T — Y — N — E — & — W — E — A — R

A T A R I 8 B I T

TYNE & WEAR

ATARI 8-BIT USER GROUP

Newsletter of TWAUG

Software
Editorial
Buy & Sell
Hardware

TWAUG
8-BIT

Reviews
Help line
Section
Repair Info

Public Domain Library

ISSUE #3

MAY/JUNE 1993

A T A R I 8 B I T

U — S — E — R — G — R — O — U — P

EXCEL DISK MAGAZINE

This disk magazine is no longer being produced.

It had been an excellent magazine, but the support wasn't there any more and therefore Bob Stuart couldn't carry on.

Bob Stuart gave the T.W.A.U.G. newsletter permission to include all his EXCEL DISK MAGAZINES, from number 1 to the present number, into their PD Library.

Anyone wishing to purchase any back numbers, contact T.W.A.U.G.

They are being sold at our PD double sided disk prices

FUTURA : XL/XE NEWSLETTER

Each issue is packed with

ARTICLES and SOFTWARE
Articles include 8-bit info, news, trivia, profiles, reviews, programming DIAMOND GOS and VCS columns, etc...
Software includes the latest UTILITIES, GAMES, and DEMOS.
Send now for the latest issue:
Disk.....................................£1.95
Printed copy & cassette.....£3.95
Cassette of programs.........£2.95
Please make cheques/POs payable to S.J.MURRAY
and send to -
Stuart J.Murray,
North of Scotland Atari User Group,
71 Walker Road,
TORRY, Aberdeen AB1 3DL

BOOT !

The Official Newsletter of LACE

THE LONDON ATARI COMPUTER ENTHUSIASTS.

As a member of LACE you will have access to: A monthly group meeting within the Greater London area. A quarterly magazine containing solely 8-bit orientated articles. A yearly Cover Disk/Cassette of exclusive PD. Discount on any PD purchased from our library. a dedicated menu on the City BBS.

All this can be yours for £7.00 per year (4 issues of BOOT).

Membership application form can be optained by sending your name address to :

**L.A.C.E.
143 Richmond Road
Leytonstone
London E11 4BT**

8 : 16

The Alternative Atari Newsletter

The Bournemouth and Poole Atari User Group

The BAPAUG club produces a quarterly newsletter for the 8 and 16 bit user. The subscription rates for the 4 issues is £6.50.

Anyone wishing to receive the 8:16 newsletter should send their address to:

**Mr.Colin Hunt
248 Wimborne Road
Oakdale, Poole
Dorset BH15 3EF**

TWAUG NEWSLETTER

EDITORIAL

Who to blame!!!

John Matthewson
David Ewens
Max Gerum

We are so grateful for the marvellous support we have received from as far apart as Israel to New York and we thank you very much for it.

I have a small request to anyone who wishes to send articles for publication, could you format the text in 80 columns instead of 40 columns and without any commands in the text. The reason is I use DD3 and most articles are printed in high density.

I also would like to appeal to any letter writer to print the letters, especially when they are for publication. Most of you do print them but we have had some hand written letters and they weren't very legible, it does take a lot of time when trying to read the letters.

This issue of TWAUG was produced with an Atari 65XE upgraded to 1 MEG, 2 Atari 1050 disk drives with US doubler and Happy enhancement in each and printed with a Citizen 120D. The software used was Textpro version 4.54 word processor and Daisy-Dot III print processor. The disk operating system I use is MYDOS 4.50, suitably enhanced for a 1 MEG Ramdisk.

The next issue will be ready by mid-July.

CONTENTS

EDITORIAL 3

LETTER SECTION
Your letters and a short reply. 5

DAISY DOT III USER'S GUIDE
Review by M. Gerum 7

DISK CONTENT
A short explanation of the programs on the disk. 11

BASIC TUTORIAL
A new series for beginners
by Ofer Safemon 12

ABOUT THE DISK FORMAT
by Nir Darey 13

FATAL CONNECTION
Instructions 14

LANGUAGES FOR ATARI
by Jim Cutler
and Word Search 15

TAGALON MAP
A map for Tagalon
by Mark Stinson 16

HINTS & TIPS
by D.Ewens 18

TOOLS FOR WRITERS
by Jim Cutler
and BASIC DEBUGGING HINTS
by Jeff Maddock 19

NEW SOUNDS & SIGHT
by Bruce Fairhall 20

REVIEWS
Reviews of three games
by Mark Fenwick 21

BUY, SALE & CONTACT
and WORD SEARCH ANSWERS 23

DRIVE NUMBER INDICATOR
How to make this gadget
by M.G.Rice 24

CRACKING THE CODE
Machine code for the beginner 25

Letter Section

Gentlemen:

Having just very avidly read your T.W.A.U.G. newsletter, issue number 2, caused me to sit down and write this little note.

I thought issue number one was extremely well done, and I compliment you for it and the great disk also, but I must happily say, this issue is even better. I am amazed at the variety, expertise, and contents of each of your articles. They reach out to all segments of the 8-BIT community, whether newcomer, or advanced.

As the Editor of the OL'HACKERS ATARI USERS Group, Inc. bi-monthly newsletter, I am well aware ot the sweat and tears that go into attempting to put out an interesting and informative missive every other month. To read and review all sorts of material, to pick and edit, to do the necessary typing, etc. and finally to put out what you hope is a worthwhile issue, is a tough job. Then too there is the funny feeling that perhaps this next issue is not going to be well received, all creating anxiety. Your dedicated group of 3, need not worry, 'cause if you continue in this same path, your future issues will continue to be sought after, and you will have succeeded. I want you to know how valuable an asset you are to us dedicated 8-BIT'ers. I and all the OL'HACKERS wish you much good luck, and I look forward to future issues.

Atarily yours,

Alex Pignato, 'President'
OL'HACKERS U.G.,Inc.

Many thanks for your marvellous and encouraging letter. Receiving such a congratulatory letter from an Editor who publishes a first class newsletter does lift our confidence. When selecting material and writing articles does put some doubt into the back of the mind, one is forever wondering what the readers think about it.
Your letter, Alex, has injected a new lease of life into our crumbling bodies, no the wonder we are running around with our tails up in the air, (pardon the expression).

This is a query from R. Mussell:
I have recently bought an 800XL and some disks most of which are Dos 3. When I transfer them to Dos 2.5, I can't get them to run. Can you help?

Reply by Max,

Firstly, format your disk with DOS 2.5 and write the DOS.SYS onto it with the 'H' Option. Make sure the DOS 2.5 disk is the original disk, this disk should contain the manual and three utilities. Insert this disk into the drive and load the utility COPY32.COM with the 'L' Option, and just follow the prompts. This utility will convert DOS 3 programs to DOS 2.5, and you will find that they will work fine after the conversion. I know many moons ago I used the same utility to convert all my DOS 3s to DOS 2.5.

Letter Section

Dear Editor,

Just a line to say thank you to Max Gerum for helping me to understand the ins and outs of the Textpro 4.56XE more clearly.

I like this word processor to use with DD3, I had a funny experience while using this, I tried to print to printer and got a flash on the screen and a sound like a short circuit plus 'ERROR 140'. I tried this 2-3 times, then looked it up in the manual, according to that I should have blown my computer up!

I must have been lucky, my three back up disks wouldn't work, so I tried my origional and this worked (I think I know what happened Max had put his CNF file on my disk and I had corrupted it because I had write-protected the disk.)

Once again thank you Max for your help, I hope I can be more constructive when I have further help.

Best regards to you all.

Bill Hall.

Reply by Max,

It is always very satisfying to hear that the help we've been giving was a success.

But it certainly is a mistery to hear that the CNF file got corrupted when write-protecting the disk and that the computer very nearly blew up. The CNF file in TextPro is the configuration file it includes the printer driver as well as the page format for printing and colour of the screen etc. and this should not get corrupted nor should it do any damage. No, I think something else must have happened, but what I couldn't tell you without knowing exactly what you've done from the moment you've loaded the WP.

We received a letter from Tony Bingham who says:

I saw the letter in issue 2 from Bill Hall, asking for info on the XF551 drive. Although I don't have one of these drives myself, I remembered reading an article in one of Dean Garraghty's news letters about it. I have enclosed a copy of the article and hope that it will be of some help.

Thank you Tony for your letter and the information, we have passed the article on to Bill Hall and we hope it will be of help to him.

LETTER SECTION & DIY

by Paul Kavanagh

I offer my congratulations on a fine mag and disk, I know it's a newsletter not a mag so forgive the error. I have read both issue's and although the info in some parts is a bit too technical for me or is not something I have any real interest in, I still read every article because it is written in an interesting way. I found most of the articles very interesting and whoever is doing the job of writing or translating letters etc. is doing a very fine job indeed.

I was wondering if there could be a DIY section for the likes of myself and others who are good at repairing things, such as disk drives or printer ribbons, or info on where to get things from so that we can all keep our Atari's alive. A bit of first aid wouldn't harm the Atari. I offer these ideas just in case anyone wishes to use them. I have tried and fully tested them with first class success. If anyone is not capable of doing these jobs then they should pass them on to someone who is.

Disclaimer: Any idea read from or used via this newsletter or author is your own responsibility. Any damage caused by the use of any DIY idea published here is your own fault. I know nothing.

RIBBONS: If you have a snapped or torn ribbon, assuming it is a nylon ribbon and you cannot just buy another, then what you can do is cut out the damaged piece and use your wifes iron to put the ribbon back together again. Don't use superglue or any other glue, it won't like it.

Set the iron on hot, it can be a bit difficult to get the temperature just right. Test the very edge of the iron on the ribbon to see if it melts, the iron should be applied at about 45 degrees angle to the material. Look at the original joint to see if you can copy it, I did and so I know it can be done, it isn't easy at first but once you get the hang of it you will find it very easy. If the ribbon melts and sticks to the iron it is too hot. Experiment a little, when the iron is at the correct temperature the ribbon will crinkle slightly when the iron is applied. Place the ribbon on a flat wooden surface (not varnished or formica, it is too slippery).

RE-INKING ribbons: Get a small jar, preferably glass. The jar I use is one of those tiny jars that you get jam in, depending on the size of the ribbon you are using will depend on the size of the jar required. I use an Atari 1029 ribbon so I don't need much.

Remove the ribbon carefully from the cartridge and put it in the jar, now put about 5 or 6 drops of ink onto the ribbon, I use endorsing ink. Put the lid onto the jar and give it a good shake, after about thirty seconds it should be done. Remove the ribbon from the jar, dab off the excess ink with a tissue and hang up to dry. A bottle of endorsing ink will cost around 80 pence and it should last for about 40 inkings. It is available at W.H.Smiths or any typewriter shops or stationers.

1027 SPONGES:(Does-not-include-Plastic-Case).

If you need a new sponge roller for your 1027 then go to a typewriter shop and ask to see a selection of the rollers available. I got one that was over twice the size in length I needed, it cost me £2.40 and I was told by the assitant in the shop, that my own little 1027 roller would actually strip down.

Use a pair of rubber gloves to do this job, what you do is remove the roller from the outer casing by prising it out with a small screwdriver. Be careful not to snap the legs off it when you grip it, grip each end of the actual roller where the sponge just starts to come over the retaining lip, this can often be felt easier than seen, gripping firmly both ends pull and eventually it will come apart, they are not glued but may be tight. Try not to use pliers or any other metal instruments to seperate them because this may cause more damage.

Now remove the sponge from the two plastic pieces you've just pulled apart and do the same with the new roller. Lay the new and old sponges side by side and with a pair of scissors cut the new sponge to the same length as the old one. Try to cut the sponge accurately, if the ends overlap this may cause jamming when replaced in the main casing, or if cut too short may cause spinning as their wont be enough grip by the outer rim.

MARK'S GAMES COLUMN....

Hello, and welcome to the first of a new series looking at games, which our trusty old 8-bits have been master of for many years, and in my opinion still are.

I will try to ensure that this column is reasonably well balanced but many of you will already know my bias towards adventure. So, much of this column is up to you, if you would like to send hints, tips, news, or anything else game-related then please post it to the address below. If you don't I will be happy to write column after column on adventure!

My address is:

Mark Stinson,
7 Arleston Lane,
Arleston,
Wellington,
Telford.
TF1 2LT.

And now to our first submission by Alex in Moscow, a full map of Tagalon. Tagalon is an excellent game by Tynesoft in which you have to search a forest and village looking for silver balls which are worth points, and various other collectables to aid you on the quest, including food, keys, shields, bombs and elixirs. If you don't own this game then get one quick! Many thanks Alex, keep sending those maps, they are excellent. Well, that map should keep you going for a while, so until next time when Alex provides us with a map of Hans Kloss, good adventuring.

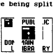
DAISY-DOT III
USER'S GUIDE

Author David Richardson
Magnifying DD3 fonts (converted icons)

by Max Gerum.

In this issue I will explain how to magnify the converted icons. If you remember in the last issue I mentioned that we cannot magnify the icons because the conversion program changed them into a magnified font of quadruple height . What we must do first is to convert these fonts back into a single font before we can proceed with the magnification.

Before explaining how to convert the icon back to a single format, let me show you a printout of a converted icon with a character spacing of 19, just to let you see how the icon looks after the conversion. If you can recall, I mentioned in the last issue, that the icons are being split vertically into three parts.



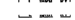
Having an icon split in three parts is the reason for using three characters for each icon from the table of characters printed in issue #2 of TWAUG. If there is someone who still needs help, drop a line to the newsletter with SAE for a quick reply.

There is a macro file available which we use for the conversion, it is a public domain program and has been specially written for TextPro by Mr.McGowan, it is called FNTSPLIT.MAC. It's on the Daisy-Dot III User's Guide disks, and is available from our PD library.

I am using TextPro version 4.54, but any other version of TextPro will do the same job, they all have a macro facility. Boot your TextPro, disabling BASIC at the same time. When loaded, press CTRL+V, this will give you the prompt "Load macro"D: in the command line, type FNTSPLIT.MAC and press Return.

If you have only one disk drive and no RAMdisk, make sure the disk with the icons you want to convert to a single format is in the drive. Only one drive can be used, the re-converted icons are written back to the same disk, but you could use the RAMdisk instead. With the macro file loaded, press Start and then any key to display the icon load prompt in the command line. Type the filename but do not include the extender, the macro will append its own ext. It will only take a few seconds to converted the font back to a single format. If we now check the directory of the disk or RAMdisk we will see that there is a set of four icons with the extender of .NLQ. I am giving the icon the filename "IC", we will now see IC1.NLQ, IC2.NLQ, IC3.NLQ and IC4.NLQ.

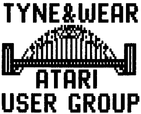
When printing these icons all four sets of NLQ's must be used to print one icon. The character and line spacing must both be set to zero. To set the character spacing we would enter \SO and for the line spacing \XVOO. Let me show you first what the icon now looks after it has been re-converted, I am setting both the spacings to 19 so the spaces are noticeable.



In the last issue of TWAUG I included an example print with text beside it. As you can see only one line was printed and then a big gap before the next line appeared. With the conversion we've just done, four lines of text can be added after the icon. Mind you I am saying after, because adding text after it is much easier than before. With having to type in four lines to print one icon, instead of just the one line, does set a problem. We must make sure that the icon is in one piece when printed, the only way to do that is by adding a set margin. If we want to print near the left of the page we would add \L, for the center add \C and the right margin \R. If we add one of these margin commands to the setup the icon will be printed in one piece. I will now show an example setup to print one icon without text, and it would be centered.

```
\XV00\S0\C
\FD8:ic1\!"#
\FD8:ic2\!"#
\FD8:ic3\!"#
\FD8:ic4\!"#
```

When we magnify a font or icon we have a possibility of four magnification options. Double height (2), treble height (3) or quadruple height (4). Again I must show you another example this time of our logo "the Tyne Bridge" in quadruple height and width.



On the main Daisy-Dot III disk is a utility program called FU.COM, which we are using to magnify our icons.

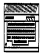
From TextPro, either the editor or the directory, it is easy to exit to DOS, simply press CTRL+X and type 'Y' to confirm the move. If you have the FU.COM program in the drive use the L Option from DOS and load the utility. When loaded you will see the menu with four options, choose the first option, M for Magnify. When loaded the first prompt for source drive, enter the filename, when using only drive one the device need not be specified. The next prompt is for destination drive to save the magnified font to. Assuming we have given the filename IC, we would have IC1.NLQ to IC4.NLQ on the disk. If for the source prompt you answered with the filename "IC1" I would recommend using "XIC1" for destination. Magnifying "IC2" I would name the destination filename "XIC2", doing the same with the other two parts of the icon. After you've entered the destination filename and pressed return the magnification size is requested, you have the option of (2), (3) or (4). When you have chosen one of these numbers press any key and the magnification will begin. Note: All four icons, IC1 to IC4 must be magnified to the same magnification.

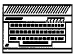
When the magnification is completed return to the word processor. Just press return and you will be returned to the main menu, press the highlighted X for EXIT and the DOS will load, or when the RAMdisk is in use the DUP menu re-appears on the screen. Load the word processor again and set up your icons for printing. I will now show you how to do it, but I am only setting up a set of three magnified icons for printing, I am sure you will be able to add the rest of the icons, depending how many you've got on the disk.

```
\XVOO\SO\W2
\L\Fxic1\!"#\C\Fxic1\$%&\R\Fxic1\'()
\L\Fxic2\!"#\C\Fxic2\$%&\R\Fxic2\'()
\L\Fxic3\!"#\C\Fxic3\$%&\R\Fxic3\'()
\L\Fxic4\!"#\C\Fxic4\$%&\R\Fxic4\'()
\XVO4\S2\W1\Ftimes\\L
```

If you have a good look at the last line above you will see the line spacing has been put back to normal, from \XVOO to \XVO4 and so has the character spacing, from \SO to \S2. Next is the width, the default setting is \W1, if we leave the width set at \W2 the text would be printed in double width. The \Ftimes is the font I am using in this text, you need that or you get a funny result. The \L is the left margin, as you can see the last icon would have been printed at the right margin of the page, we want to start printing text from the left margin.

When the font or icon has been magnified being double, treble or quadruple height, only the height is affected. To make the icon look normal instead of being tall and thin, you would use the width command. A magnified font of double height would look normal when you add a width of 2, thus \W2, only width from 1 to 4 can be used. Let me show an example of what I mean by saying looking tall and thin.





I conclude my article with a printout of a handbill I designed and printed for a painter, it's on the next page.

Enjoy DD3!

PAINTING and **DECORATING**

ARTEXING

also

OUTSIDE **PAINTING**

NO JOB TOO SMALL
COMPETITIVE RATES

FOR DETAILS PHONE

PAINTER'S NAME

NUMBER

Designed & Printed by AMG™©1992

DISK CONTENT

Here is a short explanation of issue #3 disk content. Side A contains a varied selection of programs, and Side B contains a program for Light Gun owners and a speech demo.

ALPHA SHIELD: When you first load this game you are presented with the option to read the instructions. It is an easy enough shoot them up game, you must try and blast the spaceship which is in the centre of the screen with a shield around it. The shield has two small parts open into which you can move your spacecraft. But beware the shield opening is constantly changing and the stationary spaceship is shooting back.

FATAL CONNECTION: The instructions of this game you will find on page 14 of the newsletter and is also included on the disk. I am not familiar with games in general and therefore I cannot compare this game with another which is similar. But I am sure you gamesters will soon get the hang of it.

THIRTY ONE: This is a dice game played against the computer. Some explanation is shown on the screen, it also tells you that it is possible to beat the computer. To play it you just follow the screen prompts, it will ask you if you want to go first or second and tell you when it is your turn to go. All you do then is enter a number character from one to six, one number for each go, which must not go over 31.

DEMO: Then we have a short demo with a scrolling line especially done for T.W.A.U.G. by John E.

OUNCES TO GRAMS PRINTER: In issue #2 we had a times table on the disk, well this time we have included a conversion from ounces to grams.

1020 PLOTTER PROGRAM: This program prints inlay cards for cassettes with the 1020 Plotter. I do not possess a Plotter and therefore I cannot tell you much about it.

GRAHAMWRITER: This is an 80 column word processor, it is in Basic and therefore you can list it to screen. There are no instructions, but it seems to be very to use, the most used commands are on screen. You use SELECT or OPTION to select and START to enable and also RETURN for some performances.

LIGHT GUN BLASTER: Anyone with a Light Gun will find this usefull for the gadget.

PARROT II DEMO: this is a speech demo, to get the sound just right you must enter the speech speed, and the correct speed is between 40 and 48, try 45.

Enjoy...

THE BASIC TUTORIAL

by Ofer Safemon

Hello to all you Atari fans out there that are still supporting this marvelous computer.

This is going to be our new Basic Tutorial written by Ofer Safeman from Israel.

Many Atari users have some knowledge of Basic, but since I'm going to write a tutorial, it is intended for everybody so the more experienced programers will have to endure the basic stuff, at least at the beginning.

Since many Atari users are already familiar with the superb Turbo Basic, which I think is the best Basic for the Atari ever written, you will be glad to hear that although I'm going to write the demo programs and explanations using standard Atari Basic, I'll devote in every article a segment for Turbo Basic according to the covered commands in that article.

Now lets stop the nonsense and get to work. As I said, first the basics, and what is more basic than the PRINT command. The PRINT command does exactly what the word says - it prints to the screen.

For example:

PRINT "SAMPLE" - will print the word SAMPLE after presing RETURN.

Note the use of quotation marks which is very important as if we only write SAMPLE, the computer will try to print the value of the variable SAMPLE, but about variables later on.

Now lets try some maths:

PRINT 10+5 - after pressing RETURN the computer will give the result which is 15.

Well that, I must say, was definitely basic stuff and we want to advance a little.

First every basic program must consist of lines. The examples above are in direct mode, which is a way of entering only a few commands, but to write a Basic programm we need line numbers. So if we want, just for example to type a lot of things we will need a program:

```
10 PRINT "THIS IS A SAMPLE"
20 PRINT "PROGRAM"
```

Now we are a little bit smarter lets start making things shorter. The abbreviation of the "PRINT" command is "?" - Yes just a question mark.

We have reached the point in our tutorial that we can discuss VARIABLES. There are 2 types of variables NUMMERIC variables and STRING variables.

For example:

```
VAR=10
OFER$="THIS IS A MESSAGE"
```

Variables have some restrictions. A string variable has to be DIMensioned first to let the computer now how much space you need for it. The command that performs this task is DIM or COM.

For example:

```
DIM A$(10)
```

This will tell the computer that you will use a string variable called A$ and it will be 10 characters long, meaning in those 10 spaces you can use any ascii character on the keyboard.

You can use the same name for a nummeric variable and a string variable but that could get you mixed up so it is best to avoid it.

The next thing I would like to discuss are arrays or two-dimensional strings. An array must be dimensioned just like a string but it can apply only for numeric arrays because the Atari doesn't have string arrays.

For example:

```
DIM ARRAY(100,100)
```

This will tell the computer to reserve 10000(100x100) spaces for the newly dimensioned array. It is very useful for tables because ARRAY(0,0-100) will hold the first column of a table ARRAY(1,0-100) the second, etc.

Maybe the readers till this point will ask why don't I give more examples? Well, a few reasons: First of all, most of this is quite easy to understand and could even be fun to experiment with, second I thought it best if at first I stick to the basics and after a partial knowledge of the fundamentals I'll give some tricks and longer programs to study and experiment with. There are also some overlapping issues, for example when I'll talk about graphics it will include some of the basic stuff that we will go over at first and next time I'll start writing according to ordinate topics which will make understanding easier.

So this was the intro. I hope you enjoyed it.

See you next time.

ABOUT THE DISK FORMAT

By Nir Darey

A diskette is composed of a thin magnetic disk covered by thin plastic usually black. The cover has an open area on both sides exposing the magnetic disk surface to the drive for reading or writing. As the diskette spins in the drive, the read/write head is actually over the opening, reading/writing the disk surface like a cassette recorder would.

The diskette is divided into tracks. A track is a ring about the center of the diskette. The drive head can be positioned over any one of the tracks, and data can be read from the surface.

A disk can be formated in different formats. Single Density format is dividing the disk into 40 tracks of 18 sectors each with 128 bytes per sector, a total of 720 sectors, that gives us 90 kilobytes on the disk. Dual Density or 1050 Density, the disk is divided to 40 tracks of 26 sectors with 128 bytes each, that give us 130 kb on a disk. Double Density format is dividing the disk to 40 tracks of 18 sectors each with 256 bytes per sector for a total of 720 sectors, which gives us 180 kb.

Now we will talk about how data is transferred from the diskette into the computer. A sector of data is a magnetic field that is converted into electric pulses which are fed to the floppy disk controller. The floppy disk controller is the interface between the read and write head and the drive microprocessor. The floppy disk controller performs all sector searches transfers data between the microprocessor and the disk. The disk drives processor receives a full sector of data every 1/10 of a disk spin. This is about 0.0555 seconds.

About the Disk Operating System format.
DOS 2.0 formats the disk at Single Density (a total of 720 sectors). DOS 2.5 can format the disk in Dual Density (1040 sectors). Both formats use sectors which contain 128 bytes of data. In DOS 2.0 there are 707 sectors free for files, and in DOS 2.5 there are 1011 free sectors. You will probably ask what happened to the missing sectors?, well the DOS uses most of those sectors to store information about the disk and the files that are on it.

The disk sector map for DOS 2.0:

sectors :	contents :
1-3	boot information
4-359	free for files
360	VTOC
361-368	directory information
369-719	free for files
720	not used

The disk sector map for DOS 2.5 is the same as DOS 2.0 with the additions of the following:

sectors :	contents :
721-1023	free for files
1024	extended VTOC
1025-1040	not used

Now let's examine how the DOS uses sectors to store and keep track of programs or data files.

Lets look at a DOS disk. The first three sectors contain the boot information. Sectors 4 - 359 are free for files. Sector 360 is the VTOC. Sectors 361 - 368 hold the directory information, Sectors 369 - 719 are free for files. Sector 720 is not used. Sectors 712 - 1023 are free for files in enhanced density only. Sector 1024 is the extended VTOC for enhanced density only. Sectors 1025 - 1040 are not used.

There are three different kinds of sectors that store information about the whole disk.
1, BOOT SECTORS.
First we must understand what a boot sector is. well, a boot sector is the first sector on a disk. the boot sector is actually the header of the whole disk. The most important parts of the boot sector are it's first six bytes.
Byte zero is the boot flag, it is usually unused. Byte 1 contains the number of sectors to be read as part of the boot process. the number can be up to 256. Bytes 2-3 contain the start address to load the boot data.
Bytes 4-5 are the initialization address. As for the boot sectors in the DOS disk, the boot flag is always zero, the number of boot sectors are three, the address to load the boot sectors is 1792 and the initialization address is 5440. Bytes 6-7 are the address that the computer will jump to continue load. Byte eight is unused. Byte nine is the number of sector buffers (=3). Byte ten is the drive enable bits, bits 0-7 are equal to drives 1-8. byte ten is usualy 131 in decimal, in binary it's equal to 10000011. As you can see drives 1,2 and 8 are in use. Byte eleven is unused. Bytes 12 and 13 are the start address for the buffers, equal to 6604. byte 14 indicates if there is DOS on the disk or not. If the byte is equal to zero means that there is no Dos on the disk, if the byte is 1 then DOS.SYS is on the disk. Bytes 15 - 16 are the first sectors of DOS.SYS file usualy it's 4. Byte 17 is the offset to sector link data, the byte is equal to 125. Bytes 18 - 19 indicates the start of main DOS.SYS file, it's equal to 1995 The 20 byte is the first byte of the boot data. Those values can be changed with an editor.

2, Volume Table Of Contents (VTOC).
The VTOC data is located in sector 360, and in this sector you can find out how many sectors are free, what sector is used or unused and more. The first 10 bytes contain general information about the disk. Byte zero is used to indicate the DOS type normally it's equal to 2 for DOS 2.0/2.5. Bytes one and two contains the total or sectors on the disk. for DOS 2.0 it's 707 and for DOS 2.5 it's 1010. Bytes three and four contain the number or currently free sectors.

Bytes 10 to 99 contains the values representing the first 719 sector of the disk. Let me explain how it's done: divide bytes 10 to 99 into bits. Byte 10, bit 7 is sector 0. Byte 10, bit 6 is sector 1, Byte 11, bit 7 is sector 8. and so on... If the value of the bit is one then the sector is free. If the bit is equal to zero then the sector is in use. The rest of the bytes, (bytes 100 to 127) are unused. DOS 2.5 has an extended VTOC that is located in sector 1024. Bytes 0 to 121 represent sectors (48-1023) the same way as before. Byte 0, bit 7 is sector 48. Bytes 122 - 123 contain the number of free sectors on enhanced area only.

Now Directory Sectors.
There are eight sectors that are reserved for a diskette directory. Each sector is able to hold up to eight files. This means the maximum number of files that can be placed on a single diskette is 64. There are 16 bytes available for each file entered. The first byte is the flag byte, the flag byte contains the information about the condition of the file, divide the flag byte into bits and check: If bit 0 = 1 then the file is open for output. If bit 1 = 1 the file was created by DOS 2.0/2.5. Bits 2,3,4 are not used. If bit 5 = 1 then the file is locked. If bit 6 = 1 then the file is normal. And if bit 7 = 1 then the file has been deleted. The second and third bytes contain the total sectors of the file. Byte 3-4 contain the starting sector number of the file. Bytes 5-12 contain the file name, and bytes 13-15 contains the file name extension.

3, File Sectors.
In one sector there are 128 bytes. Each file sector contains 125 bytes of file data followed by three bytes of DOS information. Bytes 0 to 124 contain the file itself, which can be data, text, basic program or anything else. Byte 125 contains the total number of bytes that are actually used, which is normally 125. The next two bytes (126-127) contain two pieces of information. The first six bits of byte 126 hold the file number, the number can be from 0 to 63. for example, the first file in the directory would have the value of zero here and the second file would have the value of one. The final two bits of byte 126 and the whole of byte 127 when put together give a 10 bit number which point the next sector of the file, this value will be set to zero of this sector is the end of the file.

That's all, if you have any questions please contact me through TWAUG and I will be happy to reply.

Electric Software Presents

Fatal Connection ver.1.0

By Brandon Clark

Public Domain Software

Thousands of light years from Earth, on a small planet commonly known as Thontaris, (a name it was given by the natives which means "Thunder") a certain species of ancient life is becoming extinct, due to the spreading of a black material known as "The Void". No life can exist in The Void, and the plinkers (small, intelligent life forms) seem to be dying. A special Void-resistant substance has been found to help the plinkers survive, and it is your job to spread this substance around each of two scenarios, and lead the plinkers to their safety boxes, represented by the numbers "1" and "2".

The plinkers will need to be guided to their destinations, a tricky job at least. You will be given control of a special rotating capsule, which can be destroyed by accidentally touching one of the many toxic plants that grow in the most inconvenient places. You can also collect special minerals for points.

Use joystick 1 to control your capsule. Each scenario can be completed by returning the plinkers to their boxes, and completely covering The Void. When one of these tasks is complete, the Void-resistant material will flash different colors. To pause the game, simply press the SPACE BAR. Press it again to resume play.

Good Luck!

Brandon Clark
14544 S. Remington Rd.
Lakeland, FL 33803)

If you enjoy this game, you might want to consider sending a small donation (1 or 2 dollars is fine).

Thanks!

LANGUAGES ON THE ATARI 8-BIT

The attraction for users of many of the high powered expensive, top of the range computers was the variation in programming languages which were available.

Present day users of the Atari 800XL or 130XE machines may be unaware of the range of programming languages available on their favourite machine. I list below those which I know of. There are certainly more, so if you know of these why not write to TWAUG and let us in on this.

1. ATARI BASIC
2. MACHINE CODE or ASSEMBLY
3. BASIC XL and BASIC XE specials from OSS
4. TURBO BASIC XL a very fast compiled basic.
5. FORTH , FIG-FORTH - both are PD
6. PASCAL - Kyan Pascal or Draper Pascal
7. C language -ACE C, Lightspeed C, ACE C
8. LOGO - if you've never tried this, do so, instant reaction on screen to your commands
9. INTERLISP/65 a good implementation of LIST Processing, the base of Artificial Inteligence - issued as part of the old ANTIC catalogue.
10. PILOT a programming language of the early Atari era
11. ACTION a cartridge based language/
12. QUICK available from Dean Garraghty.

Of these I enjoy LOGO which was, as usual, under exploited by ATARI CORP. Its a full LCSI implementation and the "Turtle" on screen responds to command such as Right 40, Left 40 etc but also quite intricate programming is possible. Originally written to encourage children (From 8 to 80) its simplicity means that the program executes before your very eyes.

Many of these languages were re-issued for the ST at vastly enhanced prices and packings, and hailed as massive breakthroughs.
So be adventurous, try something other than stodgy slow old Basic.

JIM CUTLER   March 10 1993.

```
JTRJMINEROHNYGFDFGHV
EEUSGOLDEZGCKISQWNTY
TLNLOENXAMAPRIVCULMO
BBNWARTTZGZXSEESXNAK
OAEMUBAVEUNTXANEEELN
OTRXLRELFZUCKOTIPILA
TTHEIXLXELUOSTNBNORM
JLODEAYSCORMYHAAICMC
AOARNNOLDSWAARHZHENA
CEUIGGERFORYTOSOELLP
KLFSISAMCBOBYTNUOBIN
TYRETLOVESJNJMCALOVE
RCHONABOOTOFCUKHETRO
YALSLAGSOKBONGAEDINK
OUTJSNINURETOUCCCLPD
LMYTROFYTNEWTOUCHJMC
```

8-Bit Classsics
Word search

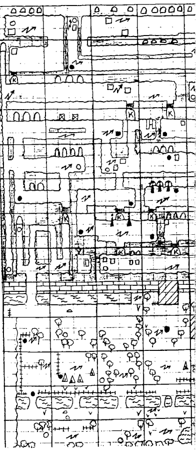
By: John McIntyre

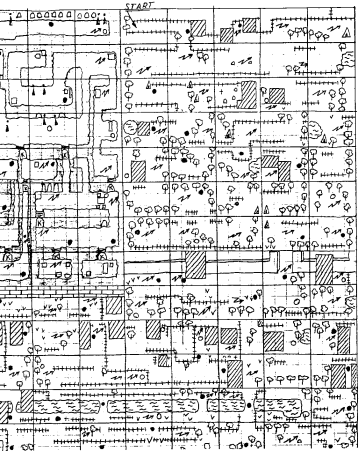
Words To Find:

Atari
Montezuma
Pacman
Bounty Bob
Archon
Final Legacy
Yukon Yohan
U.S.Gold
XLXE
Jet Boot Jack
Joust
Miner
Twenty Forty
Miner
Lode
Runner
Zaxxon
Touch
Tablet

TAGALON

GAME MAP




START

HINT & TIPS.

by D. Ewens.

The following hints and tips have been sent into us by some of our readers. I would like to thank them and all the other people who have submitted material for future issues of our newsletter. If you have any hints or tips that would help other Atarians, then please send them in and hopefully we can make this a regular feature.

Tip 1: When you boot your disk drive and the message 'BOOT ERROR' appears on the screen you can stop the drive by pressing 'CTRL+1' keys together. After the normal cut-out time, when your disk drive stops spinning, you can insert the correct disk, press 'CTRL 1' again and the disk will boot without having to turn your computer off and then on again.

Tip 2: Do you have the game MINER 2049er and still haven't got past the 3rd or 4th level, then read on. There is a way to get to any level of this game. Have fun with this one.:

1) Press START to begin the game.
2) When the graphics for Station 1 are on the screen, move Bounty Bob onto the first framework where he won't get killed.
3) Type in the following number. ---> 2137826861 <--- (it's the 'phone number of BIG FIVE Software in the USA) and also it is on the front of the cartridge.
4) Press the SHIFT key and the number of the level you wish to go to. You can do this at any point during the game. You don't have to type the number again unless you press SYSTEM RESET or turn off your computer.

Tip 3: If you own the Atari 8 bit and the ST computers and you also have Missile Command for the 8 bit Atari then you can use the ST mouse to play Missile Command. This is how:

1) Start the game.
2) When the main screen appears press CTRL-T and plug in the ST mouse into the first joystick port and press START and then move the mouse.

Tip 4: BOING II. Press CONTROL-C to enable cheat, then CONTROL-N to advance levels.

Tip 5: In TAC-TIC, the codes are: MENTOS, MARZENA and LEVEL UP to start on level 10, 20 and 30 respectively.

On our issue two disk, we had a bonus game called GUARDIAN OF THE SACRED SWORD. We have been given the solution to this adventure by the author. If anyone would like a copy of it, please send a S.A.E to us and we will be pleased to send it to you.

TYNE&WEAR
ATARI
USER GROUP

T.W.A.U.G.
P.O.BOX No.8
WALLSEND
TYNE & WEAR
NE28 6DQ

TOOLS FOR WRITERS.

If you use the ATARI 8bit machines to produce your own personal masterpieces be it Poems, Epics, Screen plays, sitcoms, letters to the Editor, or even letters to TWAUG telling them they are doing a fantastic job! Then here are details of a few programs to help you get a more professional document.

1. REFORMAT 2.0. - A super program which enables you to turn a 40 column text into an 80 column, or anything in between.
2. Those lucky enough to know a member of OL HACKER'S ATARI USER GROUP Inc of New York will find that their disk Newsletter contains a program to print out in two or three columns and with four different styles giving a quite professional "Newspaper" look.
3. If you want to produce an illustrated news sheet then three programs exist to do this.
3,1 NEWS STATION uses Printshop icons and builds up a banner head and six blocks of text/illustrations.
3.2 PUBLISHING PRO. - By the same people gives a wider range of uses. And NEWSROOM is the ultimate it comes as a program disk with four double sided disks of clipart. You can setup layout, go to the photolab, edit copy, and produce the final result in the press section. A very good, though fairly slow, system.
The only limits to all these are the fact that they only produce one offs-but thats what photocopiers were invented for!

A truly great program is PRINT TOOL which allows you to take a text file and by inserting a range of commands produce a very professional document indeed and you can do as many runoffs as you want, a little like DAISY DOT 3 but more text oriented. It's an American program and is easy to use. A guy I know in the States uses this system to produce instruction manuals of a very good quality indeed.

JIM CUTLER
10th March 1993

ATARI BASIC DEBUGGING HINT:
by Jeff Maddock

Confucius says: "Programmer who claims he can write computer programs with no bugs is fooling himself." It's almost impossible to write the first version of a program without at least one bug. Knowing that bugs exist is one thing, finding them is something else. Here is a simple routine which you may include in all new programs to help make the task easier.

```
10 TRAP 1000
1000 GRAPHICS 0
1010   ?     "ERROR      ";PEEK(195);"
DETECTED
AT FOLLOWING LINE:"
1020 LIST PEEK(186)+256*PEEK(187)
1030 ? "RUN":END
```

You can use whatever line numbers suit you, but make sure the TRAP is positioned early in the program or where it will catch all possible errors. In the example, when an error occurs control will be passed to line 1000. This prints an error message containing the error number, lists the line where it was detected, prints "RUN" and turns off the sound. The listed line may then be examined. If it contains an error, it may be corrected with the aid of the cursor control keys.

Press RETURN to enter the corrected line. This will position the cursor over the word "RUN". Press RETURN again and the program will re-start.

If the error was not in the listed line. Then it may at least supply a hint as to where the error occurred and you can try to find it in the usual manner. When the program is fully debugged, you can delete the debugging lines and everything should be hunky dory.

NEW SOUNDS AND SIGHTS

by Bruce Fairhall

Back when I first bought the 800XL I was forced to use the old family black and white television set, as cast out when COLOUR arrived.

To assist with the final stage in program writing/modifying, I set up an extension lead from the computer around the room to our colour set, and 2 switches, plus a channel change, meant I could then set the colours. All else was monochrome.

As most of my computer time is either word processing or within BASIC programs, this situation was generally bearable. That is, until recent times when picture quality, as a result of age (not mine!!), went into general decline.

Time for change and upgrading.

It seemed unnecessary to go colour so I looked around for a monitor, which would give me similar usage. However, the lack of SOUND was a minor (??) nuisance - that is until I discovered a monitor WITH sound.

It is the THOMSON VM 3102V.

AND, it is available in Amber or Green - to suit your preference.

Mine was purchased, with freight, for around $170, and apart from a slight "lean" on the picture it is a really great improvement.

The front panel, with a "cover up" little door, has a volume control for the rear speaker, brightness and contrast controls, plus screw driver adjustable picture controls.

CONNECTING MONITOR to ATARI

BUY: 2 - RCA phono plugs, best in different colours.
1 - 180 degree 5 pin DIN plug
About 2 metres of spiral wrap coaxial cable (light)

THEN: Halve the cable (2 X 1 metre).
Bare ends of the cable.
Solder 1 RCA plug onto one end of each cable.
Label cables NOW, one SOUND and one as PICTURE.
Dismantle the DIN plug. Put protector onto your cables.

NOW: Solder SOUND active lead to connector pin 3.
Solder PICTURE active lead to connector pin 4.
Twist the two ground leads together and solder onto connector pin 2.
Check soldering, then re-assemble the plug.
Plug the RCA plugs into the appropriate sockets at back of the monitor (SOUND has a music note symbol near it) and the DIN plug into your ATARI. Power on and see!!

DETAIL: Viewed from the soldering side of the DIN plug, these are the connections:

```
                  2 <GROUND

PICTURE>  4               5

          1               3 <SOUND

                ┌──┐
```

The groove is downwards, and the connectors are usually numbered.
Enjoy the sight and sound!

CRIMEBUSTER
Review by Mark Fenwick.

Crime Buster from Atari Corp, the last Rom produced for the Light Gun and relatively new over here. This has to be the best and probably most tricky of the Light Gun games.

The city has become over run with Mafia style gangs and hoodlums. They're everywhere, mixing with the public downtown, at the pier, and the Commissioner has had enough! Time for action. You as one of the Force's top detectives have been given a mission of cleaning up the city! You're not alone in your knowledge of this mission. Not only does the Commissioner and yourself know, but so does the Mob! So it's not just a case of you after them, they're after you, and they aim to get you before you get them! So impress the bigwigs at the department and blast those mobsters, but look out for the innocent bystanders. So it's down to you...

Crime Buster is a one or two player game. For two player you each take it in turn to do battle with the Mob. Don't worry though you'll only need the one Light Gun if you opt for a two player shootout.

The game starts with a title screen showing a Crime Buster book, revolver and car keys accompanied with an unusual piece of music. Just from the title screen you'll see by the hi-res graphics that this games been a good buy. Armed with your trusty Light Gun take aim and fire anywhere at the screen. Wherever you shoot a realistic bullet hole is left where you shot. From here we're at the one/two player option, surrounded by rewards for the hoodlums your about to do battle with. Simply choose by firing at either one or two player, again wherever you shoot a bullet hole impression will be left.

Once the number of players is selected you are confronted by a map of the city. The city is split into 12 sections. A white square border shows the area you are going to travel to. If you wish to go elsewhere simply fire at any part of the screen to move the square. To select that particular area just aim and fire anywhere in the square. Now you're on the road. The next screen consists of the top 3/4 of the screen showing the road with your patrol car travelling along it. The bottom 1/4 is your info panel. You start off with 3 lives and a full round of bullets which travel the length of the screen. Shown below your bullets is your speed and Rev. counter, at the side of which are two direction arrows left and right. By shooting these arrows your car will inch forward/backward accordingly. There are another series of arrows at the left these showing the direction of your bullets coming from your car towards the gangsters cars. By shooting, these bullets will come from your car at the angle chosen. The gangsters cars can easily be spotted by the stripes on their roofs. Beware of civilian cars as shooting these will lose you valuable bullets. Each time you shoot a Mob's car it will burst into flames and overturn. However, the same can happen to yourself as they are shooting at you, too! so be quick! If this happens you'll lose a life and see a skull and crossbones, shoot this to return to action or for player 2 to have his go.

If you manage to keep safely on the road you'll reach your chosen destination. There are four area's of play, Warehouse, Pier, Alley, Downtown. For Downtown you're confronted with a parked car on a street with two buildings in the background.

Now the action starts, keep an eye on the windows and doorways, don't forget the manholes in the street. The Mob will pop up from anywhere firing at you, so be ready to shoot. Shoot them and they fall back into the room. Shoot a villain at the doorway and he'll bend holding his stomach then fall to the ground. The graphics are well detailed with facial features and expressions, there's plenty of characters, but watch out for civilians. A young lad jumping up with a lolly, can easily be took for a gun! or a woman taking a look out of the window, so be careful a wrong shot can lose you valuable bullets. If you do hit a civilian you'll hear an ambulance siren. Another clever touch on some scenes are the lights, shoot these and the place goes into darkness. Each time you take a hit you lose several bullets, lose all your bullets and you lose a life.

On completing a zone on the map it will go shaded so you know which areas have been cleaned up. When you succeed in clearing a zone pick the one next to the one you've completed, that way you don't have to travel on the main road again. On losing all your lives you are given a final rating on your score ranging from Mobster to Crimebuster depending on how many Dragnets (zones) you clean up.

It would have been nice to have more bullets as they soon go, but then again it's very good fun to play.

Overall, a great game with plenty of play. Special effort has been made to small details to add to the playability. Well worth a purchase, if you have a Light Gun that is. Crime Buster is currently available from Micro Discount at £21.50.

CROSSBOW
Review by Mark Fenwick

Crossbow, a Rom for your Light Gun from Atari corp, made in 1988 but only recently heard of in the U.K. is a great adventure with a difference!

The story revolves around an evil wizard who has been terrorising the kingdom for centuries. Apparently he has robbed the kingdom of its riches for too long. Now it's time he got his come-uppance! Your mission is to lead four adventurers to his castle, retrieving the stolen treasures and then to rid the kingdom of this evil wizard. Gripping stuff!

The game starts with an opened scroll with Crossbow as the title screen. From here you can move to the map screen by pointing your trusty Light Gun at the screen and pulling the trigger. The map screen is very colourful with well animated graphics showing eight adventures to go through on your journey. There's a Desert, Caverns, Volcano, Jungle, Village, River, Drawbridge and last but not least the Castle. At the bottom of the map are two coloured areas of which you must shoot to select your route. It does say in the instructions that you can choose a path otherwise after a few moments a path will be chosen. Well, let me tell you the definition of a few moments there is about three or four seconds! so choose quickly, and pull that trigger! Don't leave it for your path to be chosen for you, otherwise you'll find yourself going backwards and forwards along the same path.

The title of the game derives from the fact that your Light Gun acts as a Crossbow, for both bumping off the nasties and choosing paths. When you first set off you have a choice of two paths from where you are, to the Desert or to the Village. Each path has a different colour corresponding to the coloured sections at the bottom of the screen which must be fired at to select your path.

Once a path is selected, coloured foot steps will appear on the map from your present position to your chosen adventure theme. Each screen is well animated to it's particular scene, The Desert scene for example is animated with cacti and mountains in the distance. Each scene has it's own set of nasties which must be dealt with. As you select a scene you'll soon have various nasties appearing ready to attack your adventurers. These nasties come in various forms eg. for the Desert you have to kill Vultures, Snakes, Ants, Scorpions and of all things Rabbits! So remember next time you're in the Sahara keep away from the Rabbits!

Playing the game is simple just aim at your target and fire. As you begin to knock off the nasties (which constantly reappear) you'll notice your adventurers plod along from the bottom left of the screen. I say plod, they don't seem to be in any great hurry. Each adventurer is a different character, colourful and cleverly animated, their movement is well implemented. You have no real control over your adventurers, your main task is to protect them from the various attackers. Keep a close eye on any nasty getting close to your adventurer, shoot it quickly or you'll lose a companion. When any nasty touches one of your companions, he/she will try to fight back for a short while, giving you a chance to help. Be quick and watch you don't shoot your adventurer, as hitting your adventurers twice will result in them shrivelling up and falling to the ground, dead. The same will happen if a nasty touches them too!

Each scene has a Jewel at the right of the screen which is picked up by your first companion on passing over it. If you succeed in getting at least one adventurer through a certain scene safely you'll be awarded an extra companion. You can have up to a maximum of eight adventurers at one time, each of which you must safely guide through the next scenes. After each successful trek you return to the map where you must choose a path again.

Some scenes have other things to shoot to aid your travelling adventurers. The Volcano scene has a boulder which must be shot at so it will drop forming a bridge over a lava river. The cavern has stalactites which must be shot down to fill crevices in the path.

I was very impressed with the game, it has a lot of depth. The graphics are excellent well thought out and very colourful. Each scene has it's own tasks which must be solved. There's been a lot of good programming here to form a very playable addictive adventure type game. Playing with a Light Gun provides a whole new concept in game play. It's a great pity that there are only the four Light Gun games available, as I'm sure others would have been a great success.

Crossbow is currently available from Micro Discount at £21.50. A little expensive, but if you have a Light Gun, what the hell!

OPERATION BLOOD
from ANG software Holland.
Review by Mark Fenwick

A.N.G. Software are better known for their amusing and informative Megamags. On the software side, puzzle and mind games seem to be the norm. Operation Blood is quite the opposite and has to be one of the best arcade clones currently available for the Atari 8-Bit. If you are a bit of an arcade freak it won't be long before you realise this is a clone of Operation Wolf, and an excellent one at that.

The instructions to the game, like with all our imports, are brief. However, they do try to point out that they are not a violent race and don't want to glorify war as being fun. I don't know if they're trying to say that compulsive playing of this game could be damaging, resulting in you going down to your local shopping arcade armed with an M-16 and wiping everyone out! Then turning around and saying "well, I had been playing Operation Blood on my 8-Bit for 5 hours, and something just snapped!"

The game starts with a nice title screen accompanied by a nice piece of music. From here you can press [OPTION] to see the present hi-score list, or press [START] or fire button to get straight into the action.

The screen layout consists of an information bar at the bottom of the screen showing score, number of Soldiers, Armoured cars and Helicopters to destroy. Each of which is represented by an icon, a soldier followed by a number which have to be killed to complete the level. To the right of these you are shown your rounds of ammo left and missiles. Down each side of the screen is a column of blocks. The left hand blocks represent your power bar, each time you take a hit you lose a block, once all the blocks have gone your life ends. (Remember it's only a game! as your inlay card will tell you) The right hand blocks represent the number of bullets in your current round of ammo, each time you shoot this will depreciate. Great effort has been made to take up as little room as possible for all this leaving the main playing area as large as possible.

As you begin play you can see at the bottom of the screen your quota's each time you kill a soldier etc the amount will depreciate. For level one 50 Soldiers, 5 Armoured Cars and 4 Helicopters. You start play with 7 rounds of bullets and 5 grenades. As the screen scrolls to the right you'll notice a gun sight, this has to be placed over the enemy then press the joystick button to fire. The action is very fast, you'll soon have a screen full of soldiers firing at you as well as throwing knives at you. The graphics are of a very high standard with buildings, towers, oil drums and a perimeter fence with trees behind it in the background. The soldiers are well detailed too and are easily popped off with your machine gun sight. The helicopters and armoured cars take a little more though, you can blow these up with several shots. Alternatively you can opt to blow them up with a single grenade. Throwing a grenade is achieved by placing your gun sight over them then pressing the space bar. A tip I found was to operate the space bar with my toe! as fumbling with the space bar can lose you some valuable power whilst being constantly shot at.

While all this action's going on keep an eye on the bottom left of the screen for power up symbols. These symbols are on the ground and scroll with the screen, each represent various power ups, some obvious, some not. A 'P' will give a few blocks of extra power. A square with an 'F' in it gives a limited time of rapid fire, which does not effect your ammo supply when firing. An offset 'B' gives an extra round of ammo. A missile symbol gives a grenade. To pick up a symbol simply point your sight at the symbol and fire. I've noticed these symbols are random, sometimes you'll get a lot, other times you'll hardly get any. They don't follow in order, you could get say 3 'P's a grenade, it's different everytime. This is good as you cannot follow a pattern everytime which makes each game a chance of luck as well as quick reflexes.

You may find Operation Blood a little difficult at first. I must confess to taking a couple of days just getting off level one. I'm sure there's a hidden cheat in there somewhere but I've not found it yet! Once you've sussed what's what though you can work out what you need to try and save for the following level. I don't know how many levels there are, I've managed to get to level 4 but unfortunately ran out of ammo! I've not got that far since, but I'm still trying! Each level is not a repeat scenario with more to kill. Each scene is different with some excellent graphics.

Overall the game is of high quality, unlike other shoot 'em ups you will not get bored with this one. I definitely recommend this to any games player and when you look at what you're getting for a mere £5.95 it's very well priced. Operation Blood is currently available from Micro Discount. Derek Fern is talking of a Rom version to use with a light gun, which should also be a success. Well done A.N.G. how about an Operation Blood II ?

```
JTR MINER   Y
EEUSGOLD Z C
TLN   N A A R
BBN    T G X E
OAE   A E   X N    N
OTR  R L Z   O I   A
T   IXLXE U   N N RM
JLODEA     M   A C C
AO  N       A H H  A
C UI         O O   P
K FS     BOBYTNUOB
    T      N
          O
         K
        U
  YTROFYTNEWTOUCH
```

8-Bit Classsics
Word search

By: John McIntyre

BUY, SALE & CONTACT SECTION

WANTED.
Does anyone have the ACTION language cartridge that they no longer use? If so, then contact:
Ken Law, 3 Guithavon Valley, Witham, Chelmsford, Essex. CM8 1HF.

Wanted Please.
Games on Cartridge boxed/unboxed.
Write to:
Mark, 55 Bridge Street, Long Eaton, Nottingham, NG10 4QS. Or TEL. (0602) 720597.

Searching for an explanation for the 'C' C8 programming language. Can anyone tell me how to replace the basic 'DATA' statement.
Write to:
D. Carrodano,
202 Impasse des fougeres,
Le Collet Redon-Bas,
La Bouverie,
83520 Roquebrune sur Argens,
FRANCE.

Wanted:
Any old Carts and Eproms, also the "130 XE" hand book.
If you have any of the above for sale, then please contact:
Paul 'Spite' on 061-8722814 (after 6 pm only!)
or write to:
12 Treelands Walk
Ordsall
Salford 5
Manchester M5 3FX

SEVEN SEGMENT DRIVE # INDICATOR
by M.G.Rice

For all the DIY freaks out there, here's a small inexpensive project for all 1050/810 drive owner's.

Last November I was asked to design a drive # indicator for the 1050 using a 7 segment indicator. The final version was resdy and fully tested early january. So far there have been no faults on any of the units under test and I think it's time to make it public for all the DIY nut's (like myself).

Here's how it all works:-

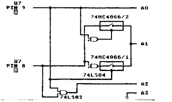
The two switches on the rear of the 1050/810 place a 2-bit binary # on pins 8 & 9 of U7 (6532), this sets the drive ID #.

```
pins 8  9   dec       ID  needed

     0  0 =  0         4   1 0 0
     0  1 =  1         3   0 1 1
     1  0 =  2         2   0 1 0
     1  1 =  3         1   0 0 1
```

As you can see from the #'s above, the binary #'s used by Atari dont match up to the corrisponding ID # (all exept # 2). Because of this a small interface is required.

Below is the actual CCT diagram from pins 8 & 9 up to the address pins of the 74LS47 (BCD LED driver).



Pin 9 of U7 is address 0 (A0) for the BCD driver, and as it is always of the correct value, we can pass it straight to the driver with out any alterations.

Pin 8 of U7 is address 1 (A1) for the BCD driver, and as it's value is only right when pin 9 is at logic 0, we will have to invert it when pin 9 is at logic 1. To do this we use two switches (part of 74HC4066) and two inverters (part of 74LS04). Both switches are controlled by U9. Switch 1 control is connected directly to pin 9 and switch 2 is connected to an inverted pin 9. The inputs to the switches are: switch 1 to inverted pin 8 and switch 2 directly to pin 8. Both outputs are connected together and connected to A1 of the driver (74LS47).

Now all we need is to generate address 2 (A2) for the driver. This is needed, as ID 4 is a 3-bit binary # and we only have a 2-bit # from pins 8 & 9. The third bit is generated by using a NOR (74LS02) gate with both pin 8 & 9 as inputs. When either 8 or 9 is high (logic 1) the output is 0. But when both 8 & 9 are low (logic 0) the output is 1. This produces an address output of binary 100 (dec 4), which is what we need.

Below is a diagram of all the pin connections needed, any pins not listed are to be left unconnected for correct operation.

To mount the display on the front panel cut a small square hole large enough to see just the 7 segments of the LED. The display can be super glued just behind the hole.

I made this unit on a small board with enough room for U7 also, this was done to make the board plug in (no soldering on the main 1050 motherboard). If anyone wishes a completed unit they may contact me through T.W.A.U.G. and I will make them one for 12 pounds (2 for twenty pounds).

At present I am working on a track # indicator for the 1050 drive ......keep reading the newsletters for information (when it's finished).

Parts List :- (obtainable from Maplins Electronics)

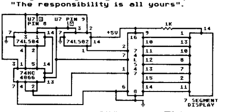
74LS02	YF 02 C	20p
74LS04	YF 04 E	24p
74LS47	QQ 52 G	56p
74HC4066	UF 10 L	36p
DISPLAY	FR 36 P	1.25p
RESISTOR	M 1 K	4p

Other parts for plug in version :-

PINS	JW 59 P	56p x 2
SOCKET 14	BL 18 U	8p x 4
SOCKET 16	BL 19 V	10p x 1
SOCKET 40	HQ 38 R	20p x 1

Plus solder / wires / board.

DISCLAIMER: We would like to make it clear that we, T.W.A.U.G., cannot be held responsible for any damage caused to your hardware through using any material published in our newsletter. "The responsibility is all yours".



CRACKING THE CODE

by Keith Mayhew and Roy Smith

Re-printed by M.Gerum

This article first appeared in "The U.K. ATARI Computer Owners Club" later renamed "MONITOR".

PART 3

Hopefully, the first two parts of this series gave you a good grounding of the basics of the machine and a brief insight into the mathematics of binary and hex numbers. From this point we can move on to some examples and explanations of Assembly Language programs. We assume that you are the proud owner of an Assembler package such as the ATARI Assembler/Editor cartridge for which all of our listings will be written, but if you own a different one then we shall give a brief idea on how to convert the few offending commands. If you do not have an Assembler, you really do need one if you intend to write seriously in Assembly Language.

All examples given in this and future articles are only intended to be fully working programs if they are listed with line numbers. Programs without these are purely to aid learning and will generally not work as they stand.

MNEMONICS.

All of the 6502's instructions are given three-letter labels called mnemonics. The reason for this is simply as an aid to your memory so that you do not have to remember a string of numbers but only a short letter sequence. When using an Assembler you are in fact writing in mnemonics and when the program is 'assembled' it is converted into its numerical equivalents or the actual 'machine code'. So, Assembly Language is Machine Code written in an easy to remember format.

GETTING STARTED!

The first mnemonic we will discuss is the instruction 'LDA'. This means LoaD the Accumulator. In other words when this instruction is executed a number between 0 and 255 will be deposited into the A register of the 6502. Where the number actually comes from is determined by the operand byte(s), remember from Part 2 that the instruction 'LDA' is an op-code and the bytes following it are it's operand bytes. The op-code 'LDA' can be used in many different addressing modes, the simplest is 'immediate' addressing:

```
LDA #40
```

Here we have loaded 'A' with the decimal number 40. The '#' symbol informs the assembler to produce an op-code which will mean 'load the accumulator in immediate mode'. It would be more usual to write the instruction as:

```
LDA #$28
```

This has exactly the same effect as the previous example, only here we have written 40 decimal as 28 hex i.e. $28, where the '$' means hex.

The lowest number that can be loaded into 'A' is 0 and the largest is 255 or $FF (remember the 'A' register is an eight bit register). When this line is assembled by the Assembler it converts our mnemonic code into a single hexadecimal number, so that our line would look like this:

```
$39 $28
```

Where $39 is the op-code and $28 is the operand byte. Note most Assemblers omit the '$' symbol. These two numbers, $39 and $28, would be stored into sequential memory locations ready to be executed. It is a point worth remembering that commercial Assemblers usually produce the machine code listing in hex format, so becoming familiar with hex notation is essential. Consider the following:

```
LDA $28
```

You've probably spotted that the '#' symbol is missing, this is because we are no longer using immediate mode but we are using zero-page or short addressing. In this mode the data to be loaded into 'A' is found at the memory location of $28, and as this is a number below 255 or $FF it will fall in the first page of memory (page 0).

For example if the contents of memory location $28 was $68, then $68 would be read from location $28 and placed in 'A' (once again the number read can only be between 0 and 255). When our line is now assembled the operand byte remains the same but as a different addressing mode is being used then a different form of the op-code is produced thus:

```
$A5 $28
```

Supposing the data, $68, was not to be found in page 0 but was located somewhere else in memory, we would need to use absolute addressing to access it. In this case two operand bytes would be needed to cover the complete memory from 0 to $FFFF. Note that the programmer does not need to identify between these modes as once the specified location exceeds page 0 then the Assembler automatically uses absolute addressing. Let's suppose location $24E5 contains our data, $68, then to load this into the 'A' register we would write the following:

```
LDA $24E5
```

Once again, when assembled, the op-code will be different because of the mode change and also an extra byte will be added for the extended addressing area, as shown below:

```
$AD $E5 $24
```

As you can see the op-code is $AD but the address has been assembled back to front, so that the low byte is placed first then the high byte next. This is done because the 6502 expects them in this form, but note that the friendly

CRACKING THE CODE

continued from previous page

Assembler allows you to write them the normal way round.

It is worth noting that the Assembler will always produce a pair of hex digits for each byte, so that although you are allowed to write a hex number in an odd number of digits the Assembler will pad out the leading zero for you. For example:

```
LDA $5 will be assembled as $A5 $05.
LDA $83C will be assembled as $AD $3C $08.
```

We have shown the accumulator being loaded in three addressing modes, as you know there are more than this but we will come to these later. The other two registers, 'X' and 'Y', are loaded in exactly the same manner as the 'A' register except that different mnemonics are used i.e. 'LDX' and 'LDY'. For example:

```
LDX #$E4 (Loads 'X' in immediate mode).
LDY $38 (Loads 'Y' in page-0 mode).
LDX $6E43 (Loads 'X' in absolute mode).
```

STORE IT AWAY

Now we have seen how to load some data into these three registers, we will now show the instruction to store that data into memory. Once again, all three registers have their own mnemonic and they are: 'STA', 'STX' and 'STY'. Note that unlike 'load', the data must be stored in a memory location, so obviously, immediate mode cannot be used. The following examples show these instructions in the other two addressing modes we have covered so far:

```
STA $12 (Store 'A' in location $12).
STX $4C5F (Store 'X' in location $4C5F).
STY $10EB (Store 'Y' in location $10EB).
```

Here we have stored 'A' in page-0 mode and 'X' and 'Y' in absolute mode. Suppose we wish to store a number into a specific location, we would go about it by loading the number into a register and then storing that register into the desired location, as shown below:

```
LDA #$60
STA $0400
```

We have loaded 'A' with our number ($60) in immediate mode and then stored that information into a location ($0400).

Now suppose that we wish to move some data from one location to another. It would look like this:

```
LDX $7E43
STX $0300
```

Here we have loaded 'X' with the data in location $7E43 (in absolute mode) and stored it at $0300. In these two examples, where we have stored some data and moved some data, the registers used could just as well have been any of the three ('A', 'X' or 'Y').

LOGICALLY SPEAKING

Included in the instruction set of the 6502 are three logical operators with which numbers can be manipulated.

These instructions are 'AND', 'OR' and 'Exclusive-OR'. Their mnemonics are 'AND', 'ORA' and 'EOR' respectively.

These three op-codes only operate with the 'A' register and not with the 'X' and 'Y' registers. Note, due to the standard that all 6502 mnemonics have three letters, the 'OR' function is designated 'ORA' for convenience.

The purpose of the 'AND' instruction is to reset specific bits in the 'A' register to a '0'. The 'AND' function has four possible states, these are:

```
0 AND 0=0
0 AND 1=0
1 AND 0=0
1 AND 1=1
```

As you can see the result of ANDing two bits together is only '1' if both the bits are '1'. If this principle is extended to cover 8-bits (1 byte) then we can use the results to obtain the bit pattern we desire.

```
        10110010    (byte 1).
AND     01110100    (byte 2).
        ____
        00110000    (result).
```

Here we have ANDed byte 2 with byte 1 to get our result. Referring to the table, and working our way from right to left; taking bit zero of byte 1 and byte 2 we have 0 AND 0 which results in 0, next we have bit 1 from both bytes, giving 1 AND 0=0, then 0 AND 1=0, etc. until we have completed all eight bits.

Byte 2 is often refered to as a 'mask' which acts like a sieve, where a '1' in the mask is like a hole through which the bits of byte 1 pass. Looking at our example you can see that the bits in byte 1 above a '1' in byte 2 are copied into the result, also where a '0' is in the mask a '0' is the result.

Suppose we wish to ensure that bits 5 and 7 in the 'A' register are reset to zero, then we would use the following mask:

```
        01011111
```

ANDing this mask onto the 'A' register, no matter what bit pattern was there, because of the zero's in bits 5 and 7 of the mask, the result will also have zero in bits 5 and 7, and the other will remain the same.

The result of the ANDing is automatically returned into the 'A' register. Suppose the 'A' register was loaded with the binary pattern: 10011010 which in hex is 9A, and was ANDed with our mask of: 01011111 (5F hex) so that bits 5 and 7 are zeroed to give the result of: 00011010 (1A hex).

CRACKING THE CODE

continued from previous page

```
        LDA #$9A
        AND #$5F
        STA $067D
```

Here we have loaded 'A' with $9A which is the hex equivalent of our original bit pattern. We have ANDed it with our mask, $5F, so the result is $1A which is now in the 'A' register.

Then we store our new bit pattern into a memory location at $067D. We have used immediate mode to load the 'A' register and to 'AND' the mask, but we could have used any of the three addressing modes described so far.

The second logical operator 'OR', is used to set specific bits in the result to a '1'. The 'OR' function has four possible states:

```
        0 OR 0=0
        0 OR 1=1
        1 OR 0=1
        1 OR 1=1
```

From this table you can see that when two bits are ORed together, the result is always one except in the case of both bits being zero's. Here is an example showing two bytes being ORed together:

```
        01011100    (byte 1).
OR      01101001    (byte 2).
        ________
        01111101    (result).
```

You can see that only bits one and seven of bytes 1 and 2 contain matching zero's thus giving '0' in the result. Suppose the accumulator contains the byte: 10100011 (A3 hex) and we wish to ensure that bits 6,5,2 and 0 are set to a '1' then we would need to OR it with the following byte: 01100101 (65 hex) where bits 6,5,2 and 0 have a '1' in them. Conversely the other bits must be '0's so that the result will be a copy of those bits from the first byte. The result of ORing these two bytes is:11100111 (E7 hex), this is automatically returned into the 'A' register. We would write this in assembler thus:

```
        LDA #$A3
        ORA #$65
        STA $480D
```

This loads the 'A' register with $A3, ORs it with $65, giving the result $E7 in the 'A' register, which is then stored in location $480D.

The final logical operator is 'Exclusive-OR' which is used to toggle specific bits of the accumulator between the two possible states i.e. '0' and '1'. In other words if the bit was a '0' and it was toggled then it would become a '1', if it was toggled again it would revert back to a '0'. The four possible states of the Ex-OR function are:

```
        0 Ex-OR 0=0
        0 Ex-OR 1=1
        1 Ex-OR 0=1
        1 Ex-OR 1=0
```

As you can see the result is '0' when both bits are the same, i.e. '0 Ex-OR 0' and '1 Ex-OR 1'. The bits to be toggled are Ex-ORed by a '1' in the appropriate bit of the second byte. Thus if we have 11111111 (FF hex) as the second byte then Ex-ORing any byte will invert every bit in it, this is called the complement, here is an example of a complement:

```
        00101110    (byte 1).
EOR     11111111    (byte 2).
        ________
        11010001    (result).
```

Note that the result is the exact opposite of the first byte, and also that if it was Ex-ORed with $FF again the the result would lead us back to our original byte. Similarly if we only wish to toggle specific bits then we only set those bits in the second byte, here is an example:

```
        00011101    (byte 1).
EOR     10000000    (byte 2).
        ________
        10011101    (result).
```

In the above example only bit 7 has been toggled with all the other bits remaining as in byte 1. If this result was Ex-ORed with byte 2 again then bit 7 would be toggled back to a zero giving the original byte 1. In assembly language:

```
        LDA #$1D
        EOR #$80
        STA $35E0
        EOR #$80
```

We have loaded the accumulator with $1D, toggled bit 7, then stored the result ($9D) in location $35E0, and finally toggled bit 7 back to give $1D in the 'A' register again.

MORE or LESS?

In Part 2, we showed some examples of adding unsigned binary numbers and we mentioned that when the result of adding two 8 bit numbers together is greater than 8 bits, then the ninth bit is placed into the carry flag.

The 6502 does not have an instruction which will add together 2 bytes directly, ignoring the carry flag. Instead, the 6502 has the instruction 'add with carry', whose mnemonic is 'ADC', which will add 2 bytes together and also add the contents of the carry flag to the result, i.e. adds nothing if C=0 and adds one if C=1. Suppose we wish to add 28 hex to 14 hex, then the first instruction would be to load the accumulator with $14 and then add with carry $28, note we could have loaded with $28 and then added $14, as it would

CRACKING THE CODE

continued from previous page

be the same result.

```
LDA #$14
ADC #$28
```

Notice that 'ADC' has been used in the immediate mode and the result is stored back into the accumulator, as are all operations on the accumulator.

The result of adding these two numbers together could be one of two answers, the reason for this is that the state of the carry flag is taken into account and we did not know if it was cleared or set i.e. '0' or '1'. Therefore to ensure we end up with the correct result we must clear the carry flag before adding the numbers together, the instruction to do this is called 'clear the carry', whose mnemonic is 'CLC'. So, the proper version of this program now reads:

```
CLC
LDA #$14
ADC #$28
```

If the result of an addition is less than 256 (decimal) then only eight bits are used and the carry flag, the ninth bit, will be set to zero again.

With a result of greater than 256 the carry flag will be set to one. If we are doing an addition of more than eight bits then this previous state of the carry flag must be taken into account.

Now we will write a program which will take two sixteen bit numbers and add them together, storing the result as another sixteen bit number, note that the result could actually be a seventeen bit number (including the carry) but we shall ignore this in our example as the two numbers will be reasonably small and therefore not generate a seventeenth bit.

In our example the first sixteen bit number is stored in two eight bit locations, $0600 and $0601 with the low byte stored first, and the second number is in locations $0602 and $0603.

The sixteen bits of the result will be stored in locations $0604 and $0605 with the seventeenth bit still in the carry. The program first clears the carry so that it is zero and then adds the two low bytes of each sixteen bit number together storing the accumulator into the low part of the sixteen bit result, as follows:

```
CLC
LDA $0600
ADC $0602
STA $0604
```

Now we must do the same for the high bytes storing them back into the high byte of the result. We will not clear the carry this time because we know that whatever the result, whether it is more or less than 255 the correct carry will be transfered into the sum of the high bytes, so the rest of the program would read:

```
LDA $0601
ADC $0603
STA $0605
```

At this point the carry flag would reflect the status ot the seventeenth bit, normally zero if the two sixteen bit numbers are small enough.

Let us assume that the first number is $14F9 and the second number is $311A, the contents of the first four locations will now be:

```
$0600=$F9
$0601=$14
$0602=$1A
$0603=$31
```

Adding the two low bytes together we have $F9+$1A=$113. The $13 will be in the accumulator and the '1' is in the carry flag. The two high bytes added together give $14+$31=$45, but as the carry flag is now set another one is added to this byte to give $46 in the result with no carry in the seventeenth bit (the carry flag). So, locations $0604 and $0605 will contain $13 and $46 respectively and so the result will be $4613.

The same technique can be used for even larger additions if necessary, e.g. 24, 32 bits etc. If when adding two numbers a carry is generated for the highest bit, i.e. the result exceeds the number of bits being used, then your program could check for this state and indicate an overflow error, using an instruction which will be covered later.

As with addition there is no instruction that will subtract two bytes directly, but there is an instruction that will 'subtract with carry', whose mnemonic is 'SBC' which will subtract two bytes and also subtract the complement, or opposite, of the carry flag. If we wish to subtract $05 from $2A and then subtract with carry $05, but as in the case of addition where we clear the carry first, we need to set the carry to one. With the carry set, the complement of this is subtracted from our result, i.e. zero is subtracted leaving us with the desired answer.

The following program subtracts $05 from $2A and first sets the carry with the 'SEC' command to ensure the correct result:

```
SEC
LDA #$2A
SBC #$05
```

You may be asking yourself why you do not clear the carry and let the 6502 subtract that directly rather than setting the carry to one and then subtracting the complement, which is also zero? The answer lies in the fact that the 6502 can only perform addition, and therefore has to use the two's complement of the number to be subtracted i.e. the negative of that number, and adds it to the contents of the accumulator. The following shows the 'subtraction' of the two numbers:

```
    00101010 ($2A)
(+) 11111011 (-$05) two's complement of $05.
------------
(1) 00100101 ($25 ignoring carry of one).
```

CRACKING THE CODE

continued from previous page

We can see that in a non-borrow subtraction, i.e. a small number from a large number, the carry is left set as a consequence and is interpreted as 'subtract zero' by any further subtractions. So remember that in subtractions when the carry flag is set to one it means 'subtract zero' and when the carry is cleared to zero it means 'subtract one'.

Let's subtract two sixteen bit numbers and see how the carry flag performs. We will subtract $32CA from $6812. The main number is stored in locations $0400 and $0401 and the number to be subtracted from this is stored in locations $0402 and $0403, the result will be left in locations $0404 and $0405. The first part of our program sets the carry then subtracts the two low bytes and stores the number in the low part of the result:

```
SEC
LDA $0400
SBC $0402
STA $0404
```

Thus the accumulator was loaded with $12 from location $0400 then $CA from location $0402 was subtracted from the accumulator and the result, which is $48 is stored in location $0404. Because the number we subtracted was larger than the accumulator's contents, a borrow of one was made from the high byte so that the $12 became $112 (the one that was borrowed is worth 225 decimal or $100), thus when $CA is subtracted from $112 the answer is $48 and to indicate that a borrow occured the carry flag would be cleared. The second part of the program will subtract the two high bytes in locations $0401 and $0403 leaving the result in location $0405, again we do not have to worry about the condition of the carry in the program as it will be treated correctly by the 6502 in the next subtraction:

```
LDA $0401
SBC $0403
STA $0405
```

Here the accumulator is loaded with $68 from location $0401 and $32 is subtracted (location $0403) from the accumulator leaving an answer of $39, but a further one is subtracted due to the carry being zero (one is the complement of zero), so the result is $38 which is stored in location $0405. The final result of subtracting $32CA from $6812 is therefore $3848.

DECIMAL MODE

As mentioned before the 6502 has a decimal mode and is activated when the 'D' flag of the processor status byte is set to one. This is achieved by using the instruction whose mnemonic is 'SED' or 'set the decimal mode', to revert back to the normal binary mode the 'CLD' instruction is used which 'clears the decimal mode'. The 'CLD' instruction should be used if you are unsure which mode you are currently in, thus ensuring that any additions or subtractions do occur in binary mode. Once the decimal mode has been cleared there is no need to use 'CLD' command again unless you have used decimal arithmetic in your program.

Here is an example of decimal mode addition:

```
SED
CLC
LDA #$48
ADC #$25
STA $0500
CLD
```

The program starts by ensuring decimal mode is set and that the carry is cleared for the addition. The accumulator is loaded with $48 and $25 is added to it and the result is stored in location $0500, and finally we clear the decimal mode so we are back to binary for further arithmetic. Because the decimal mode has been implemented the two numbers $48 and $25 are added together just like a normal sum. So 5 is added to 8 giving 13 where the 3 is placed in the low half of the accumulator and the one is carried automatically into the high part of the sum, so we now have 4 plus 2 plus our carry of one equalling 7. So the result is $73. This internal carry between the high and the low parts has no effect on the carry flag. But any external carry generated from a decimal addition is treated in exactly the same way as for binary addition and is added into the next calculation. Subtraction follows the same rules as for binary subtraction and once again an internal carry is made automatically.

UP and DOWN

If we wish to only add or subtract one at a time, such as a counter, then instead of loading the accumulator with the previous number, either setting or clearing the carry then performing the addition or subtraction of one, then storing the result back into the counter, we could use one of the increment or decrement instructions. These instructions apply to the 'X' and 'Y' registers and any memory location. The increment instructions will add one to the contents of the register or location and the decrement instructions subtract one. If when using the increment instruction the result will 'wrap around' to zero and start again. Conversely, when using the decrement instruction it wraps around from zero to $FF.

The three increment instructions are 'INX', 'INY' and 'INC' which are increment the 'X' register, increment the 'Y' register and increment a memory location respectively. The three decrement instructions are 'DEX', 'DEY' and 'DEC' which are decrement the 'X' register, decrement the 'Y' register and decrement a memory location respectively. the state of the carry flag is not important before one of these instructions is implemented.

MOVE IT

There are six instructions that allow us to transfer data quickly between the 'A', 'X', 'Y' and the stack pointer (SP) registers. As we know, only the accumulator can be used for operations such as addition, subtraction and logical operations, so it is useful to be able to transfer the

CRACKING THE CODE

continued from previous page

contents of either the 'X' or 'Y' registers directly into the accumulator to perform the necessary operations rather than storing it in memory and then loading it back into the 'A' register. The two instructions that perform this function are 'TXA' and 'TYA', which mean transfer the 'X' register into the 'A' register or transfer the 'Y' register into the 'A' register respectively. There are another two instructions to return the data back into the 'X' or 'Y' registers. These are 'TAX' and 'TAY', which mean transfer the 'A' register into the 'X' register or transfer the 'A' register into the 'Y' register respectively.

Lastly there are two specialised commands that transfer to and from the 'X' and 'SP' registers. The 'TSX' instruction transfers the stack pointer into the 'X' register so that it can be examined and the 'TXS' instruction is used to transfer the contents of the 'X' register into the stack pointer to set it to a specific value. However, these two instructions are used very often if at all ina program because the computer uses the 'TXS' instruction to set the stack pointer to $FF at power up, and should therefore not need to be altered.

NOTHING TO DO

The most unusual command in the 6502 instruction set is the 'NOP' instruction which actually means 'no operation' and the 6502 puts it's feet up for a while when this command is executed! No, seriously though, this command can be used as a 'debugging aid' to replace an instruction held in memory so that when the 'NOP' instruction(s) are encountered the program will 'skip' over them without crashing or corrupting the program.

In the next issue we will complete the instruction set for the 6502 and hopefully start on some programming techniques. Finally we suggest after reading this far, you perform a 'NOP' operation and go and put our feet up!

Have a good rest!!! Part four will be in next issue of TWAUG.

THE OL'HACKERS
ATARI USER GROUP INC.

O.H.A.U.G. is an all 8-bit user group in the State of New York, they are producing a bi-monthly first class informative newsletter on disk.

The disk is double sided full of news, views, articles and bonus games and/or utilities. The disk has its own printing utility which you can use to read the content of the disk on screen or make hard copies.

A large PD Library is also available.

Some of the T.W.A.U.G. members are contributing to the OL'Hackers newsletter and the OL'Hackers are contributing to the T.W.A.U.G. newsletter.

For more information on how to contribute to the newsletter write to:

O.H.A.U.G.
c/o A. Pignato
3376 Ocean Harbor Drive
Oceanside, N.Y.11572
U.S.A.

BRING YOUR EIGHT UP TO DATE with power products from COMPUTER SOFTWARE SERVICES

THE BLACK BOX

The BLACK BOX is an add-on board for the Atari 600XL,800XL and 130XE 8-bit computers. It is a T-shaped board that plugs into the PBI port of the XL computer, or the ECI and cartridge ports of the 130XE. Connectors for both types of computers are built into the BLACK BOX so no adapter boards are necessary. A cartridge port is available on the board itself for 130XE users.

The BLACK BOX provides many unique and useful functions. The four primary functions are:-
* RS-232 serial modem port
* Parallel printer port
* SASI/SCSI hard disk port
* Operating System enhancements

The BLACK BOX is $199.95 for the basic unit, and $249.95 with an onboard 64K printer buffer.
Shipping and Handling extra.

THE BLACK BOX ENHANCER

A must for all BLACK BOX owners. The BLACK BOX ENHANCER is a plug-in module for your BLACK BOX, enhancing the printer functions and adding an instantly available, full featured sector editor!

Installation of the BLACK BOX ENHANCER requires one simple solder connection. Only $49.95 plus shipping/handling.

THE FLOPPY BOARD

Our latest and greatest product. The FLOPPY BOARD is an add-on expansion board for the BLACK BOX interface. It allows the use of the same inexpensive floppy drive mechanisms used in IBM computers. The FLOPPY BOARD is the first floppy drive interface to support "high density" floppy drive mechanisms in either 5.25 inch or 3.5 inch. Built into the FLOPPY BOARD are our BLACK BOX ENHANCER and a version of our SUPER ARCHIVER to allow copying of protected disks for 3.5 inch format. Included with the FLOPPY BOARD is our program to <u>read and write</u> to IBM or ST formatted disks. This makes the FLOPPY BOARD the best way to transfer files to and from your 8-bit.

The FLOPPY BOARD is only $149.95 plus shipping & handling.

THE SUPER ARCHIVER

The SUPER ARCHIVER is not just another double density high speed upgrade. Rather it is the most powerful copying and disk editing tool available for the 8-bit computer. Copies of disks are exact duplicates of the originals, eliminating the need for patches,PDB files, or extra computer hardware.

The SUPER ARCHIVER is only $69.95 plus shipping/handling

THE SUPER ARCHIVER II

The SUPER ARCHIVER II edits and copies all <u>enhanced density</u> programs plus retains all the features of the SUPER ARCHIVER.

The SUPER ARCHIVER II is only $99.95 plus shipping $ handling. <u>NOTICE</u>: if you already have THE SUPER ARCHIVER you may upgrade to S.A.II for only $29.95 plus shipping/handling. Software only.

THE BIT WRITER

The Super Archiver BIT WRITER is capable of duplicating even the "uncopyable" Electronic Arts and Synapse Syn-series, which employ 34 full sector tracks. The BIT WRITER must be used with the SUPER ARCHIVER

The BIT WRITER is only $79.95 plus shipping/handling.

For more information on these and other 8-bit products:
CONTACT
COMPUTER SOFTWARE SERVICES
PO BOX 17660
ROCHESTER, NEW YORK 14617
USA

ORDERING LINE: (716) 429-5639
FAX: (716) 247-7158
BBS: (716) 247-7157

or contact T.W.A.U.G. we will do our best to help

```
OUNCES   GRAMMES         OUNCES   GRAMMES           OUNCES   GRAMMES
------   -----------     ------   -----------       ------   -----------
   1/4....7              7 1/4....206               14 1/4....404

   1/2....14             7 1/2....213               14 1/2....411

   3/4....21             7 3/4....220               14 3/4....418

     1....28                 8....227                   15....426

 1 1/4....35             8 1/4....234               15 1/4....433

 1 1/2....42             8 1/2....241               15 1/2....440

 1 3/4....50             8 3/4....248               15 3/4....447

     2....57                 9....255                   16....454

 2 1/4....64             9 1/4....262

 2 1/2....71             9 1/2....269

 2 3/4....78             9 3/4....277

     3....85                10....284

 3 1/4....92            10 1/4....291

 3 1/2....99            10 1/2....298

 3 3/4....106           10 3/4....305

     4....113               11....312

 4 1/4....120           11 1/4....319

 4 1/2....128           11 1/2....326

 4 3/4....135           11 3/4....333

     5....142               12....340

 5 1/4....149           12 1/4....347

 5 1/2....156           12 1/2....355

 5 3/4....163           12 3/4....362

     6....170               13....369

 6 1/4....177           13 1/4....376

 6 1/2....184           13 1/2....383

 6 3/4....191           13 3/4....390

     7....199               14....397
```

OUNCES TO GRAMS PRINTER (C) R. BRADLEY