-83 WORD SET

LOGIC

NOT	(x1 -- x2)	One's complement.
AND	(x1 x2 -- x3)	Logical bit-wise AND
OR	(x1 x2 -- x3)	Logical bit-wise OR
XOR	(x1 x2 -- x3)	Logical bit-wise exclusive OR,"x-or"

STACK MANIPULATION

DUP	(x -- x x)	Duplicate top of stack
DROP	(x --)	Discard top of stack
SWAP	(x1 x2 -- x2 x1)	Exchange top two stack items
OVER	(x1 x2 -- x1 x2 x1)	Make a copy of second item on top
ROT	(x1 x2 x3 -- x2 x3 x1)	Rotate third item to top "rote"
PICK	(n -- x)	Zero based. Duplicate nth item to top (0 PICK is DUP, 1 PICK is OVER)
ROLL	(n --)	Zero based. Bring nth item to top. Others move down. (1 ROLL is SWAP, 2 ROLL is ROT)
?DUP	(x -- xx) or (0 -- 0)	DUP if non-zero, "question dup"
>R	(x --)	Move top item to return stack for temp storage (use caution), "to-r",(C)
R>	(-- x)	Retrieve top item from return stack, "r-from",(C)
R@	(-- x)	Copy top of return stack onto stack, "r-fetch"
DEPTH	(-- n)	Count number of items on stack

COMPARISON

<	(n1 n2 -- flag)	True if n1 less than n2, "less- than"
=	(w1 w2 -- flag)	True if w1 equals w2, "equals"
>	(n1 n2 -- flag)	True if n1 greater than n2,
0<	(n -- flag)	True if n is negative, "zero less"
0=	(w -- flag)	True if w is zero, "zero-equals"
0>	(n -- flag)	True if n is greater than zero
D<	(d1 d2 -- flag)	True if d1 less than d2, "d-less- than"
U<	(u1 u2 -- flag)	True if u1 less than u2, "u-less- than"

ARITHMETIC

About Division: In FORTH-83, the quotient is floored (rounded to negative infinity). The modulus (remainder) has the same sign as the divisor.

+	(w1 w2 -- w3)	Add w1 and w2, "plus"
D+	(wd1 wd2 -- wd3)	Add double numbers, "d-plus"
-	(w1 w2 -- w3)	Subtract w2 from w1, "minus"
1+	(w -- w+1)	Add one to w, "one-plus"
1-	(w -- w-1)	Subtract one from w, "one-minus"
2+	(w -- w+2)	Add two to w, "two-plus"
2-	(w -- w-2)	Subtract two from w, "two-minus"
*	(w1 w2 -- w3)	Multiply. "times"
UM*	(u1 u2 -- ud)	Unsigned multiply, double result
UM/MOD	(ud u1 -- mod quot)	Unsigned division, double dividend, single divisor and results
2/	(n -- n/2)	Arithmetic right shift, "two-divide"
/	(n1 n2 -- quot)	Divide n1 by n2 signed, "divide"
MOD	(n1 n2 -- mod)	Modulus signed, (remainder of n1/n2)
/MOD	(n1 n2 -- mod quot)	Division with remainder and quotient
*/MOD	(n1 n2 n3 -- n4 n5)	Multiply then divide with double precision intermediate, n1*n2/n3
*/	(n1 n2 n3 -- quot)	Like */MOD but give quotient only

Note: Handy approximation (better than 10 to the 7th) pi as 355 / 113, *e* as 25946 / 9545, sq.rt. 2 as 27720 / 19601, sq.rt. 3 as 32592 / 18817, sq.rt. 10 as 27379 / 8658

MAX	(n1 n2 -- n3)	n3 is the larger of n1 and n2
MIN	(n1 n2 -- n3)	n3 is the smaller of n1 and n2
ABS	(n -- u)	if n is neg, u is n's twos-complement
NEGATE	(n1 -- n2)	Two's complement
DNEGATE	(d1 -- d2)	Double precision two's complement

MEMORY

@	(addr -- x)	Fetch 16-bit value at address, "fetch"
!	(x addr --)	Store 16-bit value at address, "store"
C@	(addr -- byte)	Fetch 8-bit byte at address, "c-fetch"
C!	(byte addr --)	Store 8-bit byte at address, "c-store"
+!	(w addr --)	Add w to 16-bit value at address

STRINGS

CMOVE	(from to u --)	Move u bytes starting at address-from to address-to, "c-move"
CMOVE>	(from to u --)	Like CMOVE, used to slide string to higher addresses, "c-move-up"
FILL	(addr u byte --)	Fill u bytes at addr with byte
COUNT	(addr1 -- addr1+1 byte)	Move string count from memory onto stack. Also used as C@ with auto increment.
-TRAILING	(addr +n1 -- addr +n2)	Modify string count to exclude trailing spaces

NUMERIC CONVERSION

BASE	(-- addr)	Contains radix for number conversion
DECIMAL	(--)	Set number base to decimal
CONVERT	(+d1 addr1 -- +d2 addr2)	Convert string at addr1+1 to double number. Add to d1 leaving sum d2 and addr2 of first non-digit.
<#	(--)	Start output string number convert
#	(+d1 -- +d2)	Convert next digit of d1 to string
#S	(+d -- 0 0)	Convert remaining digits of d to string
HOLD	(char --)	Insert edit character in string, $., etc
SIGN	(n --)	if n negative insert sign in string
#>	(xd -- addr +n)	Drop xd, terminate numeric convert, leave string addr and count for TYPE

TERMINAL INPUT-OUTPUT

.	(n --)	Display signed number, "dot",(M)
U.	(u --)	Display unsigned number, "u-dot"(M)
." text"	(--)	Compile/display text messages, (CIM)
.(message)	(--)	Display msg from input stream, (M)
CR	(--)	Display new line (Carriage retrn),(M)
EMIT	(char --)	Display char. Cntrl chars not portable
TYPE	(addr +n --)	Display a string +n bytes long, (M)
SPACE	(--)	Display a space, (M)
SPACES	(+n --)	Display +n spaces, (M)
KEY	(-- x)	Get a byte from terminal. Often used: KEY 127 AND - remove hi bit, (M)
EXPECT	(addr +n --)	Get string from terminal and store at addr. Store and display +n chars or until return. System may intercept cntrl chars, (M)
SPAN	(-- addr)	Contains actual character count stored by EXPECT, (U)

FORTH-83 HANDY REFERENCE

Information provided by FORTH INTEREST GROUP, P.O. BOX 8231, SAN JOSE, CA. 95155

MASS STORAGE INPUT-OUTPUT

BLOCK	(u -- addr)	addr is a 1024 byte buffer with data from mass storage block u. (M)
BUFFER	(u -- addr)	Same as BLOCK, but the buffer contains garbage, (M)
UPDATE	(--)	Mark last referenced blk as modified
SAVE-BUFFERS	(--)	Write all updated blks to disk
FLUSH	(--)	SAVE-BUFFERS and de-allocate buffers. Used before changing disks

CONTROL STRUCTURES

DO	Used in the form	DO ... LOOP or DO ... +LOOP (C)
runtime	(limit start --)	Setup loop given index limits
LOOP	Used in the form	DO loop-body LOOP (C)
runtime	(--)	increment index and repeat loop-body until limit reached
+LOOP	Used in the form	DO loop-body +LOOP (C)
runtime	(n --)	increment index by n and repeat until index crosses boundary between limit and limit-1
LEAVE	Used in the form	DO ... LEAVE ... LOOP or DO ... LEAVE ... +LOOP (C)
runtime	(--)	Exit loop immediately
I	(-- w)	Used inside loop to get current index
J	(-- w)	Used in nested loop to get outer index
IF	Used in the form	? If true-body THEN (C)
runtime	(? --)	If ? zero branches to THEN
IF	Used in the form	? If true-body ELSE false-body THEN
runtime	(? --)	If ? zero branches to false-body.
ELSE	Used in the form	IF true-body ELSE false-body THEN
runtime	(--)	Terminates the true-body continues after the THEN
THEN	Used in the form	IF ... THEN (C)
BEGIN	Used in the form	BEGIN ... UNTIL or BEGIN ... WHILE ... REPEAT
UNTIL	(Used in the form	BEGIN loop-body ? UNTIL truebody
runtime	(? --)	Repeat loop-body until ? is non-zero
WHILE	Used in the form	BEGIN loop WHILE true REPEAT
runtime	(? --)	Repeat true-body and loop-body until the ? is non-zero
REPEAT	Used in the form	BEGIN loopbody WHILE...REPEAT
runtime	(--)	Unconditional backward-branch to loopbody
EXIT	(--)	Terminate execution of a colon definition. Do not use in a DO-LOOP
EXECUTE	(comp-addr --)	Execute the word definition associated with comp-addr

PROGRAM BEGINNING AND TERMINATION

FORTH-83	(--)	Assure that FORTH-83 is available
QUIT	(--)	Return to terminal. Parameter stack not changed. No message.
ABORT	(--)	Return to terminal. Clear parameter stack. No message.
ABORT" text"		Compile the text string
runtime	(? --)	Continue execution if ? zero, otherwise display text, display system dependent message, and do ABORT

VOCABULARIES

FORTH	(--)	Main vocabulary contained in all other vocabularies. When executed, will be the first vocabulary in the search order
DEFINITIONS	(--)	Sets compilation vocabulary to be the same as the first vocabulary in the search order
' name	(-- comp-addr)	State dumb. looks up next word in input stream, "tick" (M)
['] name	(--)	Executes during compilation to look up the next word in the input stream and preserve its compilation address as a literal. "bracket-tick", (CIM)
runtime	(-- comp-addr)	At run-time leave the compilation address of name
FIND	(addr -- addr 0)	Lookup counted string at addr, not found leave 0
or	(addr -- comp-addr -1)	if non-immediate return -1
or	(addr -- comp-addr 1)	If immediate return 1
FORGET name	(--)	Delete name from compilation vocabulary and all words after name.

DICTIONARY ADDRESSES

HERE	(-- addr)	Addr nxt available dictionary location
PAD	(-- addr)	Addr of scratch area at least 64 bytes
TIB	(-- addr)	Addr text input buffer. Not user var.
>BODY	(comp-addr -- apf)	Convert comp-addr to parmfield addr Used with "tick"ing of DOES> words

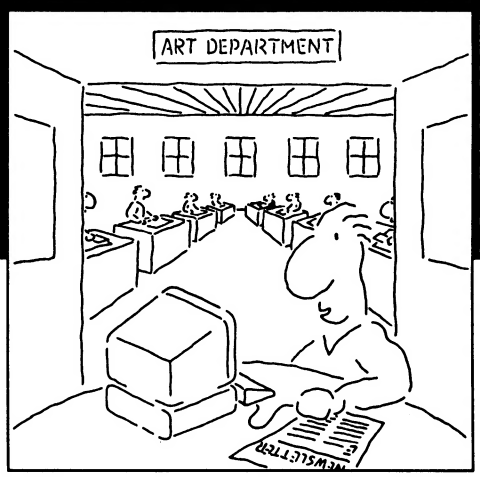
COMPILER AND INTERPRETER WORDS

LOAD	(u --)	Interpret screen U, then resume interpretation of current input stream
(comment)	(--)	Comment. Compiler ignores.
,	(x --)	Add x (two-bytes) to parm-field of most recently defined word. Sets initial values in an array. "comma"
ALLOT	(w --)	Add w bytes to the parm-field of most recently defined word. Allocates work space in dictionary
DOES>	Used in the form	: Definer ... CREATE create-body
runtime	(--)	DOES> does-body ; "does" (CI) Runtime of DOES> ends execution of Definer (after create-body). Definer could be used in the form of Definer Definer-child.
runtime effect on Definer-child		(-- apf) A definer-child executes the does-body
IMMEDIATE	(--)	Cause last defined word to execute immediately, even within a colon definition
[COMPILE] name	(--)	Compile name immediately even if name is immediate.
runtime		name is executed.
COMPILE name		(--) At run-time name is not executed but is recompiled.
STATE	(-- addr)	Content is non-zero if compiling.(U)
LITERAL	(x --)	Compile the number x (C)
runtime	(-- x)	At run-time put x on the stack
[	(--)	Switch from compilation to interpret
]	(--)	Switch from Interpret to compilation
WORD text	(char -- addr)	Get the char delimited string from the input stream and leave as a counted string at addr. (M)
BLK	(-- addr)	Contains the block number currently being interpreted, or 0 if from text input buffer."b-l-k", (U)
>IN	(-- addr)	Contains character offset into input stream buffer, "to-in", (U)
#TIB	(-- addr)	Contains the current length of TIB

DEFINING WORDS

: name	(--)	Begin colon definition of name (M)
;	Use in the form	: name ... ; "semi-colon" (C)
runtime	(--)	Terminates a colon definition
CREATE create-name	(--)	Dictionary entry with no parameter space reserved. (M)
create-name	(-- apf)	create-name leaves its address
VARIABLE var-name	(--)	Dictionary entry with two-bytes of parameter-field space reserved (M)
var-name	(-- apf)	var-name leaves its parm field addr
CONSTANT con-name	(x --)	Dictionary entry with x preserved
con-name	(-- x)	x is left on the stack
VOCABULARY voc-name	(--)	Dictionary entry for a new ordered list of word definitions (M)
voc-name	(--)	voc-name becomes the firt vocabulary in the search order

Open your own art department.



If you're a desktop publisher with big ideas and a small crew, let Migraph staff the desktop art department of your dreams.

Picture this: Professionally drawn images and illustrations at your fingertips. Powerful drawing tools, extensive editing tools, and a complete paint program at your command.

Plus high-quality printouts. Every time.

All that and compatibility, too. Migraph files load easily into your favorite Atari ST* publishing programs.

Powerful. Versatile. And easy to use. Migraph's the ideal candidate for every job in your graphics department.

Touch-Up™ The complete design tool for high-resolution monochrome images.

Easy-Draw® The professional object-oriented drawing program.

Supercharged Easy-Draw* Easy-Draw plus basic publishing features.

Easy-Tools™ A 5-tool GEM desk accessory to enhance Easy-Draw.

DrawArt Professional™ A library of over 150 professional line art drawings.

Scan Art™ A library of over 100 high-resolution, bit-mapped images.

Border Pack A library of over 40 attention-getting border designs.

QSpooler A configurable background file spooler and print buffer.

Whatever desktop graphics project you have in mind—be it big or small, simple or ornate, traditional or avant-garde—Migraph's got you covered.

See your Atari ST* dealer today for more details.

MIGRAPH™ INC

200 S. 333rd St., Suite 220 Federal Way, WA 98003 800/223-3729 206/838-4677

ST LIBRARY REPORT

DAL-Ace ST Disk #128: Tiny Pictures.

DAL-Ace ST Disk #127: Tiny Pictures.

DAL-Ace ST Disk #126: Forth disk. FORTH83.ARC and **FORTHMAC.ARC** are two public domain Forth compilers for the ST. **HISOFT.ARC** contains two demo programs using the new HISOFT Basic Professional from Michtron. One is the Towers of Hanoi puzzle and the other is a sorting demo. Source code is included. You do not need a run-time module to run these demos.

DAL-Ace ST Disk #125: Education Kid Publisher version 2.1 for young writers age 4-12; makes a 5 page document with simple graphics. It's one of the series of "KIDPRGS" from D.A. Brumleve, a professional educator and avid ST enthusiast. Can be made to Autoboot. Color only.

DAL-Ace ST Disk #124: Applications Deluxe Fontmaster ST from Germany not only helps you to design your own printer fonts, it even has features for the loading of ASCII text files and printing them out as complex documents with several styles on one line. Documentation in English and German. Monochrome only.

DAL-Ace ST Disk #123: Graphics Demos. This disk contains the **ANIMATE4.PRG** to run CyberPaint Animations and two very nice animation demos: Cola Wars and Computronics. One meg memory and double-sided drive required.

DAL-Ace ST Disk #122: Games; Canyon Bomber, Lost Treasure 1.0 is a wonderfully complex game requiring you to recover the treasure from pirates, great docs included along with the GFA source code. The author asks for a shareware donation, or that you design new levels for the game and upload them for others. **Blackjack** is in GFA Basic source and includes the GFA run-only program. All games on this disk require Color.

DAL-Ace ST Disk #121: Utilities -- Alert Maker will help you to construct GEM Alert Boxes with ease; it generates the GFA Basic code for the alert boxes. **The Card Master** is a pre-release version of a commercially available program to index your card collection, (baseball, football, Garbage Pail Kids, whatever). It's a "try before you buy" special from SofT Copy SofTware. **Database** doesn't need any explanation. If you need a relatively simple database, (not relational), then try this one. **FORM525.PRG** formats 5.25" disks. **GROUPER.PRG** is a simple mailing list generator for user's groups. **Virus Destruction Utility** and **Vkiller** are here; practice safe computing. **MEMFILE20.ACC (V2.0)** is an accessory memory/sector editor; it has been enhanced since the last version. And **MEMORY.ACC** displays the free memory in your system, includes GFA source code.

See 8-bit Disks on Pg 6-7.

DAL-ACE CLUB DISK ORDER FORM

NAME: _____

ADDRESS: _____

_____

SELECT: ___8-BIT ___ST

MEMBER:

___ Disks @ $4.00 $_____

NON-MEMBER:

___ Disks @ $8.00 $_____

prices include postage & handling

DISK NUMBERS

Mail form with check or money order to:
DAL-ACE
PO BOX 851872
RICHARDSON, TX 75085-1872

Most Wanted List

Dal-ACE Experts

Donny Arnold...289-6746...Call before 10 p.m. 8-bit General knowledge.

Eb Foerster...357-7602...Call from 7 to 10 p.m. Turbo Basic, SynFile, SynCalc, Assembly.

Ron King...(817)283-0674...Call from 5 to 10 p.m. 8-bit hardware.

John Saunders...(817)566-0318 C and Assembler languages.

Michael Trombley...429-6134 ST general knowledge.

Ralph Tenny...235-4035 Call from 7 to 10 p.m. ST general knowledge and hardware.

Rene Tucker 223-6176 8-bit general knowledge.

John Winer...907-1349 Systems Programming and general knowledge.

Infomart Directions

From North Dallas, take either Stemmons (I-35E) or the Dallas North Tollway south. From Stemmons, take the Oak Lawn exit, turn East, and park at the Infomart. If you are using the tollway, exit right on Wycliff, go left on Harry Hines Blvd. to Oak Lawn, and turn right. From the South, take Stemmons north, then follow above directions. Infomart is the big, white, steel and glass building south of the other marts. **GUESTS ARE WELCOME!!!**

Newsletter

Advertisements

Personal ads are free to all current members. Please see the editor for details.

Commercial Rates

Full Page $35
Half Page $25
Quarter Page $15
Business card $10

For an additional $10 per full page, or $5 per partial page, you can request that your ad be placed on the inside front or back cover or the center page spread. This service is first come, first served.

Ads must be camera ready. Submission deadline for ad copy is the first of the month prior to publication date. That is, November 1st will be the deadline for your ad to appear in the December newsletter. Mail copy to the address on the back page, or contact the advertising manager, editor, or the Vice President of Communications. Copy received after the deadline will be run the following month. For contract advertisers, if no new ad is received prior to the deadline, the most recent ad will be run.

Editorial Policy

The editorial staff of the Dal-ACE newsletter reserves the right to edit your submissions for spelling, punctuation, grammar, clarity, and for reasons of space limitations.

Newsletter

Submissions

Submissions are welcome in any form. Submissions can be uploaded to the club BBS or they may be submitted to the editor at the club meetings or by mail.

Disclaimer

The material printed in this newsletter reflects the opinions of the authors. Opposing opinions are solicited. Unless otherwise stated, the material in this newsletter is not copyrighted and no rights are reserved.

The purpose of this newsletter is to present information for your consideration. Neither the editor nor Dal-ACE make any claims for the validity or usefulness of this material. The reader is the final judge of any product or advice presented.

Infomart Meeting Dates

Firm Dates: June 10, July 8, August 12.

Tentative Dates: September 23, October 14, November 11, December 16.

Meeting Schedule

10:00-11:00...8-bit SIG
11:00-11:30...Disk sales
11:30-12:00...Main meeting
12:00-12:30...New Users SIG
....Newsletter Exchange SIG
12:30-2:00....ST SIG

DAL-ACE OFFICERS

President:
Donny Arnold 289-6746

Vice President:
Brenda Arnold 289-6746

VP Communications:
Anita Uhl 492-8682

Secretary:
Michael Duke 739-3116

Treasurer:
Rene Tucker 223-6176

Members at Large:
Terry Borchardt 296-4699
Dave Gramm (214)370-7143
Randy Randolph 381-7624
Marc Salas 255-8425
Ralph Salmeron 254-8633

DAL-ACE BULLETIN BOARD

(214) 255-8256

DAL-ACE VOLUNTEERS

Editor:
Open

Ad Manager:
Marc Salas 255-8425

BBS Sysop:
Ralph Salmeron 254-8633

8-Bit Librarian:
Tim Mixon 356-4725

ST Librarian:
Gary Loges (817)481-4186

DAL-ACE INC.

Dallas Atari Computer Enthusiasts

Dal-ACE is an independent user education group that is not affiliated with the Atari Corporation. This is the official newsletter of Dal-ACE and is intended for the education of its membership as well as for the dissemination of information about Atari Computer Products.

Dal-ACE membership is $20 per year. BBS only membership is $10 per year. This newsletter is written, edited and published by volunteers. Its availability and/or distribution may, at times, be subject to circumstances beyond the control of the club officers. A pink address label indicates that your membership expires this month. Other user groups may obtain a copy of this newsletter on an exchange basis.

P.O. BOX 851872
RICHARDSON, TEXAS 75085-1872

U.S. POSTAGE PAID
BULK RATE
PERMIT NO. 1203
RICHARDSON, TEXAS 75080

Address correction requested. Return postage guaranteed.