STOS Newsletter

Issue 11/12 STOS User Club

❑ Editorial

Ta-da ! The biggest issue yet of the STOS Newsletter. Welcome to the bumper double issue 11/12. We have reviews for you of 2 games from a new software company **DIGITAL DIMENSIONS**, extended editions of the Absolute Beginners and Learners series (with Professor Speck), and 5 ten liners, including two arcade games, a puzzle game, a database and a sequencer ! The second of the arcade games is FULLY documented, line by line so that you can learn how to write 10-liners !

Richard Gale has sent in a program listing, which lets you convert STAC fonts into STOS fonts, and we also have articles on how to get more than 15 sprites on screen at once, and on which STOS commands are the fastest. Not to mention, a full index listing of all the programs and articles that have appeared in the newsletter since issue 1 !

❑ Resubscription

Yes, it's that time of year again ! All STOS club members (except for the winner of our survey competition) need to re-subscribe to the STOS Club for another year.

This Newsletter was compiled and edited by Aaron Fothergill using Timeworks DTP on the ST. All articles are by him unless stated. Printing and P.D Library by Sandra Sharkey.

As with last year, we've got a special offer for re-subscribers, which is exclusive to STOS Club members. A disk containing nearly 100 extra commands for STOS (as extensions), giving STOS plenty of extra power. This includes extra commands for the STE giving you access to ALL the STE's joystick ports (and twin sticks on a normal STFM) and the DAC sound chip, the STOS SQUASHER extension, previously only available with GAMES GALORE, Stallion Software/ Riverdene PDL's 512 colour

Inside this issue:

EDITORIAL

routine from STOS Paint Master, so that you can use 512 colour pictures in your programs. Not to mention the USEFULL extension (including source code), Francois Lionet's STOS Tracker extension (4 channel sampled sound music in STOS !) and the TOME Junior extension (entry level version of TOME V2.1) . From issue 13 onwards, we will be covering programming using all these extensions, and with no STOS PLUS planned, this is the best way of improving STOS. We will also be giving membership cards to STOS Club members with a personalised code, so that the STOS/AMOS Helpline doesn't get jammed by calls from non-members. Chris Payne from Mandarin sent the helpline phone number in to New Computer Express, thus causing problems, as the line is only for STOS Club (and AMOS Club) members.

Add a years access to the STOS Helpline and it's not bad for a mere £10 (£15 overseas) !

You will find a resubscription form on the back page.

So who won the Reader's Survey competition ? (I'll keep you in suspense for a while!)

❑ STOS Paint Master

The excellent STOS Paint Master package by Stallion Software (see issue 10's review), is now being sold by Riverdene PDL instead of direct from Stallion Software. So please take note of the change of address and make your cheques payable to Riverdene PDL instead of Stallion Software.

❑ STOS P.D Library

The P.D Library is currently undergoing a major transformation ready for issue 13. All the disks are being converted over to double sided format (all 9 sector 80 track, no silly mega formats), and all the programs that can, are being converted to version 2.6 of STOS (the latest version). Any programs which cannot be converted (due to not supplying source code) or that are not up to date will be dropped from the P.D library. If you have any of your programs on P.D, then get them upgraded as soon as possible !

If you still only have a single sided drive in your ST, then Sandra will still supply the programs on single sided disks, but you will have to pay for two disks instead of one.

for further information, please send a stamped, self addressed envelope to Sandra for a catalogue.

❑ Letters

Quite a lot of you have mentioned the lack of reader's letters in the newsletter recently. Most of the letters are along the lines of "Why can't we have more reader's letters ?". Unfortunately, most of the letters I get as requests for help ask exactly the same questions as in earlier issues of the newsletter. As STOS club members already have these newsletters, it is a bit silly to print the same reply twice, when a different tip can be placed there. I do want to print reader's letters though, so if you do have any new STOS problems, or a solution to one (or both !) then send them in, I'll be only too glad to print them, as writing and editing a newsletter is so much easier when someone else has done the writing bit ! Thank you, bye the way, to John Fermor of Margate, for his second accurate clock routine.

❑ Binders

STOS Club Binders have been suggested by several people who have written in, which of course is a great idea. The STOS package should have been supplied in a folder in the first place ! So I am working on a way of creating a STOS Club Folder/Binder, that will contain the

(Continued on Page 4)

Digital Dimensions

Games Review:

Jiggers & Fruit Pursuit

Digital Dimensions are a new software company, set up to design and market good quality ST budget software. So far, they have released two games, Jiggers and Fruit Pursuit. Both of which show that they intend to make their prescence felt in the industry.

When bought by STOS Club members, Digital Dimensions also supply the source code free with their games, a very generous offer, which is very worthy of mention.

The lads at DD are on the lookout for more games to market, so send in your games to them for review. Send them to:

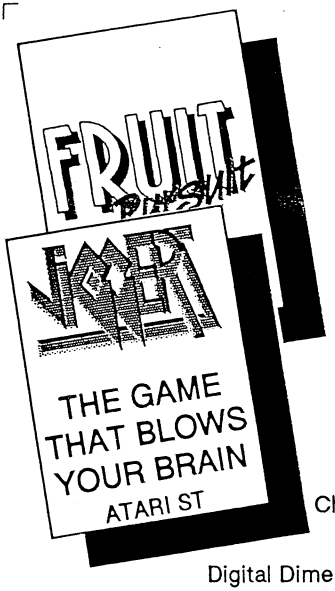
Digital Dimension,
Winterbourne,
Bristol.

Here follows reviews of Jiggers and Fruit Pursuit:

Jiggers (£5.99)

This game reminds me of some of the more devious puzzles invented by that mad Hungarian, Mr Rubik.

The idea of Jiggers is to change a grid of squares, so that they are all the same colour. By clicking on a square, it and all the squares around it change by one colour, going one way if the left button is pressed, and another if the right button is pressed. Eventually, you will end up with



Fruit Pursuit - A game that combines the addictive qualities of a fully featured fruit machine with the strategy of a traditional board game.

Only £5.99 inc P&P

One armed bandit lovers will be in their element here.

(ST User - Feb 91)

Jiggers - A game of strategy and quick thinking, that follows the tradition of all great puzzles. A simple concept that hides complex and mind bending gameplay.

Only £5.99 inc P&P

Try it. It's definitely worth the money.

(ST User - Feb 91)

Source code included for all STOS Club members.
Cheques/postal orders to be made payable to Digital Dimension.
To order or join our free club write to :
Digital Dimension, Winterbourne, Bristol,

Digital Dimension

all the squares the same colour. Which is like saying all you have to do to complete a Rubik's cube is to twist it a few times ! This game is difficult and incredibly addictive if you are into logic puzzles. Level one involves 3x3 squares which I managed to complete after only about twenty attempts (There is a time limit on each level). Level two has 4x4 squares, and level 10 has 12x12 squares and should be rewarded with instant Mensa membership if you can complete it !

Fruit Pursuit:(£5.99)

So far, every fruit machine "simulator" to be released on a computer has been incredibly boring, due to the fact that there is no real winning involved. Fruit Pursuit takes the fruit machine one step further and adds a snakes and ladders type board game to it.

You progress along the board by scoring points on the fruit machine (again with a time limit), the different matches of fruits scoring a varying number of moves.

The fruit machine itself has hold, super hold and a feature win system, with 6 different features, and you must swap some of your moves for strength points to pass through the three zones on the board.

Fruit Pursuit can be played as a 1-4 player game and has a little more luck and reaction time involved in it than Jiggers, but fortunately is less of a wear on the old grey matter. The good thing about Fruit Pursuit is that you can play it straight away, almost without reading the instruction manual, thus making it a game you can easily get into.

The only thing I can fault it on, is the fact that it has too many pauses for the player to press things. For instance, every go a one button alert box comes up, which you have to click on, before you can get at the fruit machine buttons. This box isn't really needed, and slows the gameplay down a little, however, this isn't a disastrously bad point and the game doesn't suffer too much from it, it's just annoying that's all.

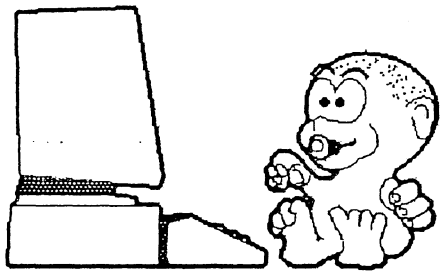
In all, Jiggers and Fruit Pursuit are very good games. Neither involves incredibly graphics, neither comes with a free T-shirt and neither of them has a complete soundtrack written by some so called professional musician that the music industry has never heard of (Although both make good use of sound effects), but both Jiggers and Fruit Pursuit are good wholesome fun, either as a completely mind boggling brain teaser, or as a game to play with your friends like Fruit Pursuit. We should be seeing great things from Digital Dimensions. By the way, DD are also offering free membership to the "Software Pirates Club", which aims to eliminate software piracy by backing reasonably priced games. the SPC offers a regular newsletter with up to date news on all DD's new software, special offers and competitions, reviews off expensive full price software and reviews of all the hottest PD entertainment software. I think they'll worry quite a few people with the name of the club, but it is a good way of backing budget software.

Resubscription

Remember, it's time to resubscribe to the STOS newsletter ! For another year's worth of information packed newsletters, access to the STOS helpline, and an exclusive free disk with nearly 100 new STOS commands !

This is the **ONLY** completely STOS related club with the full backing of the original STOS programmers !

ABSOLUTE BEGINNERS



Good news for all you beginners !

Work has started on a book specifically written to teach programming from the ground up in STOS (and AMOS). "the book" (no title yet) will be based around the Absolute Beginners and Learners series in the newsletters, and will take you from your first STOS command all the way up to writing your first arcade game. As it is aimed at beginners, there will be no annoying jargon or cases of "I thought you already knew that", but you will be able to start at any chapter of the book if you already have a grounding in Basic. So far the plans for the book are along the 80 pages + Disk line, with the programs on the disk being interactive (NOT just a listings disk). No prices or release date as yet (I've only just started writing it !), but watch this space, as STOS Club members will get a special discount on the book.

Right, plug over, let's get on with the programming ! As you can see, we've got a double helping of Absolute Beginners here, so I'm going to do some explaining for those of you who are having problems with the punctuation marks used in programming. As the STOS manual doesn't cover them and many people have started with STOS as their first programming language, this is not unusual. Those of you who are comfortable with these terms, simply skim through in case you missed anything and then get on to the next part of the article which explains what arrays and the Read and Data statements do.

❑ Punctuation...

Various punctuation marks are used in Basic, for various jobs. Usually they work to seperate parameters like the comma, and some are used to mark out particular things, like the Quotation marks " (refered to simply as "Quotes"), which are used to mark out strings. Some other symbols are used, such as the dollar sign **$**, again used with strings. Here are the most popular ones, what they are generally called, and what they are used for...

, Comma... Used to seperate parameters for functions and commands, e.g:

```
A=max(A,3)
```

Normally, anywhere a command or function needs more than one parameter, they will be seperated by commas. Sometimes the **TO** word is used, for instance in the command..

```
for A=1 TO 5
```

or

```
screen copy 1 TO 4
```

but most other commands use commas...

" Quotes... Used to declare strings, for instance, when and printing to the screen anything in quotes is **$** String output directly. e.g

```
Print "Hello There"
```

You must make sure that there are quotes on both ends of the string (STOS compensates but it can cause problems)

Normally in the English language, two types of marks are used (Sixty Sixes and Ninety Nines as they are commonly known), however, in programming, only the one type of mark is used. **SHIFT 2** gives you a quote.

Generally, anywhere a string parameter

is going to be used, you either need a dollar sign **$** to show that the variable being used is a string, or quotes are used around the direct string. For example, the **FLIP$** function requires a string to be used, this can either be a string variable (using the **$** symbol) or a constant string with quotes at either end. e.g

```
A$=Flip$("Hello There") : rem
 erehT olleH is stored in A$
```

or

```
A$=Flip$(S$) : rem whatever
 string is in the variable S$ is
 reversed and stored in A$.
```

Note that when speaking to other programmers, the **$** is pronounced as **STRING**, as it is only ever used to show strings. The only exeption to this rule is when the $ itself is to be printed out. e.g

```
Print "$20,000.00 is a lot of
 money"
```

where the symbol is being used to show money rather than strings (otherwise the American economy would become rather tangled !)

 # Hash Used to show that a variable is a Floating Point or "real" variable (i.e it needs to be able to store fractions). e.g

```
A#=12.56
```

As STOS basic has been optimised for speed, any calculations are normally done in integer form (numbers are rounded down) unless you force them to be floating point. Note that the hash is only used with variable names. When making a constant floating point, you simply need a decimal point in it. e.g

```
Print 12.6+3.4
```

A floating point variable or constant anywhere in an equation will force STOS to do the whole calculation in floating point mode.

^ Power This symbol (Shift 6) is used to show "to the power of" in an equation, such as 5 to the power of 3, which would be 5^3 (this is effectively 5x5x5).

Note that Hashes are also used when file handling, but we'll cover this in a later issue.

/ Divide and *** Times** As the normal divide and multiply symbols used in and maths look rather like the **+** symbol and lower case **x** on most screens, different symbols are used in programming to show these functions. Note that the "Bottom left to Top Right" slash is used for divide (the one next to the right shift key). Please excuse the bad English in calling the multiply function "Times", but it is a lot easier to hear A equals B times three, than A equals B multiplied by 3.

( ) Brackets These are the standard brackets used (the round ones above 9 and 0. ALL Functions use brackets to mark out the parameters, and arrays also use them to show the index (see the later part of the article). Normally, in conversation, the line:

```
A=SGN(B)
```

would be spoken as A equals SGN brackets B, as a quick way of saying that the B should be in brackets. Basically though, all Open (brackets should have a matching Close) bracket after them in the equation, so:

```
A=Max(0,Min(3,A))
```

is legal, wheras:

```
A=Max(0,Min(3,A)
```

would give you a syntax error...

The Square and curly brackets are not used in STOS commands.

: Colon Used to seperate commands in

a listing, e.g

```
A=B+3 : Print A
```

not to be confused with little brother ; **semi-colon**, which is not used.

Hopefully this guide will have helped you get over some of the annoying confusion involved in first learning to program. Remember that there are more details on what Variables, Commands and Functions are in issue 7's Absolute Beginners article.

Now for some more on STOS Basic's commands...

Using Read,Data and Arrays

A little bit of history here... The **READ** and **DATA** statements were intended to be used in Basic listings to replace the **INPUT** command. When writing out your Basic program in the days of yore (up to a few years ago) for the program to be typed in at some distant technical college, you had to write down a list of data to be entered into the program if you were going to use **INPUT**. A typical listing went thus:

```
10 Input A
20 Input B
30 For C=1 to B
40 Print A*C
50 Next C
data 5,12
```

The computer operator who had just typed in your program would then run it and type in the data values. You would then get back a printout of the results about a week later (I did my 'O' level like this !). Obviously, this meant extra work for the operator, and the possibility that an error would occur when typing in the data. This was in the days when the "oops wrong envelope" bug used to ocur, as the homework envelope of CBM Pet listings was often sent to the tech by mistake. As the operator tried in vain to type an advanced (at the time) version of Basic into a very primitive mainframe, all sorts of syntax errors would appear. I remember a printout coming back from a several hundred line PET Basic with syntax errors on almost every other line, yet the operator still tried to run the program !

Obviously, it would be useful to be able to enter data into the program without typing it in, preferably by making it part of the program. Thus **READ** and **DATA** were born.

The **READ** instruction works exactly like the **INPUT** command. It can accept all types of variables, and like Input can read in more than one variable at once. e.g

```
READ A,B,C,D$,A#
```

(remember that **A#** is a completely different variable from **A**)

The values to be read in, are stored in the order that they are to be read, as data statements. e.g

```
DATA 1,2,3,"Hello", 3.1415
```

Note that Data statements can have multiple values after them, or they can be split over several statements. e.g

```
DATA 1,2
DATA 3,"Hello"
DATA 3.1415
```

as long as the data matches what the **READ** statement is going to read in...

Thus, you could write a program that reads in a series of names and tells you the length of them.

```
10 Read A$ : Rem read in a
name
20 If A$="End" then goto 80
```

```
30 N=Len(A$) : rem Len
returns the length of the
string into N
40 Print A$;" is";N;"
characters long"
50 Goto 10 : rem read in
another
60 End
70 data "Aaron","Adam",
"Peter"
80 data
"Richard","Christopher"
90 data "Francois","Sandra"
100 data "End"
```

In this program, I have used a "rogue" value to tell the program to stop. When the read statement reads in the **"End"** data, the check on line 20 stops the program. Rogue values can be used for numbers as well, and are usually values that would not normally be used (-999 is a favourite), so if the normal values used where 1-12, then your rogue value could be any number outside this range.

One thing you must make sure of, is that your data matches up with the read statements, otherwise you could get type mismatch errors, where the Read statement is expecting a numeric variable, and instead, a string is being read.

When STOS starts running the program, it looks for the first **DATA** statement it can find, and positions the **DATA POINTER** on the first piece of data. Every read done moves this pointer on one. When it runs out of data on a line, it looks for the next **DATA** statement in memory and starts again there. If there are no more, then you will get an **"Out of Data"** message.

You can use the **RESTORE** statement to either point the data pointer to start on a particular line or get it to restart from the begining of the data. Using **RESTORE** with a line number will tell STOS to start reading the data from that line. With no line number makes STOS look for the first data statement again.

Array variables are used to store several numbers within the same variable name. For instance, if you wanted to keep six numbers in memory, you could do:

```
10 Read A
20 Read B
30 Read C
40 Read D
50 Read E
60 Read F
70 data 1,12, 43, 64, 95,26
```

Thus storing each value in a different variable. However, this can be awkward if you want to use the numbers in a table. Say, for instance, that you wanted to be able to input a number from the user, and then print out the relevant variable. With the above example, this could be achieved using a lot of **IF...THEN...GOTO** statements (or an **ON n GOTO**). However, there is a better way..

In an array variable, you can determine how many numbers (or strings) you want to be able to store. If you imagine a normal variable as a box that you can store a number in, an array variable would be a set of boxes linked together, that can be indexed by another value. In our example, we could define the array variable **A()** as having 6 values, read all the data into the array, and then directly access any of the values. e.g

```
10 DIM A(5) : rem this tells
STOS that the A() array will
be able to store 6 values
```

```
(from value#0 to value#5)
20 For B=0 to 5
30 Read A(B) : rem B is used
 as an index
40 Next B
50 Input "Enter a number from
 0-5 ";N
55 rem the next two lines
 make sure that N is in the
 range 0-5
60 If N<0 then goto 50
70 If N>5 then goto 50
80 Print "The value of
 A(";N;") is";A(N)
90 goto 50
100 data 1,12,43,64,95,26
```

If you run the above program, it will read the data into the 6 seperate locations in the **A()** array. When you enter a value for **N**, the program will display the contents of **A()** number **N**.

Array variables can be used for the same calculations as any other variable, and can be integer,floating point or string. Note that an array variable with the same name as a non array variable is a completely seperate variable. So **A()** is a completely different variable from A, and in fact the **A** variable could be used to index **A()**, although it would look confusing !

To set up an array, you must use the **DIM** command, which follows the syntax:

DIM variable(*number of elements*)

The number of elements can be practically any value, and this will be the amount of seperate values that can be stored in the array. It is also possible to have multi-dimensional arrays, such as: **DIM A(9,9)** : which would set up array A as a 10x10 grid (remember 0-9), this multi-dimensioning can be done to almost any level, although you would be hard pushed in your program if you had to do: **DIM A(5,2, 100, 23, 59, 87, 10, 6)**

which would be an array 6x3x101x 24x60x88x11x6 values in size, and would probably run you out of memory !

To make life easy (and your programs shorter) you can define more than one array with the same **DIM** statement. Simply seperate the array declarations with commas. e.g

DIM A(12),B(12),C(4,93) etc

Just because this arrays article is tied in with the read/data article does not mean that you can only use arrays with read and data. In fact you can use arrays to do anything that you can do with normal variables. For example, the following routine counts the occurence of the 6 numbers returned by a dice in a series of 20 random throws.

```
10 DIM A(6)
20 For T=1 to 20
30 R=Rnd(5)+1 : rem this
 generates a number from 1-6
40 A(R)=A(R)+1 : rem add one
 to the count for this value
50 Next T
60 For N=1 to 6
70 Print N;" Was
```

10-LINERS

☆STOCKING FILLERS☆☆☆

A bumper pack of 10-liners this issue, including an old (old old) arcade favourite, a puzzler, a sequencer and a database ! These cover a range of programing tricks, including sprite manipulation, collision detection, screen manipulation,sequencial & random access files, arrays (including printing them), sound and several control tricks.

❑ Brickout....

Based upon the old arcade favourite, brickout is a simple game where you have to knock all the bricks out of a wall using a bat and ball. Control is similar to last issues PING game, in that you control the bat with the mouse (horizontal movement). Each level, more rows of bricks will appear, making the game more difficult. Spin can be put on the ball by the bat being in movement when the ball hits it.

For this game, you will need some sprites...

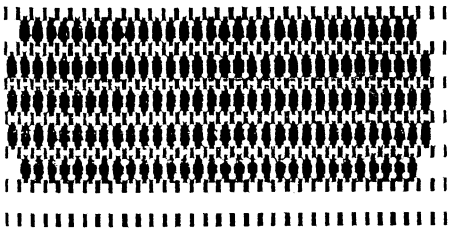
Sprite 1: Player's bat (32x5 pixels) hotspot in bottom right

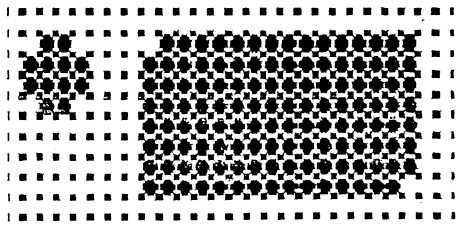
Sprite 2: Ball (4x4) pixels, hotspot in centre

Sprites 3-8: Bricks (16x8 pixels), hotspot at 1,0. Various colours.

Once these sprites are done, simply type in the listing and enjoy !

```
10 dim T(18,10) : mode 0 : key o
 ff : curs off : change mouse 4:
 limit mouse 0,190 to 303,190 : A
 =hunt(start(1) to start(1)+
 length(1),"PALT")+4 : for B=0
 to 15 : colour B,deek(A+B*2) :
 next B : set zone 125,0,0 to 7,1
 99 : set zone 126,312,0 to 319,19
 9 : set zone 127,0,0 to 319,7 :
 LVL=1 : SCRE=0 : LIVES=3 :
 while LIVES>0
20 ink 1 : bar 0,0 to 319,7 : bar
  0,0 to 7,199 : bar 312,0 to 319
 ,199 : for B=1 to LVL : for A=0
  to 18 : T(A,B)=1 : set zone B*
 19+A+1,A*16+8,B*8+16 to A*16+2
 3,B*8+23 : sprite 1,A*16+8,B*8+1
 6,(B mod 8)+3 : update : put
 sprite 1 :next A : next B : K=0
  : Y#=185 : X#=12 : DX#=rnd
 (20)/10.0 :DY#=-1 : locate 0,0
 : paper 1 : pen 0 : print
 "SCORE" : gosub 100 : repeat
30 XO=X : X=x mouse : XO#=X#
 : YO#=Y# : sprite 1,X#,Y#,2 :
 update : X#=X#+DX# : Y#=Y#+
 DY# : Z=zone(1)-1 : if Z>=0 and
 Z<124 then reset zone Z+1 : A=
 Z mod 19 : B=Z/19 : T(A,
```





```
B)=0 : ink 0 : bar 16*A+8,16+8*
B to 16*A+23,23+8*B : SCRE=
SCRE+10 : inc K : X#=XO#-
DX# : Y#=YO#-DY# : DY#=-
DY# : bell : gosub 100
40 if Z=126 then Y#=YO#-
DY# : X#=XO# : DY#=1 :
shoot
50 if Z=125 then Y#=YO# : X#=X
O#-DX# : DX#=-abs(DX#) :
shoot
60 if Z=124 then Y#=YO# : X#=X
O#-DX# : DX#=abs(DX#) :
shoot
70 C=collide(0,15,3) : if C then
DY#=-1 : DX#=max(-2,min(2,DX#
-(X-XO)/10.0)) : bell
80 until K=LVL*19 or Y#>185 : L
VL=LVL-(Y#<=185) :
LIVES=LIVES+(Y#>185) : wend
90 locate 0,10 : paper 0 : pen 4
: centre "GAME OVER" : end
100 locate 6,0 : paper 1 : pen 0
: print SCRE;" "; : return
```

PUZZLER

Many years ago, some bored office executive invented the sliding block puzzle. Where a grid of numbered tiles (with 1 missing) had to be slid around to get them into the correct order. This version allows you to load in a picture in either low or High resolution (Yes, this one works in High rez !). The computer will shuffle the tiles, and you must re-organise them. Simply clicking on a tile with the mouse cursor will move it into an empty square if it is next to it. There are 2 different levels of difficulty, either a 4x4 or a 5x5 grid, and you can play the game with numbered tiles by selecting QUIT from the file selector. If you do this, the game will not load a picture, but simply create numbered tiles (Tile 1 is the blank).

No sprites are required for this listing...

```
10 cls : N$="" : input "Difficult
y Level (1-2) ";D : D=D+3 :
for A=1 to D*D : N$=N$+chr$(A
) : next A : if mode<2 then
mode 0
20 reserve as screen 5 : key off
: curs off : X=640/(divx*D) :
Y=400/(divy*D) : dim B(D-1,D-
1),S$(D*D-1) : F$=file select$
("*.*","Load a Picture (Select
Quit for numbers)",1) : if F$
<>"" then load F$,5 : get
palette (5) else gosub 100
30 for A=0 to D-1 : for B=0 to
D-1 : S$(A+B*D)=screen$(5,A*
X,B*Y to A*X+X,B*Y+Y) : R=max
(1,rnd(len(N$))) : B(A,B)=asc(mid
$(N$,R,1))-1 : N$=mid$(N$,1,R-
1)+mid$(N$,R+1) : next B : next
A : if F$<>"" then S$(0)=
screen$(back,0,0 to X,Y)
40 WIN=0 : repeat : C=0 : for A
=0 to D-1 : for B=0 to D-1 :
set zone A+B*D+1,A*X,B*Y
to A*X+X-1,B*Y+Y-1 : C=C-
999*(B(A,B)<>A+B*D) : next B :
next A
50 WIN=(C<999) : M=0 : cls back
: for A=0 to D-1 : for B=0
to D-1 : screen$ (back,A*X,B
*Y)=S$(B(A,B)) : M=-(B(A,B)=0)*
(A+B*D)+M : next B : next A :
```

```
  screen copy back to physic
60 while mouse key=0 : wend : Z
 =zone(0)-1 : if Z=-1 then 90
70 A=Z mod D : B=Z/
 D : if B(A,B)=0 then 90
80 if (Z>M-2 and Z<M+2 and Z/
 D=M/D) or Z=M+D or Z=M-
 D then swap B(M mod D,M   /
 D) , B(A,B) else bell
90 until WIN<>0 : screen copy 5
 to back : screen copy 5 to
 physic : locate 0,10 : centre
 "Puzzle Complete!" : wait key
 : end
100 hide : logic=5 : for A=0 to
 D-1 : for B=0 to D-1 : ink-
 (A+B>0) : bar A*X+1,B*Y+1 to
 A*X+X-2,B*Y+Y-2 : paper-
 (A+B>0) : pen 1+(A+B>0) :
 locate xtext(A*X)+1,ytext(B*Y)
 +1 : print A+B*D+1 : next B :
 next A : logic=physic : show :
 return
```

□ Database...

Just to prove how simple it is to use random access files, here is a full index card system in 10 lines. There are no search and replace facilities, but you can create new records, edit them, move backwards and forwards through the records and also print them, basically what it would have taken a whole bank of PC's to do only a few years ago.

Because it requires a file to work with, the first time you run the program, it will create a workfile and stop. You can then run the database normally.

Again this program will run in high resolution (as well as medium), although this one uses a simple keyboard menu rather than mouse control.

Sprites in a database, you've got to be either kidding or Douglas Adams ! (Might be interesting though !)

```
10 dim F$(7) : RN=1 : open #1,"R
 ","DATABASE.DAT" : field #1,
 321 as R$ : if mode<2 then
 mode 1
20 on error goto 100
30 get #1,1 : NR=val(R$) : RN=2
 : key off
40 cls : print "Record#";RN-
 1 : print "(f)|";space$(40);"|" :
 if RN<=NR then get #1,RN : for
 A=0 to 7 : F$(A)=left$(R$,40) :
  R$=mid$(R$,41) : next A else
 for A=0 to 7 : F$(A)=space$(40
 ) : next A
50 for A=0 to 3 : print A;")";F$
 (A) : next A : print "Q)uit , E)
 dit, N)ew , P)rint" : A$="" :
 while A$="" : A$=upper$(
 inkey$) : wend : if A$="Q"
 then R$=left$(str$(NR)+space$
 (360),360) : put #1,1 : close #1
  : end
60 if A$="E" then R$="" : print
  "Editing Record #";RN : for A
 =0 to 3 : print A;")"; : line
 input "";F$(A) : R$=R$+left$(F$
 (A)+space$(40),40) : next A :
 put #1,RN : goto 40
70 if A$="N" then RN=min(NR+1,R
 N+1) : cls : print "Record #";
 RN : print "(f)|";space$(40);"|"
  : R$="" : for A=0 to 3 : print
```

```
 A;")"; : line input "";F$(A) :
 R$=R$+left$(F$(A)+space$(40),
 40) : next A : put #1,RN : NR=
 RN : goto 40
80 if A$="P" then lprint "Recor
 d #";RN-1 : for A=0 to 3 :
 lprint A;")";F$(A) : next A :
 goto 40
90 RN=max(2,min(NR,RN-(A$=
 "+")+(A$="-"))) : goto 40
100 R$="1"+space$(320) : put #1,
 1 : print "Initialising Database,
 re-run" : end
```

Sequencer

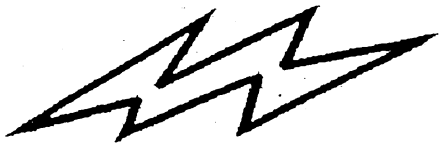
One for the kiddies of music buffs this, (although some of you might find it preferable to the STOS music editor). A simple real time, monophonic (one note at a time) sequencer program. Including Record, Play, Save and Load. The program uses the ST keyboard to play it, the notes being as in diagram 1.

MIDI buffs might want to convert it to work from a MIDI synth instead of off the ST keyboard. Read up on how to access MIDI from STOS (in issues 4-7 of the newsletter) and let me know how you get on, it's not that hard, although it will work out at around 15 lines. If you can get it into 10 lines without being to messy, then send it in!

```
10 dim EV(2000),EV$(2000) : mod
 e 0 : key off : curs off : hide
 on : print "10 Line Sequencer"
 : K$ = "ZSXDCVGBHNJMQ
 2W3ER5T6Y7UI9OOP[=]" :
 C$="RPLSQ" : click off
 : OCT=24 : envel 9,20000 :
 print "By Aaron Fothergill.
 Press a key" : wait key
20 N=0 : A$="" : while N=0 : NE
 V=0 : cls : print "R)ecord,P)lay
 ,L)oad,S)ave or Q)uit" : A$=""
 : while A$="" : A$=upper$
 (lnkey$) : wend : N=instr(C$,A$)
  : wend : timer=0 : on N goto
 30,50,70,80,90 : goto 20
30 A$="" : while A$="" : A$=
 upper$(lnkey$) : wend : EV$
 (NEV)=A$ : EV(NEV)=timer :
 N=instr(K$,A$) : if N then play
 1,N+OCT,0 : inc NEV : if NEV=
 2000 then 20
40 if A$=" " then 20 else 30
50 while timer<EV(NEV) : wend :
  N=instr(K$,EV$(NEV)) : play 1,
 N+OCT,0 : inc NEV : if EV$(
 NEV)<>" " then 50
60 goto 20
70 show : F$="" : while F$="" :
  F$=file select$("*.MUS","Load
  a song",1) : wend : hide :
 open in #1,F$ : A=0 : repeat :
 input #1,EV(A),EV$(A) : inc A :
  until EV$(A-1)=" " :
 close #1 : goto 20
80 show : F$="" : while F$="" :
  F$=file select$("*.MUS","Save
  a song",1) : wend : hide :
 open out #1,F$ : A=0 : repeat
 : print #1,EV(A) : print #1,
 EV$(A) : inc A : until EV$(A-
 1)=" " : close #1 : goto 20
90 print "Goodbye" : end
```

TESTING..TESTING..

Speeding



Quite a few people have asked me which is the faster of the looping commands in STOS. Here are some results which should stop quite a few arguements. All the tests where run using programs as short as possible with each of the looping commands and timed in 50ths of a second (Clicks).

Test 1: Count to 10000 in variable A.

FOR..NEXT (integer) 33 clicks

FOR..NEXT (Float) 114 clicks

REPEAT..UNTIL 141 clicks

WHILE..WEND 146 clicks

IF..THEN (with GOTO) 140 clicks

So as you can see, even using floating point varaibles, **FOR..NEXT** loops run far faster than anything else (mainly because all the others need to use an **inc A** command).

Test 2: Infinite loop, exited after a count of 10000

FOR..NEXT (with step 0) 197 clicks

REPEAT..UNTIL 271 clicks

WHILE..WEND 273 clicks

IF..THEN GOTO 276 clicks

so again, **FOR..NEXT** wins by a considerable margin, however, for this type of loop, a more direct version of **IF..THEN** goto can be used which times at **140** clicks. For example, the test program goes like this for the **FOR..NEXT** loop

```
10 Timer=0
20 for A=0 to 1 step 0
30 inc B : if B=10000 then A=2
40 next A
50 Print Timer
```

wheras with **if..then..goto** it can be shortened to

```
10 Timer=0
30 inc B: if B<10000 then 30
50 Print Timer
```

So for large game loops, this is more practical, especially as it can be jumped out of without worry.

Another point to work with if you are going for speed in your program is that multi statement lines run faster than all the same statements on seperate lines, proving the spagetti programming point of view !

HELP !

If you are stuck and need help with your programming. Just phone the

STOS Helpline
(Shadow Software)

It is best if you phone between 1pm and 7pm U.K Time, as I'm most likely to be in then.

Overseas members please note !!!! Check what time it is in the U.K before you phone, it isn't funny being woken up at 4a.m !

Repeat..Until as much as possible instead of using If..then to get out of a **GOTO** loop, again, more instructions can be put after these loops.

Arrangement...

A typical 10-liner will be arranged in the following order

Lines 1-2: Initialisation section, avoid using **READ** and **DATA** as the **DATA** statement always ends the line.

Lines 3-9: Main game loop. If joystick control is used, boolean formula can be used instead of If thens. e.g

```
30 If Jleft then X=X-1
40 If Jright then X=X+1
```

changes to

```
30 X=X+Jleft-Jright
```

Line 10 is usually used for your GAME OVER sequence and getting ready to play again. Once you have worked out where the sections of your program are going to go, you are halfway towards cramming your program into 10 lines !

❑ Practical Lesson

O.k, we are now going to write a simple 10-liner and disect each line !

The game will be a very simple variation of space invaders. You will have only 1 alien, but he will tend to dodge randomly as it comes down.

SPACED INVADER !

First of all, initialisation on line 10

```
10 mode 0 : key off : hide on :
 curs off : A=hunt(start(1) to st
 art(1)+length(1),"PALT")+4 : for
  B=0 to 15 : colour B,deek(A+B
 *2) : next B
```

The Mode 0 : key off : hide on : curs off is a standard sequence to go to low rez, get rid of the function key menu, mouse cursor and text cursor (curs off also stops the flashing colour). The rest of the line is a standard routine to get the colour palette from the sprite bank. The GET SPRITE PALETTE command in the usefull extension coming with issue 13 will replace all of this part !

```
20 EX=16 : EY=8 : EDX=1 : LIVES
 =3 : while LIVES>0 : gosub 100
 : X=160 : BF=0 : EBF=0 : sprite
  off : DEAD=0 : while DEAD=0
```

Next we set up the start values for the variables used. EX & EY are the co-ordinates for the enemy, with EDX being the movement vector (changes between 1 and -1). LIVES stores the number of lives left. We then set the outer loop going with the while LIVES>0. The loop structure, by the way, is going to go like this:

```
While LIVES>0
   re-initialise player
   While DEAD=0
      play game
   Wend (while DEAD=0)
   lose a life
Wend (while LIVES>0)
Lose game
```

the Gosub 100 updates the score (starts at 0 and is stored in the variable SCRE), then the player's gun position (stored in X) is set to 160 (centre) and the flags that show a bullet fired (BF) and an enemy bullet fired (EBF) are set to 0, along with the DEAD flag (when set to 1, you lose a ship). Then the sprite off command removes all the sprites from the screen and the main game loop starts, with While DEAD=0

```
30 sprite 1,X,190,1 : sprite 2,EX,E
 Y,3 : sprite 3,999-BF*999+BX,
 BY,2 : sprite 4,999-EBF*999+
```

Learners

How to write 10-liners with Professor Spock

"What is all the fuss about 10 line programs ?" you might ask. Well, there are several reasons why 10-liners are popular and generally good examples of programming.

1) They are short and fit nicely into a newsletter.

2) They are exellent examples of Spagetti Code.

3) They encourage compact and efficient programming.

Basically, unless you can program efficiently, utilising all the STOS language, you will find it difficult to program a 10-line game. So, I will be using the 10-liner as the basis for the Learners series for a couple of issues.

Basics If you've read my Spagetti Code article in issue 8, you will realise that commands like **rem** (aargh !), **goto**, **If..Then** and **Return** will effectively end a programming line. When you are restricted to 10 lines this is obviously going to be a major concern, so you must work out ways of getting around this problem.

❑ Tricks

The usual command that causes problems is **If..then**, as it is so often used in a game. However, any **If..thens** that produce purely mathematical results e.g

```
20 If A=B then C=C+1
```

can be turned into a **boolean** formula e.g

```
20 C=C-(A=B)
```

as the = <,> symbols are also mathematically used. When the line

```
20 If A=B then C=C+1
```

is processed, STOS first looks at the equation **A=B**. If A=B the result is -1, otherwise the result is 0 (This is a boolean function), so by subtracting the result from C (a negative times a negative is a positive) we end up with the same result, and the line can be continued afterwards. More complex lines can be used, for example, this line out of Skystrike:

```
60 SP#=SP#+TURBO*2+TH*
0.1+DY(R,0)*0.15-0.1-((16-
Y*0.1)+AL*16)*0.01-UC*0.03-
max(0,FRE)/4+(AL=0 and Y>=154
and SP#<4)*0.4+STEAM :
SP#=(0,min(MXSP+TH/
3+max(0,DY(R,0)/3)+TURBO*2-
UC,SP#)) : If SP#<0.3 and
LD=0 and CL=0 then ST=16
```

This line handles about half of the speed formula for player's aircraft in the game. For those of you crazy enough to want to work out what's going on. **SP#** is the speed of the aircraft, **TURBO** is the turbo boost flag (0=off, 1=on), **TH** is the throttle value, **DY()** contains all the Y vectors, **R** is the rotation of the aircraft, **AL** is the altitude in screens , **Y** is the Y co-ordinate on the current screen, **UC** is the undercarriage flag, **FRE** is the Fire flag (1+= aircraft on fire), **STEAM** is a flag set when the aircraft is fired from the steam catapult on the aircraft carrier, **MXSP** is the maximum airspeed, **LD** is the landed flag, **CL** is the crash landed flag, and **ST** is the Stall variable. Notice also that I have avoided a great number of divide instructions by multiplying by fractions (i.e *0.25 instead of /4) which works out slightly faster. If this line was done as a traditional textbook formula, with one calculation per line, it would take up somewhere in the region of 50 lines !

Another trick is using **For..Next** and

Goodman Enterprises &
The Official STOS Library
Present The

STOS

Programmers Competition

The competition is split into two catagories, one for games, and the other for applications, with THREE main prizes for EACH section.

1st Prize

A copy of the forthcoming STOS 3D donated by Mandarin Software Plus £50 cash.

2nd Prize

A copy of STOS Paintmaster donated by Stallion Software plus a copy of 'Introducing Atari ST Machine Code', the great new book and disk set from Roger Pearson.

3rd Prize

A copy of TOME donated by Shadow Software

All programs submitted stand a chance of being accepted to the exclusive special licensed title selection from Goodmans' and the STOS Library, so you may even earn extra pocket money, without even winning.

For your entry form send a S.A.E. to either Goodman Enterprises or the STOS PD Library at the usual address

```
EBX,EBY,2 : wait vbl
```

This line (being the first in the main loop) handles all the sprites. Sprite 1 is used for the player's ship (fixed at Y=190) using image 1. Sprite 2 is the alien ship (image 3), and sprites 3 and 4 are for the player's bullet and enemy bullet. Note the way the 999-BF*999+BX used as an X co-ordinate for the bullet means that when BF is zero, the bullet will not be plotted.

A wait Vbl is done at the end of the line to synchronise things.

```
40 BY=BY-BF*4 : BF=-BF*(BY>=-
8) : EBF=-EBF*(EBY<200) :
EBY=EBY+EBF*4 : EC=btst(1,col
lide(4,8,8)) : C=btst(2,collide(3,
8,8)) : if C then SCRE=SCRE+10
0 : gosub 100 : EX=16 : EY=
min(160,8+SCRE/25) : boom
```

This line handles bullet movement (BY=BY-BF*4), checks to make sure that the bullet isn't off the top of the screen (BF=BF*(BY>=-8)) and does the same for the enemy bullet (but checking for off the bottom of the screen), then it does a collision detection between the player's bullet and the alien ship, and between the enemy bullet and the player's ship (the result is stored in EC for checking on the next line). If they have collided, the score is upped by 100 and the score update routine on line 100 is called, then the alien ship is reset to the start (although it will be slightly nearer the ground).

```
50 if EC then dec LIVES : boom
: DEAD=1
```

If the collision between the enemy bullet and the player's ship was true, then the player loses a life and the DEAD flag will be set.

```
60 EX=EX+EDX : if EX<16 or EX>
304 or rnd(100)>98 then EDX=-
EDX : EY=EY+8 : if EY>=190
then DEAD=1 : LIVES=0
```

This line moves the alien ship, by adding EDX (the vector) to the alien's X co-ordinate (EX), if EX is out of it's normal area, or if a random from 0-100 is greater than 98, then the alien's direction is reversed (EDX=-EDX) and the alien goes 8 pixels nearer the ground. If the alien's Y co-ordinate is greater than or equal to 190, then it has landed and the player loses the game.

```
70 X=max(8,min(312,X+jleft*2-
jright*2)) : if fire and LFR=0
and (BF=0 or BY<EY) then BY=1
80 : BX=X : BF=1 : shoot
```

The first part of this line handles the player's movement. Then the if fire and LFR=0 statement checks to see if the player has fired the gun. (LFR is set to 1 if the fire button was pressed last frame, see line 80). The gun cannot be fired if there is a bullet already in motion, thus the and (BF=0 or BY<EY) but it can be fired if the bullet is already higher than the alien.

If fired, the bullet's co-ordinates are reset to the position of the gun, and the bullet's flag (BF) is set, to show that it is in motion.

```
80 LFR=fire : if EBF=0 and rnd(1
00)>80 then EBF=1 : EBY=EY :
EBX=EX : shoot
```

After checking to see if the fire button is held down, this line checks to see if the alien wants to fire. It follows the same rules as the player's gun, except that the fire button is replaced by a random value.

```
90 wend : wait vbl : wend : gos
ub 100 : locate 0,10 : centre "
GAME OVER" : wait key : end
```

This line finishes off the two loops,

updates the score for one last time and ends the game.

```
100 locate 0,24 : print "Score :"
";SCRE;" lives="";LIVES; : return
```

Line 100 is a very simple sub-routine to update the score.

All you need to do now is create 3 sprites..

Sprite 1..Player's gun (16x16)

Sprite 2..Bullet (8x8) remember, it goes both directions !

Sprite 3..Alien (16x16)

Once all this is done, you have another 10-liner, and hopefully some good clues as to how to write your own. Remember, we still want good 10-liners for the compilation disk !

MONEY !!!

Yes, we are actually paying for articles and programs for the newsletter. If your article or program is published in the newsletter, you will be paid £5 per printed page !

Currently, Richard Gale has a check for several pages worth of work on its way to him. Although he doesn't do it for the money, only to annoy me by writing neatly structured code that takes several pages to list. Please note that all lines like:

```
10 Rem ******************
****** Just for the sake of
it ******************
```

will be shortened to:

```
10 Rem !
```

Remember, this is a **USER** club. That means that **USER**s are supposed to be writing programs for it !

Upgrade now !

If you are producing PRG versions of your games or demos, it is essential that you upgrade to the latest version of STOS, otherwise they won't work on the newest Ataris. Version 2.6 works with TOS 1.62 and the Atari STE which is now on sale.

Simply send £2 or £1 plus a disk to Sandra Sharkey at the STOS Public Domain Library.

P.D Upgraded

The STOS Public Domain library will no longer accept any PRG versions of programs that have not been compiled with version 2.6 of STOS (i.e any that won't work on the STE). Most of the current P.D library has now been re-compiled to run on the STE, and the resubscription goodies disk has an extension specially for the STE !

Another thing that will be happening to the STOS P.D library, is that ALL the disks will be converted to double sided format with twice as much on each disk (as 90% of ST's are now double sided). Don't fret if you only have a single sided drive, just mention this when you order your disks from Sandra, and she will split the programs over 2 single sided disks wherever possible. However, this will cost you for the extra disk.

Each disk in the current P.D library is been scrutinised, and any that cannot be upgraded to V2.6 or are not much use anymore (like the origional STOS demos) will not re-appear in the new library.

At the time of publishing, there is not yet a full list of what is in the P.D library, so please send an S.A.E to Sandra for the catalogue when it becomes available .

SCROLLING

SCROLLING TEXT....

If you can remember as far back as issue 8, you'll remember that I said I would do a scrolling text demo with scrolling background for issue 10. And if you remember issue 10, I did nothing of the sort!

O.k, sorry, I forgot ! The following program is the new scrolling text demo. It isn't the only way to do large scrolly text over a background, and it certainly isn't the best, but it is a good one to show the basic principles.

The general form of the program is the same as the small text scroller we did in issue 8, in fact it is the same program, with a few modifications.

There are various ways of doing large text for a scroller (or for anything that requires large text). The usual ones however, are:

1) Create sprites for each character (heavy on memory)

2) ZOOM small characters up to large (slow)

3) Store large text on a screen and screen copy it (heavyish on memory)

4) Create scroller as a map using TOME (characters created from tiles can be as large as you want)

Because you won't all have TOME until issue 13 (Working on it for February) and creating sprites for each letter in the alphabet is a slightly large task, we will do methods 2 & 3.

☐ ZOOMing....

The general trick with zooming text, is to print the text on a spare area of screen, and then zoom it to the area that is being scrolled. In the small text scroller, this area was only 8x8 pixels, so obviously, we must modify the scrolling area to compensate.

In the following program, we will zoom 8x8 pixel characters up to 16x32 (in fact the zoom is from 16x8 to 32x32 because of zoom's 16 pixel X-boundary limit). Lines xx-xx handle the background screen (scrolling away on screen 8). This uses an unused area of screen 5 to work the wraparound.

Instead of using screen copy to copy the scroll line over the background, I have used the **SCREEN$()** function. Thus colour 0 will be transparent. If you want, replace the **SCREEN$()** copying to **screen copy**, and the text will always be in a colour 0 band.

The extra bits at the start of the program load in the background picture, and lines 200-240 have been modified to compensate for the change in the text size.

```
10 mode 0 : key off : curs off
15 F$=file select$("*.PI1","Load
   a background Picture",1) : if F$
   ="" then end
16 reserve as screen 6 : load
   F$,6
17 hide on
20 reserve as screen 5 : auto
   back off : screen copy back
   to 5
25 T=0
26 P=0
27 MES$="Hello this is a
   scrolling message ....."
28 get palette (6)
30 logic=back
40 screen copy 5,2,80,320,112
```

```
to 5,0,80 : rem scroll text
line
45 screen copy 6,0,0,320,2 to 5,
0,198 : rem copy top edge
of background
46 screen copy 6,0,2,320,200 to
6,0,0 : screen copy 5,0,198,
320,200 to 6,0,198
47 rem the above does the
scroll for the background
50 screen copy 6 to back :
screen$(back,0,76+Y) = screen$
(5,0,80 to 312,112)
51 inc Q : Q=Q mod 360 : Y=20*
sin(Q*pi/90.0)
55 screen swap : wait vbl
60 inc T : T=T mod 4 : rem
modifying the mod 4, to mod
anything else produces
interesting results !
70 if T=0 then gosub 200
100 goto 30
200 logic=5 : back=5
210 A$= mid$( MES$, P+1,1)
220 locate 0,0 : print A$;
221 zoom 5,0,0,16,8 to
5,288,80,319,112
225 back=default back
230 inc P : P=P mod len(MES$)
240 return
```

The following modifications should be done, if you want to use the slightly faster method of having all the characters stored on a screen, and then screen copying them to the scroll line. This variation gives you 60 characters (so lower case is automatically moved to upper case with the **UPPER$()** function).

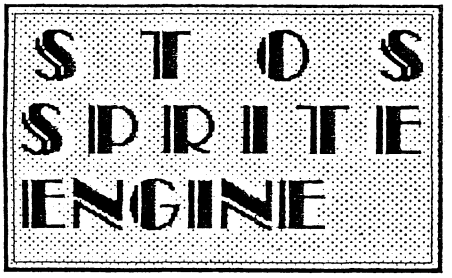
```
22 reserve as screen 7 :
gosub 300
200 rem modified large text
printer
210 A$=upper$(mid$ ( MES$,
P+1,1) )
220 A=max(0,min(57,asc(A$)-32)
) : XX=A mod 10 : YY=A/10 :
rem the character is
converted to an ascii value,
offset, then the co-ordinates
of the relevant screen block
are calculated.
221 screen copy 7,XX*32,YY*32,X
X*32+32,YY*32+32 to 5,288,80
225 rem this line is no longer
required
230 inc P : P=P mod len(MES$)
240 return
```

The following lines create all the text for you by zooming the text onto a spare screen (screen 7). Note that the right half of each character block is blank (they are 16x32 characters on 32x32 pixel blocks).

```
300 rem convert small (8x8)
text to large (32x32) text
305 logic=5 : back=5 : ink 1
310 for A=0 to 58 : X=A mod 10
: Y=A/10
315 locate 0,0 : print chr$(A+32)
: zoom 5,0,0,16,8 to 7,X*32,Y*
32,X*32+31,Y*32+31
320 next A : back=default back
325 return
```

Right, get on with it and have fun !

SPRITE ENGINE



Normally in STOS, you can only use 15 sprites on screen at the same time. However, if you are using screen swapping, with a constantly updating background, you can use a simple trick to give you as many sprites on screen as you want. Obviously, the more sprites you have on screen, the slower your program will run, but it is faster than the **PUT SPRITE** method and was used to great effect in the game **YOMO** on the Games Galore compilation (Up to 53 sprites on screen at once !)

First of all, you have to make sure that your program uses the correct method of screen display. The normal sequence of events in a screen swapping display go thus:

set LOGIC equal to BACK

copy from background screen (Stored in a bank) to BACK

display sprites..ALL sprite display (including mouse) should go here, before the UPDATE command.

UPDATE

SCREEN SWAP

Then would come your control routines (joystick/keyboard etc) and anything else in your main loop.. The above sequence should be the first part of your main loop.

To display a sprite, you could simply use the normal **SPRITE N,X,Y,I** but of course this would limit you to 15 sprites. This is where the sprite engine comes in...

The sprite engine routine is a 2 line subroutine that handles sprite placement for you. It will only work with a screen swapped display. You simply set the variables SX,SY and S to the X,Y co-ordinates of the sprite and the image number, then Gosub to the routine (I normally have it a lines 150-155) and let it handle the sprites. You also have to put the command NS=0 at the start of your main loop.

SPRITE ENGINE ROUTINE (adjust line numbers to suit your program)

```
150 sprite NS,SX,SY,S : inc
    NS : if NS=16 then NS=1 :
    Update
155 return
```

Simple isn't it ! By the way, I normally have my program organised along the following lines.

Lines up to line 39, Initialisation

Lines 40-50 screen display. Sprite display is done by a gosub to line 100

50-60 Maths for player and other sprite movement (don't use MOVEX,MOVEY)

60-70 controls

80-99 anything else required in main loop

100-149 sprite display

150-155 sprite engine

200-whatever subroutines for everything else..

TOME V2.1

TOTAL MAP EDITOR V2.1

Do you want to write profesional map based games, like Gauntlet, New Zealand Story, Rainbow Islands etc. ?

Did you know that it's possible to do so using STOS ! The TOME (TOtal Map Editor) system gives you 18 extra commands for STOS that enable you to write high speed, pixel accurate scrolling routines just like those used in all the major games ! Over 90% of games written on the ST are map based for their backgrounds or play area, and TOME gives you the power to use maps in your games, saving you huge amounts of memory and giving you a major speed advantage.

The brand new version of the TOtal Map Editor has now been released, including an even more powerful editor, extra commands and two new demo games, including sourcecode and data files. Also included are example programs and a scrolling demo written using TOME.

TOME's features are:

- Single Pixel Scrolling in all directions (Includes simple STOS example to show you how it's done !)
- Over 240 screens can be stored in 64K of memory !
- Block Animation within your maps (So you can do HUGE animations !)
- The MOST POWERFULL MAP EDITOR AVAILABLE on the ST !
- Full 28 Page manual included, and example files on disk, not to mention the two demo games CUTE CUDDLY PURPLE BABY DRAGON GOES FLOWER ARRANGING, and JITTERBUGS][.

"Both Games Galore and TOME deserve to become an integral part of any STOS Programmer's library." Atari ST user March 1990

"If you don't buy it (TOME) you must be raving mad !" Peter Hickman STOS Newsletter Issue 10

Only £14.95 to STOS Club Members (£18.95 to Non Members) Add £1 for P&P if ordering from overseas.

TOME was written by Aaron Fothergill. Send U.K cheque, Postal Order or International Money Order to:

Shadow Software, Barnstaple, N.Devon.

S T A C to STOS - Character Set Converter

By Richard Gale

After using STAC for a while before changing over to STOS I had laboriously created a number of fonts in STAC format, the thought of having to redraw them for STOS did not seem like a 'great' experience. Instead I spent half and hour at the keyboard and came up with the following program, which automatically converts STAC character sets to STOS format.

When the program is first run you will be presented with a menu :-

A) 8 by 8

B) 16 by 8

C) 8 by 16

D) 16 by 16

These selections are the size of the STOS character set, in pixels, after the STAC font has been converted. You will then be asked for the STAC character set, once you have selected one it is loaded into memory. Once the STAC font has been loaded you will be asked for the name of the STOS character set, type in the required filename, if a filename is not selected or QUIT is selected the conversion will still take place but the program will not save the character set, (the STOS character set is stored in bank 5).

When the required files have been selected the conversion will take place, and a display will show the current character being converted. When the conversion is complete the STOS set will be saved to disk (if selected), otherwise the character set is left in bank 5.

STAC Character set format

The STAC character set is simply stored as 16*256 bytes, 0 to 255 characters with each character 16 bytes high.

STOS Character set format
Offset Size Value Designation
0 .L $06071963 I.D.code (Francois' birth date)
4 .W 1 or 2 X size of characters (in bytes)
6 .W 8 or 16 Y size of characters (in scanlines)
40 to 263 1 to 233 An index to characters (?)
254 to end 0-255 Character data

For characters 8 pixels wide the character is stored as one byte per scanline, for characters 16 pixels wide the character is stored as one word per scanline.

Notes on conversion

The conversion routine locates the start of the STOS and STAC character set, and copies the first 233 characters from the STAC set to the STOS set. Because some STOS sets can be only 8 scanlines high scaling is needed to fit the STAC character set, this is done by storing only odd or even scanlines of the STAC set.

The Program

Boot up your STOS Language disk and type in the following program. All the comments on the program are listed later.

```
100 rem ~ STAC to STOS chr
    set converter ~
110 cls : key off
```

```
120 gosub 450 : gosub 390 :
 gosub 180
130 If DEST$="" then end
140 locate 0,14 : centre
 "Saving "+DEST$
150 save DEST$,5
160 end
170 rem ~ Convert STAC to
 STOS chr set ~
180 MEM=start(6)+12 : SET=
 start(5)+264
190 for CHAR=32 to 255 :
 locate 0,12 : centre
 "Converting "+chr$(CHAR)
200 for A=0 to YSIZE-1
210 BYTE=peek( MEM + CHAR
 *16+A*16/YSIZE)
 : on XSIZE gosub 270,280
220 next A : next CHAR
230 erase 6 : reserve as data 6,
 1 : erase 6
240 locate 0,12 : centre
 "Conversion complete"
250 return
260 rem ~ XSIZE = 8 ~
270 poke SET,BYTE : inc SET :
 return
280 rem ~ XSIZE = 16 ~
290 WRD=0 : If btst(7,BYTE)
 then bset 15,WRD : bset 14,WRD
300 If btst(6,BYTE) then bset 13,
 WRD : bset 12,WRD
310 If btst(5,BYTE) then bset 11,
 WRD : bset 10,WRD
320 If btst(4,BYTE) then bset 9,
 WRD : bset 8,WRD
330 If btst(3,BYTE) then bset 7,
 WRD : bset 6,WRD
340 If btst(2,BYTE) then bset 5,
 WRD : bset 4,WRD
350 If btst(1,BYTE) then bset 3,
 WRD : bset 2,WRD
360 If btst(0,BYTE) then bset 1,
 WRD : bset 0,WRD
370 doke SET,WRD : SET=SET+2
  : return
380 rem ~ Create bank ~
390 SIZE=264+224*(XSIZE*YSIZE)
  : erase 5 : reserve as
 set 5,SIZE
400 SET=start(5) : fill
 SET to SET+SIZE,0
410 for A=0 to 223 : poke SET+
 40+A,A : next A
420 loke SET,$6071963 : doke
 SET+4,XSIZE : doke SET+6,
 YSIZE
430 return
440 rem ~ Title Page ~
450 locate 0,0 : centre "STAC
 to STOS Character Converter"
460 locate 0,1 : centre string$
 ("-",32)
470 locate 0,3 : centre "written
  by Richard Gale 6/1/90"
480 locate 0,5 : centre "A) 8 b
 y 8"
490 locate 0,6 : centre "B) 16 b
 y 8"
500 locate 0,7 : centre "C) 8 b
 y 16"
510 locate 0,8 : centre "D) 16 b
```

PROGRAM

```
 y 16"
520 locate 0,10 : centre "Select
  character size :"
530 repeat : KY$=upper$(inkey$)
 : until KY$>="A" and KY$<="D"
540 XSIZE=1 : YSIZE=8
550 if KY$="B" or KY$="D"
 then XSIZE=2
560 if KY$="C" or KY$="D"
 then YSIZE=16
570 STAC$=file select$("*.FNT",
 " Select STAC Font to
 convert")
580 if STAC$="" then default :
 end
590 erase 6 : reserve as data 6,
 5000 : bload STAC$,6
600 DEST$=file select$("*.MBK",
 " Select STOS Font to save")
610 return
```

Comments

110 clear the current window and removes the function keys

120 this line calls subroutines to execute the title page, initialise the STOS character set, and convert the STAC character set.

130 if there is no STOS destination file the program is terminated after the conversion and the STOS character set is left in bank 5. (The destination file is stored in DEST$ which is set on the title page, if this is set to null ("") then the END command terminates the program.)

140 this displays a helpful message to the user.

150 SAVE DEST$,5 saves bank number 5 (the converted character set) to disk as the name in DEST$.

160 the program is terminated.

Conversion routine

180 sets variables MEM and SET to the address in memory of the STAC and STOS characters sets respectively. (The character data for STAC is offset from the start by 12 bytes, and the character data for STOS is offset from the start by 264 bytes.)

190 starts a FOR..TO.. loop from 32 to 255, and displays a useful message. (The FOR..TO.. command is used to cycle through all the characters 32 (space) through to 255 (a border), 200 starts another FOR..TO.. loop from 0 to YSIZE-1 (The FOR..TO.. command is used to cycle through each byte which makes up each character. The number of bytes is stored in YSIZE and is calculated on the title page.)

210 calculates each byte of each STAC character and stores as STOS character data. (MEM+CHAR*16+A*16/YSIZE is the address in bank 6 of byte A in character CHAR. Each character is stored as 16 bytes, and because some STOS characters are only 8 bytes high A*16/YSIZE is used to reduce characters where necessary.)

220 end FOR.. TO.. loop for character bytes, and end FOR .. TO loop for each character. (The NEXT I will increment I and loop back to the FOR .. TO command until I is equal to YSIZE-1.)

230 erases STAC data, and creates then deletes a null bank.

(ERASE removes a STOS bank, and RESERVE creates a bank. A null bank is created and deleted to make sure that the new character set is accepted, without this line STOS occasionally forgets to acknowledge the character set.)

240 displays a useful message to the user.

250 end subroutine.

270 store BYTE at address SET, and increment SET. (this stores the current BYTE for the current character at the address SET in bank 5, and moves onto the next byte in bank 5.)

290-360 clears WRD, and checks each bit in BYTE and stretches it into two bits in WRD.

(this routine checks through the bits 7 to 0 in BYTE and replaces each BIT with two BITS in WRD. If bit 7 is set in BYTE then bit 15 and bit 14 are set in WRD. This routine is used to double the horizontal size of each character.)

370 store WRD at address SET, and add 2 to SET. (this stores WRD as a word at address SET (which must be even), and the increases SET by 2 (a word is equal to two bytes).)

390 calculates the size of the character set and reserves bank 5 as a character set. (the size of a character set is the size per character (xsize*ysize) times the number of characters (224, 255 characters minus 31 control characters at the beginning), plus 264 bytes (used as a header).

400 calculates SET as the start address of bank 5, and clears all data in bank 5 (the START command is used to find the start address of a STOS bank, and the FILL command fills a range of memory with a single word. The FONTS accessory defines a blank character set with full blocks where the window border characters are.)

410 this sets up a table of values from 0 to 223 offset from the start of bank 5 by 40 bytes. (the FOR .. TO cycles from 0 to 233 filling memory in bank 5 from 40 to 263.)

420 sets the STOS character set identifier $6071963, and sets the character size. (the id $6071963 is used to that STOS recognised the bank as a character set (without this identifier the character set cannot be used), and words offset from the start by 4 and 6 define the character size.)

430 end subroutine.

450-520 show some useful messages.

530 wait until A,B,C or D is pressed.

540 set default character sizes

550 if B or D pressed then the XSIZE=2 (if either 16 by 8 or 16 by 16 is selected then XSIZE is set to 2 (2 blocks of 8 bit = 16 bits)

560 if C or D pressed then the YSIZE=16 (if either 8 by 16 or 16 by 16 is selected then YSIZE is set to 16 (16 bytes per character)

570 select a file and store the name in STAC$ (the command file select$ shows a STOS file selector, with all *.FNT files showing.

580 if QUIT selected then terminate program (if QUIT is selected on the file selector then STAC$ is null (="") and the program is terminated by the END command)

590 erases bank 6, and recreates bank 6, and then bload's the STAC character set in it. (The RESERVE and ERASE command create and remove areas for STOS memory banks, the bload"" command is used as a STAC character set file is not a recognised file , bank 6 is erases in case a STAC character set is already in memory.)

600 select a file and store the name in DEST$ (the command file select$ shows all the *.MBK files in a STOS file selector)

610 end subroutine .

NEWS

(Editorial Continued from Page 2)

STOS Manual, the newsletters and any other STOS extension manuals (like the Compiler, Maestro and TOME). Due to the varied nature and size of these manuals, the result could be quite interesting ! If anyone comes up with any brilliant ideas, please send them in !

❑ No STOS Plus !

Due to the amount of time it would take Francois to re-program STOS into STOS Plus, Mandarin are not planning to release an upgraded version of STOS. They do, however, plan to keep on supporting STOS and releasing new products (such as STOS 3D, which should be released at the same time as AMOS 3D, planned for March-June 1991 and STOS VIDI, which is being developed by Peter Hickman).

Don't worry, however, about STOS becoming out of date though. As Francois is still updating STOS to handle the newer ST's as they are released, and there are plenty of new commands being written as extensions (nearly 100 of them are on the re-subscription goodies disk !). So far, just about all the things the ST/E can do, can be done with STOS commands (and some of the things that the ST isn't supposed to be able to do !), and there are enough machine coders working with STOS to keep it updated well into the mid 90's.

Be warned about some slightly dodgy advertising in some of the ST magazines. Several software shops have been advertising STOS products which haven't even been released or planned for release yet. This is due to the advance time the magazines need the adverts by. The shop hears a rumour about a new program, and then advertises it, hoping that it will be released when the magazine goes to press some 2 months later. They do get it right about 30% of the time, but STOS VIDI,STOS MUSICIAN and STOS 3D are not out yet ! STOS 3D should be released between March and June, STOS VIDI is just about finished, and STOS MUSICIAN should be finished in June (around 18 months later than planned). The easiest way to check whether some STOS software is released before you send off your money to the shop, is to check with us first !

❑ STOP PRESS !

The STOS P.D game "Plumb Crazy" by Keith Feeney has been withdrawn from P.D on the request of Empire Software, who distribute the game "Pipemania", as they feel that it is too close to the original. It's nice to know that STOS is now considered good enough, that profesional software companies can be worried that a game written in STOS that is similar to their own, is of enough quality to take away sales of their own game.

This obviously makes the point, that if you are writing a game in STOS, you should come up with your own ideas rather than cloning an already existing game. As STOS written clones can be a little too close to the original !

❑ STOS Programmers Competition

Goodman Enterprises and the Official STOS Library (i.e Sandra) have come up with a programming competition for all STOS programmers. The challenge is to write the best P.D disks of either games or utilities, with some great prizes for the winners in both categories. Full details of the prizes are on page 17, and to enter, please send a stamped, self addressed envelope to either the STOS P.D Library, or Goodman Enterprises for an entry form. The winners will be anounced in June 1991, and the winning disks will be available through both the STOS P.D Library and Goodmans PDL.

SNEAKY M-CODE

Being

Sneaky

How to protect your STOS programs from listing...

Wouldn't it be nice to stop peopl e from listing your programs and stealing your code ? Also, wouldn't it be good to be able to assign particular character codes to do special jobs.

The Sneaky Mk1 extension resets the Trap 3 vector to its own routine, so that it can intercept command 1 (**PRINT STRING**). If this command is called, then the string is checked for a special control character (the ~ **Squiggle**). If there is no control character, or if it is not command 1, then control is passed to the normal trap 3 routines.

If the squiggle is detected, then the next character in the string is checked. The following pair are implemented in this version, but you could add your own if you want !

N - No list mode. All strings sent through the trap are changed to **"PROTECTED PROGRAM !!!"** so that anything printed (including listing to the screen) will be scrambled. Unfortunately this doesn't work for the printer.

L - List normally. Returns the routine back to normal.

If you want to add your own character codes, then add the checks in the **SUPRISE** routine. Remember that you must save ALL the registers, and return via a **bra** command to **NOCHANGE**.

Two commands are added to STOS by this extension:-

SAFE - Switches the trap back to normal (~**N** wont work anymore)

UNSAFE - Switches the trap back on.

The trap is in active mode when you boot STOS (as it is set up in the extension's cold start routine). It is a good idea to change the **SAFE** command token to a codeword of your own (so that only you can switch of the No List mode), but of course, make sure that you don't use any reserved words.

```
* Sneaky Mk1 extension
  (assemble as SQUIDGE.EXT)
* this is only required as an inte
  rpreter extension
* trapping the output traps in S
  TOS
* to protect programs from listi
  ng
* By Aaron Fothergill, STOS Cl
  ub 1991 *
          bra     INIT
          dc.b    128
TOKENS:   dc.b    "unsafe",128
          dc.b    "safe",130 ;
  change this to your own
  code
          dc.b    0
JUMP:     dc.w    3

          dc.l    UNSAFE
```

```
        dc.l    DMY
        dc.l    SAFE
MESSAGE:
        dc.b    10," ",0 ; we d
 on't want anyone to know
        dc.b    10," ",0 ; what
  this extension is !
SYSTEM: dc.l    0
RETURN: dc.l    0
INIT:   lea     EXIT,a0
        lea     COLDST,a1
        rts
COLDST:
        move.l  a0,SYSTEM
* First re-vector
 the trap (trap 3)
        lea     NCJ+2(pc),a0 ;
 get address of jmp
        move.l  $8c,a1 ; get the
  old trap
        lea     MYTRAP(pc),a2
 ; get address of ours
        move.l  a2,$8c ; stuff !
 t into trap 3
        move.l  a1,(a0) ; save t
 he old trap 3
* initialisation over
        lea     MESSAGE,a0
        lea     WARM,a1
        lea     TOKENS,a2
        lea     JUMP,a3
WARM:   rts
SAFE:
        move.l  (a7)+,RETURN
        movem.l a0-a6,-(a7)
* Toggle the sneaky mode off
        lea     MYTRAP(pc),a0
 ; get address of jmp
        move.l  $8c,a1
        cmp.l   a0,a1
        bne     SAFEEND   ;
        lea     NCJ+2(pc),a0
        move.l  (a0),a2 ; get th
 e old trap
        move.l  a2,$8c ; stuff !
 t into trap 3
SAFEEND:
        movem.l (a7)+,a0-a6
        move.l  RETURN,a0
        jmp (a0)
UNSAFE:
        move.l  (a7)+,RETURN
        movem.l a0-a6,-(a7)
* Toggle sneaky mode on
        lea     MYTRAP(pc),a2
        move.l  $8c,a1
        cmp.l   a2,a1
        beq     SAFEEND   ;
        lea     NCJ+2(pc),a0
        move.l  a1,(a0)
        move.l  a2,$8c
        bra     SAFEEND   ;
DMY:    rts
MYTRAP: cmpi.w  #1,d7 ; is it the
  PRINT STRING trap
        bne     NCJ ; nope !
        move.w  #0,d7
LOOP:   tst.b   (a0,d7)
        beq     NDLOOP
        cmpi.b  #"~",(a0,d7) ; i
 s it the Special character ?
        beq     SUPRISE
```

```
        addq.w #1,d7
        bra LOOP
NDLOOP: move.w #0,d7
        cmp.w NOLISTON(pc),d
7 ; is no list mode on ?
        bne     KILLCHAR ; yes
, kill the character
NOCHANGE:
        move.l #1,d7
NCJ:    jmp     $12345678 ; nor
mal trap address gets put
here
KILLCHAR:
        lea MYSTRING(pc),a0
        bra     NOCHANGE
SUPRISE: addq.w #1,d7
        cmpi.b #"N",(a0,d7)
        beq     DONTLIST
        cmpi.b #"L",(a0,d7)
        beq     LISTON
        bra     NOCHANGE
DONTLIST:
        move.l a0,d7
        lea     MYDATA(pc),a0
        move.w #1,NOLISTON-
MYDATA(a0)
        lea MYSTRING(pc),a0
        bra     NOCHANGE
LISTON:
        move.l a0,d7
        lea MYDATA(pc),a0
        move.w #0,NOLISTON-
MYDATA(a0)
        lea MYSTRING(pc),a0
        bra     NOCHANGE
MYDATA:
NOLISTON: dc.w 0 ; flag for NO
 list mode
MYSTRING: dc.b "PROTECTED P
 ROGRAM !!!!!",0
EXIT:
```

The basic idea behind these routines are that you could quite easily give yourself some new STOS commands, without having to write them as an extension. These commands could also be embeded in text strings, so that whenever they are printed, they are activated. For example Self Justifying text strings that automatically locate themselves, or if you tie it in with an interupt routine, animating text (just print it, and watch it animate itself) !

Page 6 New Atari User

Page 6 magazine (also known as New Atari User) regularly run a STOS feature in their ST section.

Pete Hickman, the journo hack who writes their column has reviewed (and still does review) STOS products, P.D and just about always has some sort of program listing in his column. Well worth a read !

The Games Maker's Manual

The Games Maker's Manual by Stephen Hill is no longer available through the STOS Club as we are now out of stock and no further orders can be taken. The Listings disk is now available through the STOS P.D Library. Fortunately most bookshops either have the book in stock, or can order it in a short time.

NEWSLETTER INDEX

Here's a handy Index to all the articles and usefull programs from Issues 1-12, listed by Newsletter and page number, with some usefull comments.

□ PROGRAM LISTINGS

Name	Issue	Page	Comments
STOSCOPY	1	6	Working version of the STOSCOPY program
RAINBOW REVELATIONS	1	8	Colour shift demo
SPIRAL	2	5	Spiral graphics demo
4 WAY SCROLLING	2	9	Simple 4 way scrolling routine
CHARACTER GENERATOR	2	9	Role playing game character generator
GRABBER	2	11	Grabs graphics and stores them to disk
INVISIBLE SPACE INVADERS	4	9	How to con your local Amiga owner !
KIDDIES KEYBOARD DESTROYER	3	14	Educational game
WHAT DAY	3	15	Calculates what day it is for any date
WHEELS	3	16	Wheel drawing program
TRAPS FROM BASIC	4	13	Accessing traps via STOS
DISC SECTOR SALVAGE	4	14	Recovers text files from some corrupted disks
SPRITE MERGER	4	15	Allows sprites to be stuck together to make larger ones.
ALERT BOX	4	16	Simple Alert Box routine
TEXT PARSER	4	18	Adventure game text parser
SPRITE+JOYSTICK	5/6	3	Controlling a sprite from the Joystick
PCP	5/6	10	Pre-editor program (so that you can use procedures and labels)
MIDI COMS	5/6	15	Communicates between 2 ST's via MIDI
SCROLLING TEXT	5/6	16	Simple PCP program
BLOCKER	5/6	17	10-liner game
RACE WAY & SCREEN DESIGNER	5/6	18	Simple car game
MIDI MONITOR	5/6	27	MIDI Monitor program
MIDI SYS EX DUMPER	5/6	28	Sound storage program for MIDI synths
PICTURE SPLURDGER	5/6	33	New appear routine
PARALLAX STARFIELD GENERATOR	7	5	Scrolling Starfield routine
SCROLL	7	5	Simple scrolling demo
HORIZONTAL SCROLLER	7	6	Various horizontal scroll routines
WORD MIX	7	7	10-line text game
The MIXER INTRO	7	8	new appear routine (+total documentation)

NEWSLETTER INDEX

Name	Issue	Page	Description
FAKE VIRUS	7	12	(M-Code) Francois' Practical joke
EXTENDING ZONES..	7	13	routine for more than 128 zones
VECTOR NUMBERS	7	15	Number display routine
HELLO EXTENSION	8	11	(M-Code) Simple STOS extension
FRANCOIS' SHOOT HIM UP	8	16	Francois' face all over the screen demo
The GAUNTLET STYLE GAME WITH TOME	9	3	Simple multi-directional scrolling with TOME
BOMBER	9	5	10-liner game
DEGAS FILE-DECOMPACTOR	9	6	File decompaction routine (+total documentation)
NOT QUITE SO USELESS EXTENSION	9	9	(M-Code) Simple but usefull extension for STOS
DICE GAME	9	11	Simple dice game for beginners
SCROLLING TEXT (Learners)	9	15	Smooth scrolling text routine
SCRAMBLE	9	21	10-liner game
PING	10	2	10-liner game
REMOVER	10	3	Rem removal program
TANK BUSTER	10	4	10-liner game
USEFULL pt 2 (see Issue 9)	10	5	(M-code) usefull compiler extension
TINYPIC DECOMPACTOR	10	9	Decompact TINY format pictures
CALCULATOR ROUTINE	10	12	Simple calculator
JUMPING BALL	10	14	Various Gravity effected programs
BRICKOUT	11/12	10	10-line breakout clone
PUZZLER	11/12	11	10-line sliding block puzzle
DATABASE	11/12	12	10-line card index database program
SEQUENCER	11/12	13	10-line music sequencer
SPACED INVADER (Learners)	11/12	16	10-line game with FULL documentation
SCROLLING TEXT	11/12	20	Large text over a backgroud screen
SPRITE ENGINE	11/12	22	More than 15 sprites at once from a 2 line routine !
STAC to STOS Font Converter	11/12	24	Converts STAC fonts into STOS format (+Total Documentation)
SNEAKY M-CODE	11/12	29	Small STOS extension to protect your programs from being listed.

NEWSLETTER INDEX

NEWSLETTER INDEX

Digital Dimensions review, Issue 11/12,P3 Reviews of two games from a new budget software company
Speeding, Issue 11/12,P28 STOS commands speed tested
STOS Newsletter Article Index, You're reading it dummy !
About STOS P.D, Issue 11/12,Pxx Public Domain, Shareware and Licenseware explained

Not counting listings done in reply to letters and the programs on the STOS Word disk, that's 59 programs ! That's probably more STOS programs than all the ST Magazines put together ! Now would be a good time for you to test out your database programming skills, to put all this information on an index system (see the Database 10-liner in this issue).

Advertising in the Newsletter

As you may have noticed, we are running advertisments in the STOS newsletter. These adverts pay for extra pages in the newsletter (so you get more articles) and also pay for the reader's articles that we publish (so I don't have to write all the new articles !). From the advertisers point of view, 700 people subscribe to the STOS Club, (and at least twice that read the newsletter).

Considering the page rates, we reach more STOS users per £ than any other magazine. In fact, Digital Dimensions advert in issue 10 got more response than either of their adverts in ST User or ST Format !

Advertising rates for the newsletter are minimal, but space is at a premium (we would prefer usefull STOS articles on the pages, rather than adverts), so, if you are planning on advertising, contact us now on

The prices are:	
Two Pages:	**£80**
One Page:	**£50**
Half Page:	**£30**

Advertisements can be supplied either as a text file (with or without graphics), or as an A4 sheet, ready to photocopy. The latter is the better option, as you can send the advert directly to Sandra for inclusion in the newsletter, thus saving time and allowing you to design the advert to your own liking.

No obscene advertisements or any advertising for illegal products or activities (such as software piracy or Knunge wurdling) can be accepted for the STOS newsletter.

PUBLIC DOMAIN NEWS

❑ Plumb Crazy Too Good

It seems that STOS is definitely good enough to produce commercial games with, as Empire, producers of the game Pipemania, have asked Sandra (very politely) to remove Keith Feeney's game, Plumb Crazy (SPD58) from the STOS P.D library, as it was to close to the original, which is about to be re-released as a budget game. As Empire asked very nicely, and didn't threaten massive legal action or anything, Plumb Crazy is no longer available on P.D.

As you might have read in the editorial, the P.D library is undergoing a massive change, and is being completely re-built for the new year. ALL the single sided disks are being converted to double sided, with source code and extra programs included on the new disks. All programs that don't reflect the current state of STOS programming are being dropped from the library (such as the earlier demos), as are any which won't run on the STE. The end result should be a much better P.D library.

Single sided drive owners needn't worry, as Sandra is prepared to split double sided disks over two single sided disks, but you'll have to pay for two disks.

Because of this re-vamp, there is no P.D list this issue. Please send a stamped, self addressed envelope to Sandra to get the latest P.D catalogue. Instead, we'll have some mini-reviews of the best P.D software:

Pile Up: Author Russell Moll. SPD65

This is likely to be the next "banned" program if Tetris is ever re-released. It's better than the original in gameplay, although the graphics could be better.

The major difference between Pile Up and Tetris, is that Pile Up is much more devious, and changes from level to level. There are also a couple of suprises for finishing some of the levels.

One very good idea has been the inclusion of bonus points at the end of each level (completed by getting a set number whole lines), the lower your pile of bricks is, the more points you get. Very simple, very effective !

The deviousness comes in with the different blocks. One of them is made of of four bricks in a diamond, with a hole in the middle ! Which means that unless you know some great dimension twisting tricks with the joystick, you will always end up with a gap that can't be filled ! Fortunately, you also get bombs instead of bricks from time to time. These will obliterate the bricks in a small area, thus clearing bits that make things difficult.

Verdict: Great fun, get it !

Bounty Hunter: Author Budgie UK: SH14

This is a Wild West Role Playing Game that puts you in the shoes of a great bounty hunter, who has to hunt down wanted criminals and either arrest or shoot them. This involves wandering around the wild west, trying not to run out of food (or bullets !) and tracking down the baddies by listening in towns for rumours. You have to be carefull in towns, as you can be run over by the stagecoach, and shot by the Sheriff if you do naughty things like shooting shopkeepers (and not running away fast enough !). Again the graphics aren't brilliant, but they do the job, and this game scores points on its gameplay. If you're into role playing games, you are probably going to like this one.

Verdict: Good fun for role players or western fans.

THE INTERNATIONAL
Public Domain & Shareware Guide

With thousands of ST customers worldwide, as far afield as Japan, Italy, Australia, France, Saudi Arabia, Holland, Belgium, Singapore, Sweden, Denmark, Germany, Norway etc.etc.

A.S.P. Approved

The FIRST and at present the ONLY ST approved Shareware service

(The Association of Shareware Professionals)

ST Format Gold Award

Winner of the public domain and Shareware service of the year award

(By the readers of ST Format

Special 10 Disk Starter Sets

Only

£14.95

Education & Games Sets

Five disks from only

£10.00

One of the country's leading distributors of the Budgie UK collection

Exclusive Adventure Helpline service Available NOW

Send £1.95 for the latest edition of the International Public Domain & Shareware Guide, with shareware reviews, articles, and details of all the best in public domain available for the Atari ST range, or send only £2.50 for catalogue AND disk cat' database

Goodman Enterprises

Longton, Stoke-on-Trent.

Modem owners - you can now order via the Bath BBS FAX service on

A LETTER !

Dear Aaron,

This is the first time I have written to you. I enclose a listing to a problem that Mr P.J.Turner set in issue 9 of your STOS Newsletter 'letters' column regarding the two second update of the Atari's system clock. I have devised a routine and briefly tested it (so have I . . . Aaron) and find that it is quite reliable for use in many types of applications.

I hope this may be of some use to Mr Turner or any of your readers with a similar problem.

John Fermor.

```
10 rem ****************
20 rem * REALTIME Clock
 Routine *
30 rem * Written by John
 Fermor *
40 rem * In STOS Basic *
50 rem * Atari ST
 NUMAsoftware *
60 rem ******************
70 rem
80 cls : key off : hide on :
 curs off : H=val(left$(time$,2))
 : M=val(mid$(time$,4,2))
90 S=val(right$(time$,2)) :
 S$=str$(S) : M$=str$(M) :
 H$=str$(H) : timer=0
100 if timer=50 then gosub 130
110 gosub 160 : gosub 190 :
 gosub 220 : locate 0,1
120 print H$;":";M$;":";S$ :
 goto 100
130 if S=59 then S=0 else inc
 S : timer=0 : return
140 if M=59 then M=0 else inc
 M : timer=0 : return
150 if H<>24 then inc H :
 timer=0 : return
160 rem check SECOND string
 lengths
170 if (S>=0 and S<=9) then
 S$="0"+right$(str$(S),1) :
 return
180 if (S>=10 and S<=59) then
 S$=right$(str$(S),2) : return
190 rem check minute string
 length
200 if (M>=0 and M<=9) then
 M$="0"+right$(str$(M),1) :
 return
210 if (M>=10 and M<=59) then
 M$=right$(str$(M),2) : return
220 rem check hour string
 length
230 if (H>=0 and H<=9) then
 H$="0"+right$(str$(H),1) :
 return
240 if (H>=10 and H<=23) then
 H$=right$(str$(H),2) : return
250 if (H=24) then H=0 :
 return
```

A Good solution, However, I am going to be a little critical ! An easier way to fix the length of a number for printing as a string would be:

```
S$=right$("00"+mid$(str$(s),2),2)
```

and the last line of the subroutines should always be a fixed return (You never know what values a warped user is going to put in your program !). But the program works, so it doesn't really matter too much. Well done !