STOS Newsletter

Issues 5 & 6 • STOS User Club

Welcome to a bumper issue of the Newsletter!

Because of the long summer break we've decided to put issues 5 and 6 into one big magazine that's twice its normal size – and twice as useful to all you STOS users.

There's exclusive information on a brand new range of STOS products that will be released over the next few months, a complete car racing game with vertical scrolling, a greatly extended Public Domain library, a listing for a full-screen editor that allows you to do away with line numbers and add labels and procedures, articles on using the Midi port for music or head-to-head games, and much more.

It's also time to resubscribe – and we've put together a very special offer to thank you for your support and to encourage you to carry on making the most of STOS.

Send a £10 cheque for the next six issues and you'll get a disc containing STOS Word – a full-feature word processor which even allows you to use icons in the text – perfect for letterhead designs, signatures, and so on. You'll find more details in the special sheet which comes with this issue.

It's at this point I'd like to say that I will be stepping down from the editor's seat after this issue because of other commitments. It's been a fascinating year in which the Club has developed in leaps and bounds. I've really enjoyed corresponding with many of you and talking on the phone – and I'll miss you all. Aaron Fothergill will take over the STOS Hotline and editorship of the Newsletter.

Now, on with the issue...

Pat

STOS in the mags

New Atari User has joined Atari ST User in producing a regular section on STOS. The August issue includes a program to dump sprite data as text and an animated demo. The August issue of Atari ST User includes a version of PacMan on the cover disc to match up with an article inside by STOS artist Dave McLachlan.

STOS book in November

This autumn will see the publication of **Game Maker's Manual: Atari ST and STOS Basic** by Stephen Hill, author of the STOS manual. Published by Sigma Press, and with a provisional price of £11.95, the title will effectively take over from where the manual left off – explaining particular topics in greater depth and benefitting from Stephen's further experience with the inner workings of STOS gleaned over the last year. The book will explain how to develop adventures, platform games and other arcade games, and will include full details of how to create extension files and generate 3D graphics.

New releases for Xmas

The festive season will see the launch of a number of STOS products. STOS Musician consists of a top-class music editor together with more than 200 attention-grabbing tunes. There's even a jukebox program so you can hear them all.

STOS Games Pack contains four excellent games, many being finalists in the recent Awards, and STOS Vidi Digitiser allows you to grab frames off a video in real time. See inside the newsletter for more details.

More news on the back page and inside.

LETTERS

STOP THE MUSIC

I've recently joined your excellent STOS club and on reading the newsletters it appears that I know of a bug which hasn't come to light yet – MUSIC FREEZE.

I've written a simple two player dogfight game which has music playing in the background. I wanted to SHOOT and BOOM machine guns so:

```
MUSIC FREEZE : SHOOT : MUSIC ON
```

... seemed the logical way to do it! Yeuch!! Try it!! The note playing when MUSIC FREEZE is hit continues to play (i.e. it holds), the SHOOT is ignored and then the tune continues. Any ideas?
David Russell, Leicester,

You must wait for the SHOOT sound to finish before restarting the music. For example:

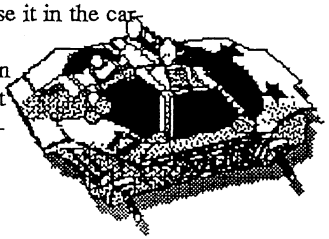
```
MUSIC FREEZE : SHOOT : WAIT 50 :
MUSIC ON
```

This is necessary because STOS produces the SHOOT effect during interrupts and not in a queue list.

MAESTRO MODS

I have a problem with STOS Maestro. I am using the sampler as mentioned in ST/Amiga Format and the only way I can use it is to sample using the sampler program on the disk supplied with the magazine and save it to disk and then load it into Maestro. This is because Maestro samples through the cartridge port whereas the ST/Amiga Format sampler samples through the printer port.

Is it possible to change the program to accommodate this or is there a way of getting or making an adaptor to use it in the cartridge port?



Any light on this subject would be gratefully received.
R. Rose,

It would be very difficult to make these changes as the printer port operates in a different way – it's also slower to access than the cartridge port. I'm afraid you'll have to carry on as you are or buy the Maestro Plus upgrade.

HINTS & TIPS COMPILED

I bought STOS mainly for its advanced BASIC features – I have no ambitions to write games! Despite this I have learned to appreciate what a comprehensive and well-documented product STOS is.

I also appreciate your newsletters – each new issue contains more hints and tips about STOS. The trouble with all this information is that the more there is, the harder it is to find: When you hit a problem you remember reading about it but it seems to take ages to locate.

In an attempt to solve this problem I wrote a program (in STOS Basic of course) which allows me to store all these hints and tips on disc. It gives me the capability to hold up to 500 such tips, each with up to three IDs (words such as SPRITE, PROGRAM, FIX or anything you like) with a brief description and the issue and page number so that it can be easily located.

The tips can be added, modified or deleted. They can be displayed on screen and saved when the exit routine in taken. Once saved the tips can be reloaded when the program is next used.

I wonder whether the program would be of any interest to you? There's nothing fancy about it (it is in Basic after all). I did put some sound into it – but this did get a bit tiresome – so I made the sound optional.

One thing I didn't manage to do was to get rid of the STOS menu at the top of the screen – is this possible?
Alan Constable, Maidstone, Kent

It's a doddle to get rid of the menu – simply type KEY OFF. I'm sure lots of members will be interested in your handy database – perhaps interested parties could send to you for the latest version (with disc and SAE of course) – Pat

LETTERS

MOVING ARROWS

I am having a couple of problems with STOS:
1. I am having trouble manipulating a sprite with the joystick and changing the sprite according to the direction the joystick is moved. An example of what I am trying to achieve is moving an arrow around the screen with the arrow pointing in the direction it is moving.
2. I cannot get a map I have made on the map definer into STOS Basic.
James Turner, Middx,

To get the sprite to change with the joystick direction use a variable to store the sprite number and change the value when you move the sprite. For example:

```
10 rem Sprites+joystick routine (needs a
sprite bank loaded)
20 X=160 : Y=100 : SPN=1
30 rem Sprite 1=left,2=right,3=up,4=down
40 sprite 1,X,Y,SPN : update
50 if jleft then dec X : SPN=1
55 if jright then inc X : SPN=2
60 if jup then dec Y:SPN=3
65 if jdown then inc Y:SPN=4
70 goto 40
```

The .MAP files created with the STOS map editor can only be loaded into the editor itself or by using the BLOAD instruction. To use a map generated by the map editor use the SAVE.ASC option to generate the required section of code to add to your program. You then load up your program and do a LOAD "MYMAP.ASC" to add the lines to your program (they start at 50000). There are full instructions for this on the accessories disc (MAP.DOC). However the STOS map editor system can only handle maps generated with sprites, it can't handle scrolling maps like in Gauntlet etc. and it is very slow. It is good for designing backdrops for multiple level games where the background is only made up of a limited number of sprites and doesn't have to move. – Aaron.

HELLO WORLD

Thanks again for your help on the phone yesterday with STOS. *(Sue is producing a disc magazine called SYNTAX and was having difficulty with producing .PRG files – Pat)*

After I came off I did a new disk, dragging over the STOS folder, did your "TEST.PRG" of 'hello' (10 print "hello" 20 goto 10) and it worked perfectly. Great! Tried again with my program and though I didn't get a TOS 35 error or two bombs, after about 30 seconds it dropped back to the desktop.

I resaved my program again calling it "test.prg" and it worked so all I can think is that the name I was using "syntax1.prg" is the problem. Perhaps SYNTAX is a restricted word?
Sue Medley, Kent,

The problem you experienced was due to a problem with STOSCOPY which is now resolved. See Newsletter Issue 1 or write to Mandarin.

ON THE FLY

One feature that seems to be missing from STOS Basic is the ability to evaluate expressions (mathematical equations) that are entered while a program is running. The old Atari 8-bit Basic had its 'Return key' mode and Fast Basic (like BBC Basic) has an EVAL command. I have tried various dodges to simulate this but to no avail. Is there anyone out there in the STOSiverse that has cracked this one?

Also, hands up all those who have pressed F4 instead of F5 and overwritten the very file they were loading! A 'file already exists' prompt would have been nice.
Caryl Jones, Sussex,

I suggest you change the CONFIG program so you can't use F4.

The following EVAL routine works by stuffing the formula into the key buffer using PUT KEY along with ": GOTO line"

LETTERS

and a "!" to act as a RETURN keypress. Another way would be to write a line with the formula to disc as an ASC file, then load it into the program. Both of these methods will only work with an interpreted program.

```
5 A=1 : B=2
10 line input "Enter Your Formula :";F$
20 clear key
30 put key "C="+F$+":goto 50`"
40 end
50 print "The Result is ";C
```

MIDI MAGIC

I am a music teacher and would like to make a request concerning that area. How about some articles on using MIDI with STOS programs?

Alternately could you provide info via the newsletter on books that would be of use in this area for a not too experienced programmer?
Leigh Marriott, Portland Australia.

Your wish is our command. See the articles on MIDI later on in this Newsletter.

BOOK SEARCH

My knowledge of Basic is a bit limited (VIC 20 Basic part 1 & 2, Spectrum 30 hour) and I read these some five years ago. Could you suggest a title that would help?

Try Alcock's Illustrated Basic (Cambridge University Press) as suggested in the STOS manual.

HELPING HAND

I would like to help try and sort out some of the many problems experienced by fellow STOS users. I am a computer operator by trade but I have a BTEC National Certificate in computer studies and have been programming for about six years. I have a fairly broad knowledge of computers and high level languages but very little experience of low level languages or hardware.
Dorking, Surrey,

I'm sure that a number of readers would like to take you up on this offer.

HIDDEN COMMAND

The reference card (first page) mentions a command under "Variable and string commands" which I cannot find in the manual. This is listed as:
=INPUT list/" chars" ,delay
What does this mean? Or is it a misprint?
Martin Taylor,

This produced some red faces at Mandarin – it's a misprint!

LONGER MENU NAMES

Is it possible to use LINE INPUT # with separators other than 'ENTER'?

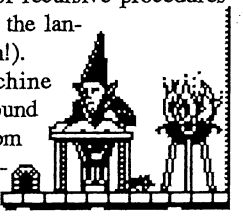
Another question: Is there any poke pill or potion that would allow me to make a menu option (title) more than twenty characters long? This may seem unlikely but I can think of a use (I won't go into it just now).

I wonder if Mandarin will pull their fingers out and produce a new version of the manual.
Depressed of Banbury!
The answer to your first two questions is a straight 'No'. Mandarin have no plans to rewrite the manual at the moment. It is far from perfect, but a damned sight better than other manuals I can think of!

EXTENDING STOS

I have a few questions and suggestions for improvement of the STOS package.

First the inclusion of recursive procedures would greatly improve the language (please Mandarin!).



When using machine coded additions I have found that unless cleared from memory after use an error occurs (even with

code that runs perfectly by itself). Can anyone help?

Also is it possible to add my own machine code routines to the language as an Extension file?

As long as the machine code is in a bank you should have no problems. See Stephen Hill's new book for detailed information.

FUN WITH FUN SCHOOL

First of all thanks for a brilliant newsletter – the last issue was better than ever. Having been a newsletter editor myself, I feel for you with regard to receiving articles from members. I have finally had pangs of guilt and decided to write, if nothing else I get into "the 5%" who have written in.

I have worked on a couple of simple programs: A disc label printer, a maths program for my son, flashcards for my younger daughter, using the zoom example on page 143 of the manual, (which, incidently, has a bug... change the end of line 70 to :Y2=Y1+16). These are some of my 'better' products. I intend to provide you with listings in the near future.

The only 'disadvantage' with the STOS Compiler is that once a program has been compiled to a GEM run.prg, it cannot be listed using STOS. This prevents examining and changing the program, which we have probably all done to ZOLTAR etc. In fact, I bought the Fun School 2 pack, which is written in STOS, and I really enjoyed pulling the various programs to pieces to see how they worked (on back ups of course). The author of Fun School 2 has included many rem statements explaining how the program functions. Reading these statements is an excellent way to learn STOS. If the program had been compiled, I could not do it. I hope that future programmers (particularly educational software) do not compile, because adapting the software to your own child's ability is a real benefit.

Larry Green Decimomannu

PS: My 9 year old son, would like a pen-pal STOS user of similar age, any takers?

I can see your point that you can't list PRG files created by the Compiler – but you can't call it a disadvantage as that's exactly what you want the Compiler to do!

SPRITES IN A SPIN

I need some help with STOS. How do you get sprites to go round in a circle. I can not even think of where to start so please could you help me ? I have enclosed a disk so if you have an answer I would be thankful if you could place it on the disk. I have got all the new packages to do with STOS e.g STOS compiler and yet again Mandarin have done it, they're Ace!

Daniel Bates, Selsey

The sprite now circles! The program ORBITER.BAS I've sent you works by using the same sort of routines that you would have to use to draw circles without the CIRCLE command. The variables CX & CY contain the centre x,y co-ordinates of the circle, and RX,RY are the X & Y radii of the circle (or ellipse). The program then uses the variable Q# to count from 0 to 2pi (there are 2pi radians in a circle). The point on the edge of the circle at Q# radians is calculated by:

```
X=CX+Cos(Q#)*RX Y=CY+Sin(Q#)*RY
```

so the command:

```
sprite 1,CX+cos(Q#)*RX,CY
+sin(Q#)*RY,1
```

...would put sprite #1 on the edge of the circle at Q# radians. You can use this routine for drawing circular objects as well as making sprites go round and round, and by changing bits in the formula you can generate quite a few weird effects! I hope the program solves the problem. Have Fun! – Aaron

MUSIC EXTENSIONS

I think that the bargain of the year must be buying STOS Games Creator. It's Brilliant!

One thing that I haven't been able to work out

yet is how to use the MIDI out port. Although the manual is quite comprehensive, it does not give you any examples of MIDIing. Input is enabled using the PORT(n) instruction.

If the programmers of STOS (or anyone else for that matter) are looking for ideas to improve their product, I have one. It is to do with the Music Definer. Instead of just having voices 1,2 and 3 playing from the Atari's sound chip, have channels 4,5,6 (or more) going out to a MIDI device. Playing the background music and the sound chip doing the explosions. Perhaps even one of the channels could be reserved for the Maestro Sampler. I hope this doesn't sound too complicated.

Also will anyone be doing a review of the sampler and the compiler?

Daniel Shaw, Guernsey

(See the article about Midi this issue). Maestro was reviewed in Issue 4. The compiler is brilliant and Maestro isn't quite perfect but it does the job for sound effects and sampling sections of music and is far better than any other ST sampler I have tried. Its only drawback is the fact that a stored sample can only be played back at either a fixed rate from 5-32Khz in 1Khz steps or in steps of a semi-tone over 1 octave, so you can't do fine shifts of the sample's pitch like pitch bend on a synth. Otherwise it is damn good quality for an 8-bit sampler. – Aaron

BUGGY PATTERN

Adam Parker Rhodes of Muswell Hill phoned the Helpline with a strange problem with STOS. He was trying to use the SET PATTERN command to define his own fill pattern for a program. However all he got were Atari Symbols! Also he asked how to do graphics and multiple sprites without getting flicker from sprites overwriting each other.

The good news is that you are completely sane and didn't type the program in wrong! The bad news is that having tested the SET PATTERN routines myself and found them somewhat lacking in the area of doing things. I phoned Richard Vanner at Mandarin. According to him the SET PATTERN command will only work in high resolution due to someone forgetting to finish it! (So file it away with all the Trap #4 and #6 commands unless you are using High res!).

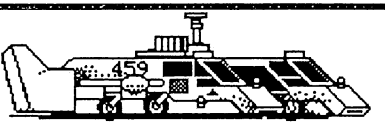
To help fix your other problem with the flickering sprites, here is a suitable screen flip routine (feel free to change line numbers and bank numbers to suit.).

```
10 reserve as screen 5 :rem Reserve a
screen for the background (non
changing) graphics.
20 logic=5:autoback off: rem We are now
working on the screen in bank 5
30 rem Draw or load in your background
here
40 logic=back : rem From now on we are
doing all the graphics out of sight on the
background screen and then screen
swapping them into view
```

(whenever you want to update the screen do a gosub 100)

```
100 rem Do screen update
105 screen copy 5 to back
110 rem Do sprites and graphics here
120 update : rem This fixes the sprites
(and mouse) in position for a frame
130 screen swap : wait vbl : rem Make
the background screen visible and wait
one update to make sure there is no
flicker
140 return
```

This tends to cure 99% of all known sprite flicker problems although you need 32k of extra memory for the background screen as you are effectively working on three screens instead of the usual two. Remember that the screen you are working on is not the one in view, as this may cause confusion if the program ends or is stopped while Logic=Back. Anything you type will appear out of sight on the background screen. Just enter logic=physic or logic=default logic to get back to normal (or UNDO twice). Have fun, and good luck with the program. – Aaron ■



Everything you wanted to know about PACK

... but were afraid to ask!

François Lionet explains how his clever screen compacting routine works

Packing a screen in STOS is made simple by using the COMPACT accessory. But how does this program manage to crunch whole or even sections of screens using the PACK command? Well this article explains all, giving the techniques used and revealing hidden parameters of PACK.

The PACK command

The full and true syntax of the PACK command is:

pic_length = PACK picture, destination [,resolution,flags,height of block, start in x, start in y, size of x, size of y]

picture is the address of the screen to be PACKED, a bank number can be used or an absolute address.

destination is where the new compacted screen will be stored: Either bank number or absolute address may be used.

The instruction can work with just these two parameters. If this is the case then the whole picture will be packed, the other parameters will be set up by STOS.

The extra parameters mean the following:

resolution tells STOS which mode the picture has been drawn in. You can use a different number if you wish but the saving of space won't be as much as the true resolution.

flags This sets information about the colour palette.

Bit 0: If on, the whole colour palette will be set to zero before the picture is unpacked, thus giving a clean display. If this bit is clear then the palette is not altered, this is useful when unpacking to a bank.

Bit 1: When on, the palette will be changed to that of the screen to be unpacked. The palette is not altered when this bit is clear.

The default for the two bits is 1 & 1. This default is used when you only specify the two parameter version of PACK.

height of block We'll cover this later, suffice to say at this point that it sets the packing block size in Y lines.

start in x states the top left x co-ordinate from where the picture will be packed. The value must be given in the number of words - for low resolution, 0 would be the far left, 10 in the middle and 19 the far right.

start in y states the top y co-ordinate of the picture.

size of x indicates how many WORDS across from the start x will be packed (width).

size of y must state the number of packing squares in Y to process. The full length of the picture area to pack is therefore *square size * size in y*.

If you wish to use this extended form of PACK you must be certain to set all 9 parameters otherwise STOS will just throw it back at you.

When set up and called, PACK returns the size of the newly created and crunched up picture to the appropriate variable, in this case *pic_length*.

A packing process

Compacting can be achieved in a number of ways, the simplest is described here.

Imagine an empty screen, consisting of 32000 bytes, each set to zero. A simple compactor will pack this screen into two numbers: 32000 and 0. You can see that the unpack program will have no problem to recreate the original screen! Here are

two STOS Basic routines which will PACK and UNPACK a screen using the technique that I have just described for the empty screen.

**PACKing routine (not testing for the end of picture) ORIGIN=picture address DEST=packed picture address*

```
100 CPT=0 : B1=peek(ORIGIN) : inc
ORIGIN : poke DEST,B1 : inc DEST
 110 B2=peek(ORIGIN) : inc ORIGIN : if
B2=B1 then inc CPT : if CPT<255
then 110
120 poke DEST,CPT : inc DEST :
goto 100
```

**UNPACKing routine ORIGIN=packed picture address DEST=address of screen for unpacking to*

```
200 b=peek(ORIGIN) : inc ORIGIN :
CPT=peek(ORIGIN) : inc ORIGIN 210 for
X=0 to CPT+1 220 poke DEST,B : inc
DEST 230 next X 240 goto 200
```

This would work perfectly for our empty picture but imagine a screen filled alternately by 0, 255, 0, 255... Packing this picture with such a technique would be catastrophic! The result would be twice the size of the original picture. I therefore chose a process of compaction which was far more intelligent for STOS.

STOS's compacting method

For each byte of the screen we'll have a single corresponding bit within an index table. As the size of the picture is 32000 bytes, there will be 32000 bits within the table – that is 4000 bytes. We also need to build another table – the bytes table.

When the compacting begins, it starts off by examining the bytes from the original picture systematically. If the current byte examined is different from the previous one (or if it is the first byte) it sets the corresponding bit within the index table to 1 and stores the current byte into the bytes table. If the current byte is equal to the previously examined byte then the compactor simply sets the appropriate bit in the index to 0.

How does the unpacking process work?

It first reads the current bit in the index. If it is set, it will read a byte from the bytes table, if it isn't set it uses the previous value. The retrieved byte is then poked into the picture and the loop starts again, looking along the index until it reaches the last bit.

Let's see an example:

Original Picture: $00 $FF $FF $FF $0A $0A $0A $01

would produce:

Index table (in binary):	%11001001
Bytes table	$00 $FF $0A $01

The main advantage of this method is that it won't double the size of a picture. In the worst case possible it would produce a screen with an index 4000 bytes in size and a bytes table, 32000 bytes long. This would only occur if every byte in sequence changed in the original screen. Some pictures do have complex designs but it is very rare for a screen not to compact successfully using the above technique.

Using this method to pack a blank screen we would get a bytes table consisting of just one byte, but an enormous 4000 byte index table, beginning with the first bit set and the rest set to zeros. In this case the method has reduced the picture data to nothing but I end up with an ugly overhead with the index. This is very bad – packing a blank screen into 4k – I would be ashamed of selling such a routine.

The solution? Yes, of course: Pack the index table! After packing this table we'll get another index table, which will be 4000 / 8 = 500 bytes long. That's more like it! And another bytes table which will vary on the complexity of the picture. For our blank picture it would just be 2 bytes long: $80 and $00. So the final size of the compacted blank screen would be: 500 + 2 + 1 = 503 bytes (for a 32000 byte picture).

One may ask, can we pack the first bytes table? The answer is no, because each byte is different, thus the compaction would make the file bigger.

Can we pack the second index table? Yes, I

tried, and there is no advantage, it just slows down the process rather than helping save memory.

The UNPACKing program must first rebuild the first index table using the second generation index table and bytes(1) table. It then rebuilds the actual picture using the unpacked index and bytes(1) table.

Still trying to grab memory?

The Atari's screen is divided up into bit planes. The STOS compactor accessory explores memory plane by plane, So it will loop four times in mode 0, twice in mode 1 and once in mode 2.

Imagine a screen with a vertical line on an empty background.

Our first method of compacting would be line by line, taking the first byte on the left, then the next one, until the right hand byte is reached. This continues down the screen for each line. Now, because there is a line in the middle of the screen, each line will generate at least three bytes for the bytes table: $00 for the left of the screen, $01 (for example) for the line, and $00 again for the right hand side of the screen. It will therefore create 200 * 3 = 600 bytes.

The second method involves exploring the screen by small blocks. Starting with the first byte of line 1, then the first byte of line 2...up to the last line of the block. Then we come back to the top of the screen to the second block along. When a whole line of blocks have been explored the routine jumps down to the next line, and so on until the whole screen has been covered.

Block size = 4

1st byte	5th byte
2nd byte	6th byte...
3rd byte	7th byte
4th byte	8th byte
Second line of blocks, beginning at line 5 of the screen...	

Let's say that the height of our squares is 200 lines. The bytes table would then be:

$00 all blocks from left to vertical line
$01 Block that contains the line
$00 Right hand side of picture

As you can see, only three bytes long!

For a more complex picture the COMPACT.ACB accessory tries various block sizes: normal (line by line), 2, 4, 8, 16 and 32. The height of block parameter in the PACK instruction contains the the size in Y of blocks (1 being line by line). For a horizontal line on an empty screen, line by line would be the best procedure.

A packed picture header

The header of a packed picture (saved using the "Save Binary File" within COMPACT.ACB) is detailed in the table below.

Well friends, I hope you did not fall asleep reading this article – I'm feeling a little tired myself! Next issue, we'll see how to make an interpreter extension like COMPACT.EXA. Bye for now, I've got to get back to AMOS! ■

Offset	Size	Value	Designation
0	.L	$06071963	I.D. code (my birth date!)
4	.W	0/1/2	Resolution mode
6	.W		X start of picture (in words)
8	.W		Y start of picture (in lines)
10	.W		X size (words)
12	.W		Number of blocks
14	.W		— Unused —
16	.W		Y size of squares (line)
18	.W		Flags
20	.L		Offset to bytes(2) table
24	.L		Offset to index(2) table
28	.W .W .W ...		Colour palette
70			Start of bytes(1) table
...			Start of index(2) table
...			start of bytes(2) table

Adding procedures to STOS

Aaron Fothergill shows you how to create more structured programs

This month's special offer is a program to enable you to use procedures with STOS. It will also enable you to convert programs from GFA Basic, ST *(very)* Basic or basically any other Basic. However the conversion features are (hopefully) going to be ready for the next Newsletter.

The first listing enables you to edit a STOS program using labels and procedures instead of line numbers. You can load in a STOS program in ASC format, edit it and save it in either ASC or PCP (a custom file type for this program, so you can save the program with the labels). When loaded in ASC format the line numbers are removed (although the ones within the line aren't, so any GOTO or GOSUB statements will have to be re-worked. I'm working on making the program assign labels to these automatically. You can also type in your program from scratch. The only other option on the menu at the moment is "Convert". Click on this and the PCP (Pre-editor Converter Program) will convert the labels and procedures to line numbers and save the program as an ASC file. All you have to do now is type:

```
new
load "MYPROG.ASC"
```

...watch it type in your program and then type **run.**

Syntax and such

In this version of the program Labels and Procedures are exactly the same. To declare a Label or Procedure, put the command:

LABEL mylabel. or **PROCEDURE mylabel.**

...at the start of the line. *mylabel* can be any length of text as long as it doesn't include quotes or full stops. It must end with a full stop however. Any references to that label are done as just the label name with the full stop, for example:

```
    Print "This is my program in PCP"
    A=1
LABEL loop. Print A
    inc A
    if A<11 then goto loop.
    gosub thisline.
    gosub thatline.
    input a$
    on val(a$) gosub thisline.,thatline.
    end
PROCEDURE thisline. Print "This is a a pro
    cedure!"
    return
PROCEDURE thatline. Print "Just like that!"
    : return
```

Note that multiple command lines are allowed as normal. For future compatibility, any routines you want to call with parameters, define using the PROCEDURE command, as I will be adding routines to enable you to pass parameters to a procedure just like in other more structured Basics (The ones that can't do good old spaghetti code!).

Future updates to this program (all through the Newsletter) will include the ability to completely re-define all the STOS commands with alternate syntax. This will enable you to convert Basic programs from other ST Basics or even from other computer Basics! This is good news since I've just been given a VIC 20 and I've got loads of my old (very old!) Basic programs hidden away for it somewhere! Another update will be the ability to create Subroutine Libraries to save you having to type in the same bits of code for each program. You just select what ones you want from the library and add them to your program. Please phone or write with any ideas for bits you might want on the program. (The press-

one-button-and-it-writes-a-program-for-you routine is on its way!).

Now for the difficult bit – the typing in! Alternatively you can send £2 (or a formatted disc and £1) to Sandra (see page 34) ans she will send you PCP and all the other listings from this bumper issue – and anything else she can lay her hands on.

```
10 rem ************************
12 rem * STOS Basic Pre-edit
13 rem * Converter Program
14 rem * Aaron Fothergill
15 rem * Shadow Software 89
16 rem * For STOS User Club
17 rem * Newsletter .
18 rem *
19 rem ************************
20 dim L$(1000),L2$(1000),LBL$(50),
LBL(50),ERJ(20) : rem Change this for
big programs over 1000 lines or 50
labels
21 dim T$(3) : T$(0)="." : T$(1)="o" :
T$(2)="O" : T$(3)="o"
25 L$(0)="rem Created with PCP"
30 mode 1 : key off
40 A=1
41 B=1 : read M$ : if M$="" then 50
42 menu$ (A)=M$ : repeat : read M$ : if
M$="" then inc A : B=99 else menu$
(A,B)=M$ : inc B
43 until B>10 : if B=99 then 41
45 data " FILE ","Load .ASC","Load
.PCP","Save .ASC","Save .PCP"
,"Quit","","  CONVERT ","Convert
Program","","PRINT","Print
Listing","",""
50 ND=1 : menu on
55 on menu goto 500,600,700
56 on menu on
60 if ND=1 then gosub 100
61 K$="" : K=0 : while K$="" and K=0 :
K=mouse key : K$=inkey$ : wend
62 if K=1 then locate xtext(x
mouse),ytext(y mouse)
63 if K$<>"" then gosub 110 : goto 60
99 goto 61
100 rem Display lines
101 Y=ycurs : X=xcurs : A=0 : VL=0 :
repeat : locate 0,VL
102 print
mid$(L$(A+LT)+space$(80),1,80); : inc A
: inc VL : until VL>22
103 ND=0 : locate X,Y : return
110 rem Edit a line (Line length up to
max string length or end of memory!)
111 K2$=K$ : X=xcurs : Y=ycurs : P=X+1
: L$=mid$(L$(ycurs+LT)+
space$(80),1,max(80,len(L$(ycurs+LT))))
112 LP=1
115 locate X,Y : K$=K2$ : K2$="" : K=0 :
while K$="" and K=0 : K$=inkey$ :
K=mouse key : wend
116 if K=1 then X=xtext(x mouse) : locate
X,Y : P=LP+X : goto 115
117 if K=2 or K$=chr$(13) then gosub
220 : L$(LT+Y)=L$ : X=0 :  locate 0,Y :
print mid$(L$+space$(80),1,80); : gosub
185 : X=0 : return
125 if asc(K$)=0 then gosub 200 : rem
Special Controls
130 if K$=chr$(8) then mid$(L$,P-
1,len(mid$(L$,P)))=mid$(L$,P) :
L$=mid$(L$,1,len(L$)-1) : gosub 170 :
locate 0,Y : print
mid$(L$+space$(80),LP,80); : locate X,Y
131 if K$=chr$(127) then
mid$(L$,P,len(mid$(L$,P+1)))=mid$(L$,P+1)
: L$=mid$(L$,1,len(L$)-1) : locate 0,Y :
print mid$(L$+space$(80), LP,80); :
locate X,Y : K$=""
135 if asc(K$)<32 then K$="" : rem It's a
control character, ignore it (or add in
your routine here)
139 if K$="" then 115
140 if LT+Y=0 then bell : return
150 if MDE=1 then 160 : rem insert mode
151 if P<=len(L$) then
mid$(L$,LP+X,1)=K$ else
L$=mid$(L$+space$(80),1,P) :
mid$(L$,P,1)=K$
152 locate X,Y : print K$;
153 inc P : inc X : if X>79 then X=79
154 if X=79 then LP=P-79 : locate 0,Y :
print mid$(L$+space$(80),LP,80);
155 goto 115
160 L$=mid$(L$,1,P-1)+K$+mid$(L$,P) :
locate X,Y : print mid$(L$,P,80-xcurs);
161 goto 153
170 rem move 1 place left
171 dec P : dec X : if X<0 then LP=P :
X=0 : locate 0,Y : print
mid$(L$+space$(80),LP,80);
172 if P<1 then P=1 : LP=1 : X=0
173 return
175 rem move 1 place right
176 inc P : inc X : if X>79 then X=79
177 if P>len(L$) then
```

```
L$=mid$(L$+space$(80),1,P)
178 if X=79 then LP=P-79 : locate 0,Y :
print mid$(L$+space$(80),LP,80);
179 return
180 rem Move up a line
181 S=0 : dec Y : if Y<0 then Y=0 : dec
LT : S=1 : if LT<0 then LT=0 : bell : S=0
182 if S=1 then locate 0,Y : scroll down :
locate 0,Y : print
mid$(L$(LT)+space$(80),1,80);
183 locate X,Y : L$=mid$(L$(Y+LT)+
space$(80),1,max(80,len(L$(Y+LT)))) :
return
185 rem Move down a line
```

Keep going – you're nearly there!

```
186 S=0 : inc Y : if Y>22 then Y=22 : inc
LT : S=1 : if LT+22>1000
then dec LT : bell : S=0
187 if S=1 then locate 0,Y : scroll up :
locate 0,Y : print
mid$(L$(LT+Y)+space$(80),1,80);
188 locate X,Y : L$=mid$(L$(Y+LT)+
space$(80),1,max(80,len(L$(Y+LT)))) :
return
200 rem Special keys
201 bell : A=scancode
202 if A=75 then gosub 170
203 if A=77 then gosub 175
204 if A=82 then MDE=1-MDE : set curs
4-MDE*3,5+MDE*3
205 if A=72 then gosub 220 :
L$(LT+Y)=L$ : locate 0,Y : print
mid$(L$+space$(80),1,80); : gosub 180
206 if A=80 then gosub 220 :
L$(LT+Y)=L$ : locate 0,Y : print
mid$(L$+space$(80),1,80); : gosub 185
219 return
220 rem Remove extra spaces from end
of line
221 if right$(L$,1)=" " then
L$=mid$(L$,1,len(L$)-1) : goto221
222 return
500 rem Do File menu
501 on mnselect gosub
505,580,550,570,530
502 goto 55
505 rem Load .ASC file and strip off the
line numbers
510 F$=file select$("\*.ASC","Pick an
.ASC file",1) : if F$="" then return
511 gosub 535
515 open in #1,F$ : L=1 : while eof(#1)=0
520 line input #1,L$(L)
521 if mid$(L$(L),1,1)=" " then
L$(L)=mid$(L$(L),2) : goto 521
522 L$(L)=mid$(L$(L),instr(L$(L)," ")) :
inc L : wend
525 close #1
526 ND=1 : return
```

Save it now just in case !

```
530 end
535 A=1 : repeat : L$(A)="" : L2$(A)="" :
inc A : until A=1001 : return
550 rem Save .ASC file with line
numbers (start on 10, increment by 5)
551 F$=file select$("\*.ASC","Save as
.ASC",1) : if F$="" then return
552 ML=1001 : repeat : dec ML : until
L$(ML)<>""
553 open out #1,F$ : L=0 : while L<=ML
554 L$=L$(L) : if
upper$(mid$(L$,1,9))="PROCEDURE"
then LS=instr(L$,".")
: L$=mid$(L$,LS+1)
555 if upper$(mid$(L$,1,5))="LABEL"
then LS=instr(L$,".") :
L$=mid$(L$,LS+1)
556 print #1,str$(L*5+10)+" "+L$
557 inc L : wend
558 close #1
559 ND=1
560 return
```

Almost finished !

```
570 rem Save .PCP file with labels
571 F$=file select$("\*.PCP","Save as
.PCP",1) : if F$="" then return
572 ML=1001 : repeat : dec ML : until
L$(ML)<>""
573 open out #1,F$ : L=1 : while L<=ML
574 L$=L$(L)
576 print #1,L$
577 inc L : wend
578 close #1
579 ND=1 : return
580 rem Load .PCP file
581 F$=file select$("\*.PCP","Pick a
.PCP file",1) : if F$="" then return
582 gosub 535
585 open in #1,F$ : L=1 : while eof(#1)=0
586 line input #1,L$(L)
587 inc L : wend
588 close #1
```

```
589 ND=1 : return
```

Complex bit coming up !

```
600 ND=1 : cls : print "Converting Labels
to Line numbers"
601 NLBL=0 : NER=0 : ML=1001 : repeat :
dec ML : until L$(ML)<>""
602 print "Pass 1 (Finding Labels)" : A=0
: repeat : if
upper$(mid$(L$(A),1,5))="LABEL" then
LS=instr(L$(A),".") : FS=6 : gosub 650
603 if
upper$(mid$(L$(A),1,9))="PROCEDURE"
then LS=instr(L$(A),".") : FS=10 : gosub
650
605 inc A : until A>ML
606 L2$(A)=L$
610 print "Pass 2 (Replacing Label
references with line numbers . )"
611 A=0 : repeat : L$=L$(A) : L=0 : while
L<NLBL : NI=1
612 if instr(L$,LBL$(L),NI) then gosub
660 : goto 612
613 inc L : wend : L2$(A)=L$ : inc A : lo-
cate 70,0 : print A;" " : locate 79,0 : print
T$(TGL) : inc TGL : TGL=TGL mod 4 :
until A>ML
620 print "Pass 3 (Save it to disk as an
.ASC file)"
621 F$=file select$("\*.ASC","Save as
.ASC",1) : if F$="" then return
622 open out #1,F$ : L=0 : while L<=ML :
L$=L2$(L)
623 if
upper$(mid$(L$,1,9))="PROCEDURE"
then LS=instr(L$,".") : L$=mid$(L$,LS+1)
624 if upper$(mid$(L$,1,5))="LABEL"
then LS=instr(L$,".") : L$=mid$(L$,LS+1)
625 if mid$(L$,1,1)=" " then
L$=mid$(L$,2) : goto 625
627 if L$<>"" then print #1,str$(L*5+10)+"
"+L$
628 inc L : wend
629 close #1 : ND=1 : goto 55
649 return
650 rem Sort out label/procedure name
651 LBL$=mid$(L$(A),FS,LS-FS+1)
652 if mid$(LBL$,1,1)=" " then
LBL$=mid$(LBL$,2) : goto 652
653 if LBL$="." or LBL$="" then print
"Label naming error in line ";A :
ERJ(NER)=A : bell : wait key
654 LBL$(NLBL)=LBL$ : LBL(NLBL)=A :
inc NLBL : return
660 rem replace label with line number
661 if NI=1 and
upper$(mid$(L$,1,5))="LABEL" then
NI=instr(L$,".")+1: return
662 if NI=1 and
upper$(mid$(L$,1,9))="PROCEDURE"
then NI=instr(L$,".")+1: return
663 rem Check for parity of quotes
(Stops renaming stuff in strings!)
664 Q$=chr$(34) : QC=0 : Q=1 : repeat : if
mid$(L$,Q,1)=Q$ then inc QC
665 inc Q : until Q>=instr(L$,LBL$(L)) : if
QC/2*2<>QC then NI=instr(L$,LBL$(L))+1
: return
666 I=instr(L$,LBL$(L)) : L$=mid$(L$,1,I-
1)+str$(10+LBL(L)*5)
+mid$(L$,I+len(LBL$(L))) : return
669 return
700 rem Print listing
701 ML=1001 : repeat : dec ML : until
L$(ML)=""
702 A=0:repeat : lprint L$(A) : inc A :
until A>ML
703 goto 55
```

Well done! Go and make a cup of tea!

Using MIDI

And now for something for those people who have been trying to use MIDI with STOS. The program below is designed to allow two STs to communicate via MIDI's System Exclusive commands. Before going into detail on the program itself I shall clear up a few things about MIDI.

- MIDI isn't purely for musical communication. It is basically just another communications line, like the RS232.
- The MIDI standard is a set of rules for anything that wants to use MIDI to communicate. This includes the wiring of the leads and what data you are supposed to send. If you only want to communicate between two or more computers, then as long as you have the leads correctly wired, you can completely ignore the MIDI standard control codes.
- STOS's MIDI control is based on a file system. This is not the best system to use, but it does work as long as you are careful with your pro-

gramming.

The first thing you must do if you are using MIDI commands with STOS is to open a channel for the data. This is done using the Open #chan,"MIDI": command. This channel number has nothing at all to do with MIDI Channel numbers – it is just the data channel that STOS assigns. I usually use 3 (as 3 is the number of the MIDI line for all the ST's BDOS,BIOS and XBIOS routines). Once the line is opened, you can use print # statements to send data out, or the =port(#) function to receive data. Each MIDI byte is represented as 1 character when outputing, and the =port function will return 1 byte, or a -1 if there is no data present. When outputing MIDI data using print # make sure you use a semicolon at the end of the print, otherwise STOS will send a return character and character 20, thus messing up your output to your synth or computer. So use:

```
print #3,mo$;
```

instead of:

```
print #3,mo$
```

To input MIDI data you use the =Port(#) function. This pulls the next MIDI byte off the data buffer and returns it. It also returns a -1 if there is nothing there. So to wait for an incoming byte, you must constantly check the buffer for data, all the time storing the data in a variable. A while...wend loop is best for this, for example:

```
I=-1 : While I<0:I=port(#3): wend
```

This loop will exit with the data byte in the variable I. Obviously for longer messages you must use this loop more than once. You may also want to use a timer on the loop to exit if there is no data within a given time. For example:

```
I=-1:timer=0:while I<0 and
timer<500:I=port(#3):wend
```

This will exit with I set at -1 if it takes too long to get the byte, thus saving the computer from locking up if there is no incoming data. At the start of your program, or before sending request messages to synthesisers for data (see next issue), it is a good idea to clear the input buffer of rubbish. This, again, is easily achieved by using a while...wend loop.

```
I=0:while I>=0:I=port(#3):wend
```

The program below uses a special variation of this routine. It clears any useless data until it comes to the welcome message that the other ST has sent. This tells the computer that the other ST is on line and waiting to communicate. The other ST does the same of course. To use, type in the program and copy it across to a second ST which should be connected MIDI Out ST1 to MIDI In ST2 and MIDI Out ST2 to MIDI In ST1. Make sure the program is typed in correctly on **both** machines and run it. If you have made a mistake in the program, or the connection is incorrect, delete the rem on line 25, run the program and put the rem back. This completely clears the MIDI line, to make sure there are no spare welcome messages floating around. When both ST's have started talking to each other, you will get a welcome message from the other ST and then you can enter a message to send to it. Enter your message and press Return. You will then see the message that was sent to you. You continue entering messages until you get bored, at which point enter QUIT (or quit) to log off. Your ST will inform the other that you have logged off, and both programs will stop. This program actually uses MIDI System Exclusive commands to send the data. This means that if you happen to have a synthesiser still connected to either ST when you use the program, it will happily ignore it, as I have used the Fairlight ID code for the message. (If you happen to have a Fairlight CMI then (a) make sure it isn't connected and (b) phone me, because I'm desperately in need of a job!) The format for a system exclusive message is as follows:

$F0	This is the System Exclusive command
$nn	Manufacturers ID code (we are using $14). This makes sure that only a synth

from the specified manufacturer will take any notice of the data.

Your data Depending upon what you are sending the data to, it can be almost anything. However all the data bytes (and the Manufacturers ID must be in the range 0-127, as the MIDI data format reads anything 128-255 as a command (more next issue). For this program the data is your message

$F7 This is the EOX byte (End Of eXclusive) that tells the receiving instrument that the system exclusive message is over.

```
10 rem MIDI Communication between
two ST's
15 WELC$="Hello from another ST !"
20 open #3,"MIDI" : rem Open the MIDI
Line
25 rem gosub 300:end : rem Remove the
rem to clear the line of welcome data
from bugged attempts at this program .
30 gosub 200 : rem Clear the line of
garbage and send Welcome message
40 line input "Enter Message :";M$
43 if upper$(M$)="QUIT" then M$="Other
user has logged off"
45 for A=1 to len(M$) :
mid$(M$,A,1)=chr$(min(127,asc(mid$(M$,A,1))))
: next A : rem Make sure no bytes are
above 127
50
MO$=chr$($F0)+chr$($14)+M$+chr$($F7)
: rem Add (MIDI System Exclusive)+
(Fairlight I.D !)+message+(EOX End Of
eXclusive) data
60 gosub 100 : rem Send it
65 if M$="Other user has logged off"
then end
70 gosub 150 : rem Get data from other
computer
80 M$=mid$(MI$,3,len(MI$)-3) : rem
Remove header and EOX data
90 print M$
93 if M$="Other user has logged off"
then end
95 goto 40
99 close #3 : end
100 rem Send MIDI Message
105 print #3,MO$;
110 return
150 rem Get MIDI Message (terminated
by EOX Character)
155 I=-1 : timer=0 : while I<0 : I=port(#3)
: wend
160 if I<0 then print "Nothing On Line " :
close #3 : end
165 MI$=chr$(I)
170 I=-1 : while I<0 : I=port(#3) : wend
175 if I<0 then print "MIDI Line Error !" :
close #3 : end
180 MI$=MI$+chr$(I) : if I<>$F7 then 170
: rem Check for EOX Character If not
then get another byte
185 return
200 rem Send welcome message, clear
the line and wait for incoming welcome
205 MO$=chr$($F0)+chr$($14)
+WELC$+chr$($F7) : gosub 100 : rem
Send welcome
210 I=0 : while I>=0 and I<>$F0 :
I=port(#3) : wend : if I<0
and mid$(MI$,3,len(MI$)-3)<>WELC$
then 210
211 if I=$F0 then gosub 250 : rem Check
for welcome message
212 if I>=0 then 210
213 return
250 MI$=chr$($F0) : rem Check for wel-
come message
251 I=-1 : timer=0 : while I<0 and
timer<200 : I=port(#3) : wend
252 if I<0 then I=0 : return : rem Not the
welcome message so keep checking
253 MI$=MI$+chr$(I)
254 if I=247 then 260
255 goto 251
260 if mid$(MI$,3,len(MI$)-3)=WELC$
then print WELC$ : I=-1 : return
261 I=1 : return
300 rem Clear line without welcome (Do
this when line has garbage on it
due to bad programs)
301 I=0 : while I>=0 : I=port(#3) : wend :
return
```

Have fun with this program and feel free to modify it. You can use a similar system in a game to link two ST's together for multi-player games (Battleships is the best one to start on). It is possible to use more than two ST's together in a network (see my Jitterbugs game on the ST Amiga format cover disc) although this requires slightly trickier routines (They might be revealed

in a couple of issues time!).

Scrolling Text

To answer the request for a scrolling text routine, here is a small program written with PCP (Just to get you to type PCP in!) that will scroll your message horizontally across the screen in large letters.

```
rem created with PCP
rem ********************************
rem *
rem * Message Scroller Program
rem * Written in PCP STOS by
rem * Aaron Fothergill
rem * Shadow Software 1989
rem *
rem ********************************
mode 1 : key off : line input "Enter your
    Message ";MES$ :  MES$=MES$+" " :
    rem Enter a message and stick a space
    on the end
mode 0 : key off : curs off : hide on :
    click off : rem Set to low resolution
auto back off : reserve as screen 5 :
    logic=5 : wait vbl : ink 0 :  bar 0,0 to
    319,199 : rem Reserve ourselves a
    screen and clear it
paper 0 : pen 1 : RTE=4 : rem Set the
    text colours and scroll rate
locate 0,0 : logic=5 : print left$(MES$,10)
    : MES$=mid$(MES$,11)+left$(MES$,10) :
    rem Put the text on screen 5
wait vbl : zoom 5,0,0,80,8 to
    back,0,64,319,127 : logic=back :  bar 0,0
    to 80,8 : wait vbl : screen copy
    back,0,64,320,128 to  physic,0,64 : rem
    Zoom the text onto the main screen
LABEL loop1. SCR=0 : logic=back : rem
    When SCR gets to 64 it's time to draw
    the next letter
LABEL loop2. logic=back : screen swap :
    wait vbl :  screen copy back,RTE,64,
    320,128 to back,0,64 : bar 320-RTE,64
    to 319,128 : rem Scroll the text and
    blank the end
SCR=SCR+RTE : if SCR<64 then loop2. :
    rem I know repeat...until  is nicer, but
    I'm testing PCP
rem Draw the next
    letter
back=5 :
    logic=5 :
    wait vbl : locate 0,0 : print left$(MES$,1) :
    MES$=mid$(MES$,2)+left$(MES$,1) :
    back=default back :  rem Print next
    character on our screen and scroll the
    characters
zoom 5,0,0,8,8 to back,288,64,319,127 :
    rem Zoom it to the background  screen
screen copy back,0,64,320,128 to
    physic,0,64 : rem Copy it to the  physi-
    cal screen
goto loop1.: rem Go back and scroll it
    forever !
```

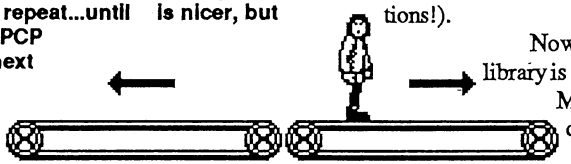
Note in PCP you can enter as many characters as you want on one line. In the above program only start a new line when the line actually starts at the left of the page, the indented bits are to be typed as part of the same line. A simpler way of scrolling text is to store the text in a string and then to stick one end of the string on the other end. For example:

```
MES$="This is a message over 20
    characters long that will scroll "
LABEL loop. MES$=mid$(MES$,2)
    +left$(MES$,1)
locate 0,10:centre mid$(MES$,20) goto
    loop.
```

This will scroll the letters left in the centre of the screen. To change the number of characters to be seen at a time, just change the 20 in the centre mid$ bit. To make the text scroll to the right, use the line:

```
LABEL loop. MES$=right$(MES$,1)
    +left$(MES$,len(MES$)-1)
```

Experiment with these routines and see what you can come up with. It's an easy way to do an intro to a utility (as it uses very little memory) and you could try doing multiple messages scrolling in different directions. (The best so far has been 24 lines of text scrolling in alternate directions!).



Now in the STOS User Club library is a demo of TOME (TOtal Map Editor) a little machine code routine of

mine that allows you to do Gauntlet-style mapping games at high speed. The demo disk contains a demo version of the editor (which times out after 10 minutes and doesn't let you save maps) and a small demo program incorporating two-level parallax scrolling. You can use TOME in your own programs, and it will soon be available as a STOS extension file. The demo disk doesn't let you use TOME in your programs, but it lets you get a good idea of what it can do! Here are some of its features:

- High Speed Map Drawing and scrolling
- 240 tiles per tile screen
- multiple tile screens useable
- very efficient map data system. You can store a map the size of 240 ST screens within 64k!
- Easy to edit maps with the TOME Map Editor (Includes Cut and Paste and Locator functions).
- TOME also works with Hi-res (not perfect yet, but getting there!) To get the Demo send a disk and handling fee to Pat Winstanley at the STOS user group. If you want an early version of TOME, details are on the demo disk of how to obtain it. Although I suggest you wait a month until I've perfected it and put the STOS version of Jitterbugs on the disk (that's the one on ST Amiga Format #9). The STOS version of Jitterbugs is going to be somewhat improved over the original GFA basic version (STOS can handle the routines better so it will be faster, have more varied aliens, bigger maps, more traps and much much more!) Coming soon: Language converter update for PCP and Invisible Galaxians (ha ha).

If you need help with your STOS programming or know any good jokes, ring me on 0271 816037 anytime after 1pm and before 7pm (this number may change from October 1), or write to me (or Adam Fothergill who is the artwork expert) at our new address: Barnstaple, North Devon

preferably with a large order and blank cheque for our programs, but we do answer questions and provide help for free. Please include an SAE if at all possible. ■

This STOS Newsletter was edited by Pat Winstanley. Layout by Chris Payne.

10-LINERS

Missile Command comes to STOS in just 10 lines with BLOCKER! In this excellent little game by the ubiquitous Aaron (look it up!) your lawn on Tau Ceti 3 is being attacked by Stringy stuff. Ward it off with a super-duper aerosol controlled with the mouse, but use it wisely as it runs out with overuse. On the later levels a fog will come down, but you can disperse this with the spray.

You will need sprites for your gardener. Use ALIENS5.MBK from STOS Sprites 600 or use six animations of a circular flying saucer (or six movements of the same sprite).

```
10 mode 0 : key off : click off : curs off :
dim MX(20),MY(20),MXO(20),MYO(20) :
LVL=1 : DEAD=0 : SCRE=0 : change
mouse 4
20 X=hunt(start(1) to start(1)+length(1),
"PALT")+4 : for A=0 to 15 : colour
A,deek(X) : X=X+2 : next A : repeat : ink
0+LVL/5 : bar 0,0 to 319,199 : ink 8 : for
A=0 to 319 : draw A,199 to A,180-rnd(20)
: next A : for A=0 to LVL : MX(A)=160-
rnd(100)+rnd(100) : MY(A)=0 :
MXO(A)=MX(A) : MYO(A)=0 : next A :
colour 4,$237
30 NM=LVL : BNX=0 : AMMO=50+LVL*30
: timer=0 : while timer<1000+LVL*500 :
X=x mouse : Y=y mouse : K=mouse key
: for A=0 to NM : if DEAD=0 then ink 1+A
mod 7 : draw MXO(A),MYO(A) to
MX(A),MY(A) : MXO(A)=MX(A) :
MYO(A)=MY(A) : if point(MX(A),MY(A)
+1)=8 then ink 0 : epie MX(A),MY(A),20
+LVL,20+LVL*4,0,3600 : MX(A)=160+
rnd(140)-rnd(140) : MY(A)=0 :
MXO(A)=MX(A) : MYO(A)=0 :
SCRE=SCRE+LVL*10 : boom
40 MX(A)=max(0,min(MX(A)+rnd(4)-
rnd(4),319)) : MY(A)=MY(A)+LVL/4+1 : if
MY(A)>199 then DEAD=1 : timer=9999 :
A=99 : boom
50 next A : if K and AMMO>0 then dec
AMMO : ink 8 : pie X,Y,10,0,3600 : shoot
```

(continued on page 22)

Writing a complete game using STOS

Ralph Effemey, author of STOS Paint, Arthur of the Britons and various public domain games shows you how to create a game from scratch.

Everyone who has used STOS will, by now, have realised how easy it is to write games to commercial standard. "It's not easy!", did you say? Of course it is – and over the next few issues I'm going to show you how.

Scrolling seems to be one of the sticking points for a lot of you so let's jump in at the deep end and write a game that uses vertical scrolling (this being smoother than horizontal scrolling). I don't know about you, but personally I'm fed up with the usual 'shoot-'em-up'-type games, so why not introduce an element of stick control and not just fast reaction on the fire button.

I'll call the game Race Way – you've guessed it, it's a rally/race game. The aim is to drive your car over a series of sections ranging from forest, through mountain and desert sections and finishing in a town. You have a number of laps (I've used eight) per section and your running time and score is displayed at the end of each section with your total score displayed at the end of the game. The game also takes into account the number of crashes, deducting points for each crash, eliminating you if you have too many.

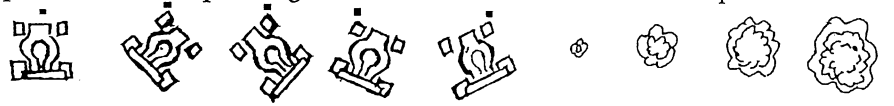
So – on with the program. Using Dégas, Neochrome or STOS Paint you need to draw an opening title screen and 20 track screens (five screens per section) but at this stage you can make them as simple as you like. The track which the car runs on **must** be drawn in colour 13. Everything else (trees, kerbs, houses etc) **must** be drawn in any colour other than colour 13 – you'll see why later. You will also need to design your car in five positions as sprites 1 to 5 and an explosion as sprites 6 to 9 (see the illustrations below). These are my designs, but you can use any design you like as long as they are no bigger and the hot spot is in the same place. Finally, the game scrolls approximately two thirds of the screen while the remaining right hand third remains static – this is where your score, lap counter and time are displayed. These figures are displayed using windows and must be at the positions indicated.

As each section comprises five screens, all the screens you design for that section must link up i.e. screen one must join screen two and so on. That way, when you scroll, you will not see the join.

The short program below will help you create a section of five screens. It will draw the first straight for you, then you take over and finish the first course by holding down the left mouse button and drawing in the normal way or moving the pointer to a new position and then pressing the left mouse button to draw a straight line to the new position. Do not draw a track that is more than 45 degrees away from the vertical in either direction. When you have finished the track, press the right mouse button and both the track

Hot spots to be in the centre one pixel line above the sprite image

Hot spots for the explosion must be in the **centre** of the sprite



and surrounding area will fill in. (This will only work if your pointer is right at the bottom of the screen.)

You can then add 'trees' (polymarkers) if you so desire. Pressing the right button will exit to the save screen window. I suggest you name your tracks TRACK1.PI1, TRACK2.PI2 and so on. On the final screen you will see two small points at the bottom. You must join your track to these points to enable a smooth scroll with no join!

After five tracks, quit the program and start again at TRACK6.PI1 and so on, and continue until you have drawn all 20 tracks. Then use the compact accessory to compact the pictures ready for the game itself.

```
10 rem ********** SCREEN DESIGNER
20 rem ********** BY RALPH EFFEMEY
30 rem *** EVERYTHING OFF!
40 key off : mode 0 : curs off : flash off :
reserve as screen 6
50 gosub 230
60 X1=85 : X2=85 : ink 13 : set paint
1,1,1 : TRACK=1 : colour 13,$555
70 rem ******* DRAW ROAD
80 if TRACK=1 then for I=1 to 130 : plot
X1,Y1 : plot X1+35,Y1 : Y1=Y1+1 : next I
90 if mouse key=1 then X2=x mouse :
Y2=y mouse : draw X1,Y1 to X2,Y2 :
draw X1+3
5,Y1 to X2+35,Y2 : X1=X2 : Y1=Y2
100 if mouse key=2 and Y1=199 then
bell : ink 13 : paint X2+10,Y2 : ink 5 :
pain
t 1,1 : paint 224,198 : goto 130
110 goto 90
120 rem ********* DRAW 'TREES'
130 ink 7 : set mark 3,39
140 if mouse key=1 then polymark x
mouse,y mouse
150 if mouse key=2 then goto 170
160 goto 140
170 rem ********** SAVE SCREEN
180 hide on : screen copy physic to 6 :
show on
190 F$=file select$("*.PI1","      SAVE
AS DEGAS SCREEN")
200 if F$="" then screen copy 6 to
physic : screen copy 6 to back : show
on : end
210 if right$(F$,3)<>"PI1" then goto 190
220 save F$,6 : inc TRACK : Y1=0 : Y2=0
: cls physic : cls back : gosub 230 : in
k 13 : goto 90
230 paper 13 : ink 13 : bar 227,0 to
319,199 : ink 5 : draw 225,0 to 225,199 : p
en 1 : bar 235,150 to 311,191
240 locate 30,3 : print "SCORE" : locate
30,7 : print "LAPS" : locate 30,11 : pr
int "TIME" : paper 5 : locate 32,20 : print
"RACE" : locate 33,21 : print "WAY"
: if TRACK=5 then ink 13 : plot 85,199 :
plot 120,199
250 return
```

The game itself is listed below. You will need a title screen which you must save in bank 6 and you will need to design your own car as also discussed above and load this into the sprite bank.

Now for the listing explanations:

Lines 10 - 110 set up the title screen, windows and so on, initialise the variables and numerous other house-keeping duties.

Lines 120 - 230 is the main loop which checks the joystick, scrolls the screen via a sub-routine at line 240, detects collisions and counts the crashes.

As mentioned, the scroll routine starts at line 240 and thanks must go to Richard Vanner for these lines of code as they allow a scrolling screen and moving sprites!

Lines 290 - 330 moves a new car onto the track after a crash. This sub-routine simply checks for the colour of the track. If the new car has reached the track then the FOR...NEXT loop is terminated. The car is then moved 10 pixels further to the right to ensure the car is in the middle of the track.

Lines 340 - 360 checks the number of laps completed. If L=0 then increment the track variable (TR) and load in the next set of tracks.

Lines 370-440 is the end game screen which shows you your current time, score and the fastest time so far.

Lines 450 - 560 produce what I call the 'earthquake' effect. Rather than just have the car explode, I thought it would be nice to have the whole track shake as well. This routine does just that. It might be simple, but it works!

Lines 570 - 590 checks the number of crashes. I've chosen 12 - you can change this by re-setting the value of CR in line 70 and 220. You might like to introduce a difficulty factor by altering the number of crashes you can receive before having to retire. Make CR player variable.

The remainder of the code simply loads the four track sequences in turn. Now for some of the most important variables:

SC = Score
TT = Number of tracks
CR = Number of crashes
FTT = Fastest overall time (The 100000 in line 20 is dummy value)
SP = Speed
U = Vertical position of car
X = Horizontal position of car
L = Number of laps (change this if you like)

You will notice that in lines 720, 840 and 960 I've named each section. You can, of course, change this to suit your own design of track.

This is really quite a simple game but it uses a lot of STOS features, especially the scrolling effect. I hope you find it useful and I'm sure you can improve on it. Try using STOS Paint to produce your screens. It's miles better than the utility I provide above.

Try adding other sprites, maybe tanks or planes that try and shoot you off the road. The possibilities are endless.

If you don't compact the screens you might not be able to get them all onto a single-sided disc. If you use a double-sided disc and don't want to be bothered with packing and unpacking your screens then simply remove all UNPACK commands. You can find these with the SEARCH command.

```
10 fade 3 : wait 21 : mode 0 : flash off:
scroll off : curs off : key off : hide on :
screen copy 6 to physic : screen copy 6
to back : for I=10 to 15 : erase I : next I :
get palette (6) : wait key : windopen
4,3,5,24,1,0,3
20 gosub 600 : screen copy 10 to physic
: screen copy 10 to back : get palette
(10) : SC=0 : SP=0 : FTT=100000 :
volume 16 : click off
30 Y=1 : X1=0 : X2=225 : logic=back :
U=130 : S=10 : X=95 : paper 14 : pen 1 :
limit sprite 0,0 to 225,200
40 windopen 1,30,2,5,1,0,3 : curs off :
scroll off
50 windopen 2,30,4,5,1,0,3 : curs off :
scroll off
60 windopen 3,30,6,7,1,0,3 : curs off :
scroll off
70 SC=0 : TT=0 : TR=0 : CR=0
80 get palette (10) : qwindow 1 : print SC
: logic=physic : screen swap : print SC :
logic=back : screen swap
90 L=6 : qwindow 2 : print L :
logic=physic : screen swap : print L :
logic=back\: screen swap
100 qwindow 3 : print TT/50 :
logic=physic : screen swap : print TT/50
: logic=back : screen swap
110 gosub 250 : sprite 1,X,U,1 : gosub
250 : timer=0 : SP=0
120 rem ******* MAIN LOOP ************
130 gosub 240
140 sprite 1,X,U,1
150 if jright then X=X+(SP/2) : sprite
1,X,U,2
160 if jleft then X=X-(SP/2) : sprite
1,X,U,3
170 if jup and jright then X=X-(SP/4) :
sprite 1,X,U,4
180 if jup and jleft then X=X+(SP/4) :
sprite 1,X,U,5
190 if fire then inc SP : if SP>20 then
SP=20
200 if jdown then dec SP : if SP<2 then
SP=2
210 C=0 : C=detect(1) : if C<>13 then
boom : gosub 460 : sprite 1,-100,-100,1 :
gosub 250 : gosub 300
220 if CR>12 then goto 580
230 goto 130
240 rem ****** SCROLL *******
250 volume 16 : noise 1 : envel 12,48-SP
: Y=Y+SP/2 : if Y>200 then Y=1 : inc S :
screen copy S,0,0,225,200 to 10,0,0 : if
S>14 then S=10 : gosub 350
260 screen copy S+1,X1,200-Y,X2,200 to
logic,0,0
270 screen copy S,X1,0,X2,200-Y to
logic,0,Y
280 redraw : screen swap : wait vbl :
return
290 rem ******* MOVE CAR ONTO
```

```
TRACK ******
300 sprite 1,-100,U,1 : SP=0 : gosub 250
310 gosub 250 : inc CR : logic=physic :
screen swap : screen copy physic to
back : for I=0 to 240 : wait 1 : sprite
1,I,U,1 : C=0 : C=detect(1) : if C=13 then
I=240
320 next I
330 C=0 : for J=x sprite(1) to x
sprite(1)+10 : inc J : X=J : sprite 1,X,U,1 :
wait 2 : next J : sprite 1,X,U,1 :
logic=back : screen swap : return
340 rem ****** CHECK LAPS COM-
PLETED ********
350 volume 0 : bell : dec L : qwindow 2 :
print L;" " : logic=physic : screen swap :
print L;" " : logic=back : screen swap : if
L=0 then inc TR : on TR goto
680,800,920,380
360 return
370 rem ****** END GAME SCREEN
*************
380 SC=10000-(timer/10) : SC=SC-
(CR*50) : CR=0 : TT=timer
390 qwindow 1 : print SC : logic=physic :
screen swap : print SC : logic=back :
screen swap
400 qwindow 3 : print TT/50 :
logic=physic : screen swap : print TT/50 :
logic=back : screen swap : if HS<SC
then HS=SC
410 screen copy 6 to back : screen copy
6 to physic : logic=physic : screen swap
: if TT<FTT then FTT=TT
420 windopen 6,5,5,22,7,1,1 : curs off :
scroll off : print " HIGH score=";HS :
print " YOUR score=";SC : print
"FASTEST TIME=";FTT/50;"" : cdown :
print " Any key to go" : sprite 1,-100,100
430 wait key : clear key : windel 6 :
screen copy 6 to back : screen copy 6 to
physic : for I=10 to 15 : erase I : next I :
gosub 600 : fade 3 : wait 21 : screen
copy 10 to physic : screen copy 10 to
back : logic=back : screen swap : get
palette (15)
440 pop : goto 70
450 rem ****** 'EARTHQUAKE' SCREEN
*********
460 redraw : screen swap : sprite 1,x
sprite(1),U,6 : wait 1
470 redraw : screen swap : sprite 1,x
sprite(1),U,7 : wait 2
480 redraw : screen swap : sprite 1,x
sprite(1),U,8 : wait 3
490 redraw : screen swap : sprite 1,x
sprite(1),U,9 : wait 4
500 redraw : screen swap : sprite 1,x
sprite(1),U,8 : wait 5
510 redraw : screen swap : sprite 1,x
sprite(1),U,7 : wait 6
520 redraw : screen swap : sprite 1,x
sprite(1),U,7 : wait 6
530 redraw : screen swap : sprite 1,x
sprite(1),U,6 : wait 7
540 redraw : screen swap : sprite 1,x
sprite(1),U,6 : wait 7
550 redraw : screen swap : sprite 1,x
sprite(1),U,6 : wait 7
560 return
570 rem ****** TOO MANY CRASHES
SCREEN ******
580 boom : logic=physic : screen swap :
windopen 5,3,5,24,2,0,3 : curs off : scroll
off : print " TOO MANY CRASHES!!" :
print "Any key to go again..."
590 wait key : fade 3 : wait 21 : windel 5 :
windel 3 : windel 2 : windel 1 : for I=10 to
15 : erase I : next I : screen copy 6 to
back : screen copy 6 to physic : goto 20
600 rem **** LOAD FIRST TRACK ******
610 logic=physic : screen swap : curs off
: scroll off : qwindow 4 : print "LOADING
1st SECTION"
620 reserve as datascreen 10 : reserve
as datascreen 15 : load "track1.mbk",5 :
unpack 5,10 : unpack 5,15
630 reserve as datascreen 11 : load
"track2.mbk",5 : unpack 5,11
640 reserve as datascreen 12 : load
"track3.mbk",5 : unpack 5,12
650 reserve as datascreen 13 : load
"track4.mbk",5 : unpack 5,13
660 reserve as datascreen 14 : load
"track5.mbk",5 : unpack 5,14
670 return
680 rem **** LOAD SECOND TRACK
******
690 SC=10000-(timer/10) : SC=SC-
(CR*50) : CR=0 : TT=timer
700 qwindow 1 : print SC : logic=physic :
screen swap : print SC : logic=back :
screen swap
710 qwindow 3 : print TT/50 :
logic=physic : screen swap : print TT/50 :
logic=back : screen swap
720 logic=physic : screen swap :
qwindow 4 : curs off : scroll off : print
"Loading MOUNTAIN section"
730 for I=10 to 15 : erase I : next I :
```

```
reserve as datascreen 10 : load
"track6.mbk",5 : unpack 5,10
740 reserve as datascreen 11 : load
"track7.mbk",5 : unpack 5,11
750 reserve as datascreen 12 : load
"track8.mbk",5 : unpack 5,12
760 reserve as datascreen 13 : load
"track9.mbk",5 : unpack 5,13
770 reserve as datascreen 14 : load
"track10.mbk",5 : unpack 5,14
780 reserve as datascreen 15 : load
"track6.mbk",5 : unpack 5,15
790 fade 3 : wait 21 : timer=TT : goto 80
800 rem **** LOAD THIRD TRACK *******
810 SC=10000-(timer/10) : SC=SC-
(CR*50) : CR=0 : TT=timer
820 qwindow 1 : print SC : logic=physic
: screen swap : print SC : logic=back :
screen swap
830 qwindow 3 : print TT/50 :
logic=physic : screen swap : print TT/50
: logic=back : screen swap
840 logic=physic : screen swap :
qwindow 4 : curs off : scroll off : print
"Loading DESERT section"
850 for I=10 to 15 : erase I : next I :
reserve as datascreen 10 : load
"track11.mbk",5 : unpack 5,10
860 reserve as datascreen 11 : load
"track12.mbk",5 : unpack 5,11
870 reserve as datascreen 12 : load
"track13.mbk",5 : unpack 5,12
880 reserve as datascreen 13 : load
"track14.mbk",5 : unpack 5,13
890 reserve as datascreen 14 : load
"track15.mbk",5 : unpack 5,14
900 reserve as datascreen 15 : load
"track11.mbk",5 : unpack 5,15
910 timer=TT : goto 80
920 rem **** LOAD FOURTH TRACK
930 SC=10000-(timer/10) : SC=SC-
(CR*50) : CR=0 : TT=timer
940 qwindow 1 : print SC : logic=physic
: screen swap : print SC : logic=back :
screen swap
950 qwindow 3 : print TT/50 :
logic=physic : screen swap : print TT/50
: logic=back : screen swap
960 logic=physic : screen swap :
qwindow 4 : curs off : scroll off : print
"Loading TOWN section"
970 for I=10 to 15 : erase I : next I :
reserve as datascreen 10 : load
"track16.mbk",5 : unpack 5,10
980 reserve as datascreen 11 : load
"track17.mbk",5 : unpack 5,11
990 reserve as datascreen 12 : load
"track18.mbk",5 : unpack 5,12
1000 reserve as datascreen 13 : load
"track19.mbk",5 : unpack 5,13
1010 reserve as datascreen 14 : load
"track20.mbk",5 : unpack 5,14
1020 reserve as datascreen 15 : load
"track16.mbk",5 : unpack 5,15
1030 timer=TT : goto 80
```

■

(BLOCKER continued...)

```
60 if BNX>0 and BNX<339 then sprite
2,BNX,BNY,BNS : update : inc BNS :
BNS=1+(BNS-1) mod 6 : BNX=BNX+4 : if
detect(2)=8 then SCRE=SCRE+200 :
volume 15 : for A=90 to 5 step-1 : play
A,0 : next A : timer=timer+100 : ink 0 :
pie BNX,BNY,40,0,3600 : BNX=0 :
volume 16 : envel 9,1000 : sprite 2,999,1
70 if (BNX=0 or BNX>339) and
rnd(100)=1 then BNX=1 :
BNY=rnd(20)+20 : BNS=1
80 if LVL>4 and NM<LVL+4 then
S=rnd(NM) : inc NM : MX(NM)=MX(S) :
MXO(NM)=MXO(S)
 : MY(NM)=MY(S) : MYO(NM)=MYO(S)
90 wend : if DEAD=0 then inc LVL :
SCRE=SCRE+1000*LVL : locate 0,10 :
paper 5 : pen 1 : centre "LEVEL
"+str$(LVL) : locate 0,12 : centre "Hit
Anything" : while inkey$="" and mouse
key=0 : wend
100 until DEAD=1 : paper 0 : clw : locate
0,10 : paper 0 : pen 8 : centre "GAME
OVER" : paper 1 : pen 5 : locate 0,12 :
centre "You scored "+str$(SCRE) : while
inkey$="" : wend : run
```

The senders of the best 10-liner game and the best 10-liner routine or non-game will each receive a copy of the two disc version of Skystrike Plus. For the purpose of this competition, you cannot use machine code and you cannot call an external file greater than 10k in length.

Please ensure that you include an SAE if you want your disc returned – we'll send it back crammed other goodies! ■

The STOS range just keeps on growing!

A brand new range of STOS add-ons will be released this Autumn – and Mandarin Software wants to ensure that the titles are exactly what you're looking for. Take a look at the goodies below, and if you have any suggestions to make then contact Christopher Payne at Mandarin Software. If your ideas are taken up you will receive free software of your choice.

STOS Musician

- Icon driven music editor: Drop notes onto the stave; see all three voices at once (distinguished by colour); MIDI input; multiple repeats
- More than 100 different tunes with a wide range of styles – from full-length songs to short jingles
- Animated jukebox facility to play all or selected songs

Price: £14.95. Available November 14

STOS Gamespack

- Four of the very best games submitted for the recent £5,000 Awards
- Titles include Skate Tribe (fast vertical-scrolling skateboard game), Mouthtrap (chomp the fruit with the animated teeth while avoiding the monsters (not PacMan!)) and Skystrike (take to the air in this Spitfire dogfight game)
- Each one has been compiled for maximum speed and enjoyment – these really are up to commercial standard.
- Both discs has a special format so that if you have a double-sided drive you can access the original STOS Basic files, examine the listings, grab routines, or modify the games for your own amusement.
- If you have a single-sided drive you can send for the Basic files for just £2.

Price: £19.95 Available October 31.

STOS Vidi Digitiser

- Grab frames from video in real time – up to 25 frames a second
- Define a window on the screen to select images – or select miniaturised versions of the full picture down to 1/16th size (as in the Phantom of the Opera demo)
- Store frames in banks
- Modifiable editor written in STOS Basic
- Hardware and most software supplied by Rombo Productions
- Grab characters like Scooby Doo into the sprite bank for using in your games

Price: £99.95 Available December

STOS 3D

- Design you own 3D objects using a powerful editor accessory written in STOS
- Add detail to the sides of objects: Simple lettering, portholes, coloured bands
- Add animation so you can have a flapping robot bird
- Use the brand new STOS commands to fly around or through holes in objects and view from any angle
- Full collision detection so you can blast ships out of the skies or crash into them. You can even blast holes in wings and see through the gaping gap!
- Fully documented so you can put STOS 3D to work immediately
- Includes demonstration game, example programs and pre-defined objects
- Written over the last 18 months by 3D experts who are putting the finishing touches to a 3D megagame to be released in spring 1990 on ST, Amiga and PC.

Available in spring 1990

If you have any music suitable for STOS Musician, any top-notch games for future compilations, or suggested improvements to STOS Compiler or STOS Maestro, get in touch with Mandarin **now**!

Speeding up STOS

Stephen Hill takes a close look at the Autoback command and comes up with some surprising conclusions

After a whole year's worth of deafening silence, the lunatic who wrote the STOS Basic manual has finally been persuaded to speak out. Actually I've been rather busy over the past few months, as I've been working night and day on my forthcoming new book. Hopefully it should answer at least a couple of the more pressing questions you've been airing in the STOS Newsletter. Such as how the heck do you write a game in STOS Basic?

There's also a terrific section on the STOS extension system. Suffice it to say, I'll be supporting this by providing real help to all you budding extension programmers. I'm already encouraged by the excellent work by people like David Thomson, who cleverly converted Fast Basic's speech system for use with STOS Basic. Hopefully the STOS basic story will be entering a new phase, with the active participation of the users in the development of the language. Drop me a line via Mandarin if you have any ideas – I'd be delighted to hear from you.

Well, that's the plug over with. Now for the interesting bit. While I was working on the book, I discovered several surprising new features of the STOS Basic system. Probably the most startling of these concerned the AUTOBACK directive. As you may know, AUTOBACK automatically copies any graphics you draw on the screen into the sprite background. Unfortunately there's just a teeny weeny logic bug in the design!

When I finally had the chance to give AUTOBACK, a thorough work out, I was amazed to learn how inefficient it turned out to be. Take the following example:

```
10 timer=0:? "Start"
20 mode 0
30 for i=1 to 1000
40 draw rnd(310),rnd(190) to
rnd(310),rnd(190)
50 next i
60 ? "Stopped at",timer/50.0
```

In practice, this program executed in around 8 seconds with the AUTOBACK feature switched on. But when I added a couple of lines to deactivate the AUTOBACK system, the timings dropped to an impressive 5 seconds:

```
35 autoback off
55 screen copy physic to back
```

The screen copy at line 55 moves the new picture you have created straight into the sprite background in one smooth operation. Furthermore, providing you don't intend moving the sprites while the graphics are being drawn, the results of the two approaches are identical – but the screen copy system is much faster.

The amount of time you can save with this trick will obviously depend on the type of graphics you are attempting to produce. If you are drawing only a few lines (10 or less), the AUTOBACK feature is actually marginally faster. On the other hand, if you are generating more complex graphics such as circles and boxes, even the most trivial drawings will be speeded up significantly using the new system.

There is also one other problem with AUTOBACK which effectively drives the final nail in its coffin. AUTOBACK is totally incompatible with the screen flipping technique used by many games (including ZOLTAR). I've explained this technique in considerable detail in my new book as it's essential for the production of realistic screen animations on the ST.

If the LOGICAL screen used by the drawing commands is different from the PHYSICAL screen which is being displayed, then AUTOBACK will always copy the wrong information into the sprite background. Whenever a sprite is

moved, the currently active display will be mistakenly updated from the hidden screen your program is busily constructing. This will inevitably transform your game screen into a hopelessly confusing jumble. It's quite possible that you have already encountered this difficulty during your experiments with screen flipping. The solution is simply to add an explicit AUTOBACK OFF statement at the start of your program, and copy the LOGICAL screen into the sprite background after the image has been completed. In order to synchronize your screens with your sprites, you'll also need to perform the sprite movements directly using UPDATE or SYNCHRO (see pages 101 and 151-152 of your STOS Basic manual for more information.)

For me, AUTOBACK is one of those ideas which look great on paper, but which don't quite work out in reality. I've already placed conspicuous AUTOBACK OFF statements in most of my own programs, and in some cases the improvement has been pretty dramatic. So examine your programs carefully. You may be able to speed things up by over fifty percent with just a couple of extra instructions!

Warning! AUTOBACK is automatically reactivated by CLS and MODE. You should therefore always remember to place your AUTOBACK OFF directive after these commands have been executed in your program.

■

STOS Upgrade

Update your copy of STOS to V2.4 – the latest version which comes with the STOS Compiler (now working with TOS 1.4).

For just £2 you will receive a complete, replacement Language disc containing Basic 2.4, the single-precision floating point routine (SIN and COS run 30 times faster) and CONVERT.BAS to modify any programs with floating point to work with the new routine.

Send cheque or PO to Aaron Fothergill.

Gilbert in STOS!

It can now be revealed that Again Again's Gilbert was written in STOS Basic and compiled with the first working version of the Compiler to arrive from France.

Chris Payne at Mandarin Software agreed to let the development team leave off the proper credits as they felt it would affect sales – but on the understanding that the truth would be revealed after the game had been on the market for a few months.

You can check for yourself by using a disc sector editor to look at the program: You'll see François' name close to the beginning of the file!

Snippets

A team in Toronto, Canada, are using STOS Basic to conduct lightning research – and are so enthusiastic about STOS that they've set up their own bulletin board.

STOS Atlas is on sale in Australia. Pactronics, one of the leading distributors, is responsible for this three-disc educational package which provides information on more than 200 countries. You can even see the planets orbiting the sun moving at different speeds with pop-up messages to identify each one.

Richard Gale wins last month's typing competition with a high specification program which comes with more than 50 carefully graded lessons for you to type – there's even a mini typing game. Richard wins copies of STOS Compiler, STOS Sprites 600 and STOS Maestro Plus.

Andrew Braybrook, author of Paradroid, Uridium, and other classics has used STOS to create the maps for his latest masterpiece: Rainbow Island (the sequel to Bubble Bobble).

Using MIDI with STOS Part Two

Aaron Fothergill delves deeper into the magical mysteries of MIDI

In my last article on MIDI with STOS, I demonstrated how the MIDI line could be used to transfer data between 2 ST's. The original idea of MIDI was to enable musical instruments to talk to each other and to computers, so this article will deal with talking to synthesisers (or home keyboards that are MIDI'ed). First of all let's start by looking at the way the MIDI commands work:

As previously explained, MIDI is a standard of data communication where one musical instrument (computers also count as musical instruments except for the ZX81!) can tell another what notes to play or what sound to use. Each such instruction has a command number. Each MIDI message is made up of a command byte, and then one or two data bytes (System Exclusive can send more than two).

MIDI designates all bytes from 128-255 ($80-$FF) as commands.

Those from 128-239 ($80-$EF) are ordinary common commands such as Note On ($9n means 'play a note') or Pitch Bend ($En means 'change pitch of notes currently being played').

The commands from $F0 to $FF are System commands, the most common of which being $F0 the System Exclusive command (SYS EX) & $F7 End of System Exclusive (EOX). The other system commands for those who are interested are:-

$F2 Song Position Pointer (for sequencers)
$F3 Song Select (ditto)
$F6 Tune Request (rarely used in modern synths as they tend to stay in tune) $F8 Timing Clock (sequencers send this out so that other sequencers can keep time with them)
$FA Start >
$FB Continue > For Sequencers
$FC Stop >
$FE Active Sensing (Allows a synth to make sure it is still connected. Generally referred to by MIDI programmers as Pain in the A*** code
$FF Reset. Resets the Synth (a last resort!)

For writing Synth Editors and Librarians all you need to worry about are $F0,$F7 and $FE (which you have to filter out). To do a sequencer you need to consider just about all the command bytes as they will all get used at some point!

First things first though! We will go through the common commands, starting with the Note On command ($9n). The 'n' stands for the Channel Number as ordinary MIDI data can be sent over 16 different channels ($0-F). This applies to all the common control commands. So to tell our Yamato auto electronic home keybored (not misspelt!) to play a middle C on channel 1 (The only thing the MIDI people haven't decided on yet – some manufacturers call middle C C3 and others call it C4!) you would have to send 3 bytes to the synth.

Byte 1: $90 (Note On channel 1)
Byte 2: $3C (Note value for Middle C)
Byte 3: $7F (Maximum velocity)

The Velocity byte (#3) tells the synth how hard you hit the key when you played the note. Some Synths respond differently as you play the key harder. Thus the velocity parameter.

To do this in STOS:

```
10 open #3,"MIDI": rem open the MIDI line
20 M$=chr$($90)+chr$($3C)+chr$($7F): rem get the MIDI data ready
30 print #3,M$;: rem the semi-colon is vital!
40 close #3

RUN
```

Assuming your synth is connected it should

now be doing a rendition of Phillip Glass' latest hit "Long note of Middle C"! To turn the note off, you must send the equivalent NOTE OFF ($8n) command for the same note number on the same channel so change line 20 to

```
20 M$=chr$($80)+chr$($3C)+chr$($7F):
rem Note off

RUN
```

Ah peace at last. (If this doesn't work and you're not a Phillip Glass fan, reset the Synth or switch it off and on again, then make sure you typed the program in correctly.) The next most useful command is the program change command ($Cn) which allows you to change the sound being played by the synth (assuming it has more than one sound!). Program change only has one data byte, which is the program number to change to. Edit the program to read:

```
15 Input "Patch Number ";P: rem Get
Patch Number
20 M$=chr$($C0)+chr$(P): rem Change
to program (patch) P

RUN
```

Your synth should now have changed to the patch number you typed in (remember patch numbers start at 0). Experiment with the patch number value in the program to change the patch numbers on your synth.

By now you should be getting the hang of sending data to your synth, so let's have a go at receiving some data back from it. Note the extra spaces needed in lines 205 to 214 to ensure a neat display.

```
1 rem ************************
2 rem * MIDI MONITOR PROGRAM
3 rem * Aaron Fothergill
4 rem * STOS Club 1989
5 rem ************************
6 mode 1 : key off : curs off
7 fade 1,$0,$777,$700,$7
10 open #3,"MIDI"
20 C=0 : D1=0 : D2=0
25 gosub 200
30 I=-1 : while I<0 : I=port(#3) : wend
35 if I=$FE then 30 : rem filter out active
sense
40 if I<$80 then D1=I : goto 50
45 C=I : D1=port(#3)
50 if C>=240 then gosub 100 : rem clear
line of SYS EX data
55 if (C and $F0)=$C0 or (C and $F0)=$D0
then D2=0 : goto 70
60 D2=port(#3)
70 locate 10,10 : print "Com:$";right$("
"+mid$(hex$(C),2),2);" D1:$";right$(" 0
100 I=1 : while I>=0 : I=port(#3) : wend :
return
200 paper 3 : pen 1 : locate 0,0 : centre
"STOS Club MIDI Monitor Mk1"
205 paper 1 : pen 0 : locate 40,5 : print "
Command Values    "
206 locate 40,6 : print "                    "
207 locate 40,7 : print "$8n- NOTE OFF
"
208 locate 40,8 : print "$9n- NOTE ON
"
209 locate 40,9 : print "$An- POLY AF-
TERTOUCH  "
210 locate 40,10 : print "$Bn- CONTROL
CHANGE  "
211 locate 40,11 : print "$Cn- PROGRAM
CHANGE  "
212 locate 40,12 : print "$Dn- CHANNEL
AFTERTOUCH"
213 locate 40,13 : print "$En- PITCH
BEND    "
214 locate 40,14 : print "$F0-FF SYSTEM
"
219 paper 1 : pen 2 : return
```

Most of the program should be fairly familiar if you read the first article. Line 30 for instance is the bog standard 'wait until there is some MIDI data' routine (returning the first byte in variable I). You will probably notice the check on line 40 which looks to see if the data is less than $80 (128). This is because most modern synths use something called Running Status whereby they will send a command byte and then only send data bytes until the command changes (thus saving data and time). So our program checks to see if the data is a command. If it isn't then the program remembers the last command sent and the first byte must therefore be the first data byte. The check on

line 50 gets rid of any System commands as this monitor is not equipped to handle them fully. The Command # is checked on line 55 for Channel Aftertouch and Program Change, as these only use 1 data byte. The second data byte being read in from the MIDI line on line 60. The data is printed to the screen on line 70 and the program loops back to line 30 for the next MIDI command.

Once you've typed the program in, connect the MIDI out of your keyboard to the ST's MIDI in and hold down a (synth) key. You will notice the display come up with something like:

Com:$90 D1:$3E D2:$40

This particular one is a Note On on channel 1, note number $3E (D3) at velocity $40 (Half). When you release the key the command will change to $80 for Note Off. Now try pressing the patch select/change buttons and moving the controllers on your synth and seeing what data you get.

System Exclusive on synths

Most of the modern synthesisers and quite a few of the home keyboards can send patch data via MIDI.This is all the data that tells the synth how to create the sounds. Some synths send 1 Patch at a time, some a whole bank and others can send either. The next program will enable you to request a data dump from your synth and read it into the ST's memory so that it can be saved to disk. This enables you to save more sounds than the synth can normally hold on its own. This program is set up to request a single voice from a Yamaha DX21 synth. It will also work on the DX27, DX100 and TX81Z synths. To change it to work on other synths edit the data in line 45 to the request message required by your synthesiser (It is usually in the back of the manual. You want 'request dump' or similar). To use the program click on Load or Save on the alert box (you can use the alert box routine in your own programs) to either Load the sounds from disk and send them to the synth or to get the sounds from the synth and Save them to disk. Then let the program do all the hard work.

```
1 rem ****************************
2 rem * MIDI Sys Ex Dumper prog
3 rem * By Aaron Fothergill
4 rem * STOS Club 1989
5 rem ****************************
8 reserve as work 5,8000 : rem reserve
plenty of space to put the dump in
10 mode 1 : key off : curs off : fade
1,$0,$777,$700,$7
15 open #3,"MIDI"
20 AL$="Do you want to LOAD data
from disk|or SAVE it ? ## LOAD ^ SAVE
" : gosub 1000
25 clw : print "Getting data from synth"
30 if Q=1 then 200 : rem LOAD data from
disk and send it to synth
35 paper 3 : clw
40 M$="" : repeat : read D :
M$=M$+chr$(D) : until D=$F7 : rem read
data until EOX
45 data $F0,$43,$20,3,$F7
50 gosub 500 : rem clear MIDI line of old
data
55 P=start(5) : rem point to the start of
our data bank
60 print #3,M$; : rem the semi-colon is
vital!
65 I=-1 : while I<0 : I=port(#3) : wend : if
I=$FE then 65
70 poke P,I : inc P : if I<>$F7 then 65
80 rem save it to disk
85 F$=file select$("\*.VOX","Save voice
data to disk as .VOX file",1) : if F$=""
 then 300
90 bsave F$,start(5) to P-1 : goto 300
199 goto 20
200 paper 2 : clw : F$=file
select$("\*.VOX","Load a .VOX file from
disk",1) : if F$="" then 300
205 bload F$,start(5)
210 P=start(5)-1 : rem point to the start
of the data -1
215 inc P : print #3,chr$(peek(P)); : if
peek(P)<>$F7 then 215
220 goto 300
300 AL$="Do you want to do another ? |
## YES ^ NO " : gosub 1000 : clw : if
Q=1 then 20
305 end
500 Z=1 : while Z>=0 : Z=port(#3) : wend
: return
```

```
997 rem do an alert box in any resolution
. (screen destructive)
998 rem gosub 1000 with AL$ containing
data 1-2 line of text (| as separator) en
ded by ## then data for 2 buttons (^ as
separator)
999 rem e.g AL$="Do you want to load
or|save the data ? ## LOAD ^ SAVE
":gosub 1000
1000 LL=instr(AL$,"|")-1 : if LL<0 then
LL=instr(AL$,"##")-1
1005 paper 1 : pen 0 : XX=39-LL/2 : YY=8
: locate XX,YY : square LL+2,6,1 : for
Z=1 to 4 : locate XX+1,YY+Z : print
space$(LL) : next Z : L1$=mid$(AL$,1,LL)
:L2$=mid$(AL$,LL+2,instr(AL$,"##",LL+1)-
LL-2) : locate XX+1,YY+1 : print L1$; : l
ocate XX+1,YY+2 : print L2$;
1010 reset zone : NQ=2 :
EL=instr(AL$,"##")+2 : Q$=mid$(AL$,EL)
: Q1$=mid$(Q$,1,
instr(Q$,"^")-1) : Q2$=mid$(Q$,instr(Q$,
"^")+1) : if Q1$="" then Q1$=Q$ : NQ=1
1015 locate 39-len(Q1$),YY+4 : paper 2 :
pen 1 : print Q1$; : locate 41,YY+4 : p
aper 3 : pen 1 : print Q2$ : set zone
1,xgraphic(39-len(Q1$)),ygraphic(YY+4)
to xgraphic(39),ygraphic(YY+5) : if NQ=1
then 1025
1020 set zone
2,xgraphic(41),ygraphic(YY+4) to
xgraphic(41+len(Q2$)),ygraphic(YY+5)
1025 while mouse key=0 : wend : while
mouse key<>0 : wend : Z=zone(0) : if Z<1
or Z>2 then bell : goto 1025
1030 Q=Z : ND=1 : return
```

Anyway that's enough on MIDI for one article. More in the next one about the Control Change messages. If you've any queries just ring me on the Helpline. ■

TORTOISE PRINTING

Up until now I have not written to you because each and every query that I have come across and, in some uncanny way, just as I have come across it has been answered by the current newsletter. Coincidence or what, it has stopped many mutterings!

My only small complaint is that STOS seems to take a long time to LLIST a program (via my Epson printer) and forever (chess anyone?) for a screen dump. Yes, I do set the printer up first.

Can anyone out there come up with a listing for a printer SPOOLER.ACB? Preferably size adjustable from say 6k for text to 32k for a screen dump.

Clive Swain, Greenford, Middlesex

See STOS Word for a fast print routine – available free of charge to everyone who resubscribes for next year.

SPRITES HELP

Is there going to be a version of STOS where NO line numbers are used? I find it a lot easier to give each section/routine a name/label to go to rather than a line number. This makes writing and altering programs (especially long ones) easier to re-organise either with a built-in editor (move cut/paste etc) or to edit using another word processor on the file. Is this the feeling of anybody else?

There is also a problem when using the multi mode SPRITE2.ACB accessory. When in low res mode depending on the size and number of sprites you are restricted to a limited quantity stored in bank 1. This is caused by the lines 135 and 226 in the SPRITE2.ACB program.

You can change these in the normal way of loading STOS and then loading in the file as a normal program ie. Load "sprite2.acb". You will then have a listing of the accessory and can change lines 135 and 226. The value BANKMAX can be increased. I have increased it to $20000 with no problems but obviously the amount of free memory has to be considered when increasing the value of BANKMAX.

Someone was asking if there was a way of replacing the screen after closing a window without using copy to back/logic etc. What I have been doing is working out the window area before opening it (this only works with a non-moveable window) and then copying the same area into a string variable with the command:

```
S$=SCREEN$(scrn,x1,y1 to x2,y2)
```

Then restoring the screen after closing the window with the command:

```
SCREEN$(scrn, x,y)=S$
```

Page 148 in the manual will explain.

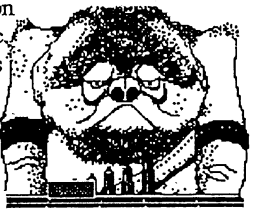
Can anyone help me with a problem? I am writing a program (strategy game) which involves using a lot of calls to gosub routines. Each routine is ended with a return which is always reached for each routine. But having performed a number of calls to these routines (about 50+ times) the computer responds with *Too many Gosubs in line*. I have tried the FREE command to force a garbage dump but with no success. Does anybody know the answer?

Finally could I point out that you do **not** have to write a routine which displays a load/save selection box, which I did. One simple command does all this for you already! Take a gander at the bottom of page 218 in the manual for the built-in Load file selector and the bottom of page 23 for Save file selector for more info. I wish I saw it $%^&* earlier!

L Groves, Swindon, Wilts

SCREEN EXPLAINED

Here is a brief description of Back, Physic and Logic.



The STOS screen is divided into three areas, LOGIC, PHYSIC and BACK.

LOGIC This is the area

of memory which all output goes to. If you print or plot then all will go to the logic area.

PHYSIC This is the area of memory you actually look at. Initially, after you boot up STOS, logic and physic point to the same area in memory so everything you print or plot can immediately be seen.

BACK This is the area of memory which is used as an 'underlay' to logic. So if a sprite passes over an area of 'logic' screen, logic will refer to 'back' to fill in after the sprite has passed. If back is changed, ie. a piccy loaded into it, then to view the screen you must somehow update logic or physic. Either use the APPEAR command or set PHYSIC=BACK. NB. If you set PHYSIC=BACK then you will not be able to see what you type until you either change physic or logic. For example:

```
100 reserve as screen,6
110 reserve as screen,7
120 load "picture.neo"
130 logic=start(6)
140 print "This is screen 6"
150 logic=start(7)
160 print "This is screen 7"
170 logic=physic
180 print "Press a key"
190 wait key
200 screen copy 6 to physic
210 print "Press another key"
220 wait key
230 screen copy 7 to physic
```

Here's how it works:
100-110 reserve memory banks 6 & 7 as screens
120 load picture to current output screen to watch while other things happen
130 set logic so all output (print/plot etc) goes to screen 6
140 print message to screen 6
150 set logic so all output (print/plot etc) goes to screen 7
160 print message to screen 7
170 set logic so all output (print/plot etc) can be seen
180-230 waits for a key to be pressed then copies screen 6 or 7 to be seen on physic

Another example would be of a driving type game where you normally see the road but with a press of the right key you switch to a map but continue drawing the road without messing the map up. Thus:

```
100 If K$="M" then physic=start(6) 110 if
K$="R" then physic=logic
```

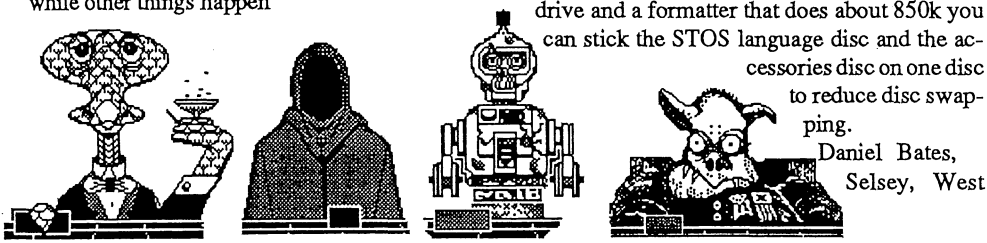
The above two lines demonstrate how you can flip between a map (if K$="M") and the road view (if K$="R").
George Ford, Dorking, Surrey,

GRABBING GAME SPRITES

I am writing in reply to the letter about taking sprites from games. The easy way to do this is to list the directory and look for files with .SPR extensions as these are normally sprite files. Then it is a matter of using the STOS sprite editor program to dig them out. Other sources may be .PI1 or .NEO files. It is just a matter of going through the files carefully. Most of the Atari games packs which come with the computer can be dealt with like this.

Another hint – if you have a double-sided drive and a formatter that does about 850k you can stick the STOS language disc and the accessories disc on one disc to reduce disc swapping.

Daniel Bates, Selsey, West



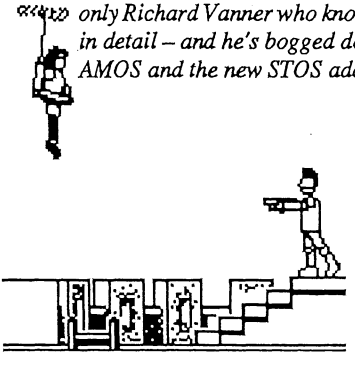
HARD DISC PROBLEM

Just one problem so far with STOS. I have it installed on my 60MB hard disc and it seems to run ok. However I have problems with the sprite editor. If I install it straight away after boot up then it runs okay. If I install the sprite editor and then another utility, for example the music editor, then when I recall the sprite editor it will not work properly. It will not draw to the main screen and I cannot change the colours as the colour change cursor is missing. This does not happen as long as I install the sprite editor first off and on its own. Any ideas? I have already tried unpacking and recopying from the original floppies to the hard disc but that doesn't seem to help. I can of course still use the package – it's just that with a Mega 4 ST it would be nice to keep all of the utilities loaded and to hand in memory when I am working.

I have tried to raise Mandarin on the phone but to no avail. I wonder if other users have found this problem? Anyway, apart from that it really is an impressive package and I am well pleased with it.

Mike Moseley, Trevillick, Tintagel,

It is difficult to get through to Mandarin, but the situation has recently changed as the company now has its own lines rather than you having to use the general Europress number. You can reach them on but Mandarin would rather you wrote in as there's only Richard Vanner who knows STOS in detail – and he's bogged down with AMOS and the new STOS add-ons!.



MAESTRO HAPPY

I have recently purchased the Maestro sound sampler and am extremely pleased with it. The ease with which you can capture and manipulate sound samples has to be seen and tried to be believed.

I include a listing for a short modification to MAESTRO.ACB which enables you to use quick keyboard inputs to manipulate the samples and lets you change pitch, add repeat, loop and sweep as well as loading and deleting of samples. The key calls are:

L = Load a sample .sam
P = Play a sample
R = Replay a sample
M = Load sample .mbk
B = Load a sample sam bank
+ = Increase speed
– = Decrease speed
0 = Samstop/reset
1 = Loop on
4 = Sweep on
7 = Play backward

Just type in lines 421-439, add line 405 and away you go. I also suggest that you delete the two PRINT commands in line 1380 as this will stop the screen scrolling up when you want to play a sample.

```
405 samstop: samsweep off: samloop
off: samdir forward
410 gosub 900
420 OPTION=mnbar: CHOICE= mnselect
421 K$=upper$(inkey$): rem Keyboard
Commands
422 if K$="L" then 740
423 if K$="R" then 1430
424 if K$="P" then 1370
425 if K$="D" then 1140
426 if K$="M" then 1080
427 if K$="B" then 1590
428 if K$="+" then inc Y
429 if K$="-" then dec Y
430 if K$="0" then samstop: samsweep
off: samloop off: samdir forward
431 if K$="1" then samloop on
432 if K$="4" then samsweep on
```

```
433 if K$="7" then samdir backward
439 if OPTION=1 and CHOICE=3 then
1490
```

Alan Ward, Purley, Surrey

SCROLLING ALONG

Here's another way to create a scrolling message across the screen – you'll find it in one of the Swedish games in the PD library.

```
5 TEXT$="Press the fire button on the
joystick to play the game"
10 def scroll 1,0,184 to 320,192,-4,0
20 repeat
30 scroll 1 : wait vbl : scroll 1 : wait vbl
40 locate 39,23 : inc PEK : print
mid$(TEXT$,PEK,1);
50 if PEK=len(TEXT$) then PEK=0
60 until fire
```

EXTRA TIP

Clarification of a minor problem with the STOS Compiler. If you use the COLLIDE() command you may only use a number or a variable inside the brackets. Any form of equation will cause an error when compiling. It is possible that some of the other instructions may have this problem, so if your compiler has trouble in compiling certain programs then check for functions which have fiddly bits in them. For example the following will most likely cause the compiler to have a minor nervous breakdown:

```
if collide
((WIDGETNUM*FLANGTIME
+Z),X^2+2*Y,Y+3)=1 then ...
```

Whereas this will work:

```
A=WIDGETNUM*FLANGTIME+Z:
B=X^2+2*Y: C=Y+3 if collide
(A,B,C)=1 then...
```

■

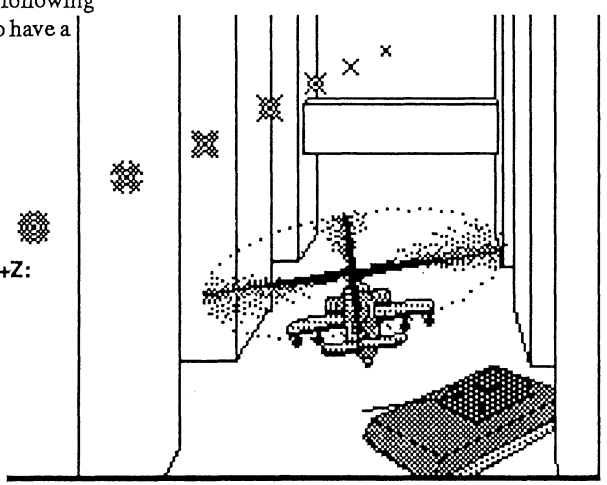
APPEARING DIFFERENT

by Richard Vanner

After seeing hundreds of STOS games I've got rather tired of the APPEAR routines which nearly everybody uses in their programs. So what I plan to do is produce a new screen appear every issue to add a bit of variety. If you can come up with a routine along similar lines, please send it in. Anyway, here's my routine, but make sure you do a RESERVE AS SCREEN 10 first and load an appropriate picture into it:

```
10 rem Picture Splurdger by Richard
Vanner
20 key off : curs off : flash off : mode 0
30 logic=physic : show on : F$=file
select$("*.NEO","Select Pic to
splurdge!") : if F$="" then end
40 logic=back : hide on : load F$,10 : get
palette (10) : Y1=99 : Y2=100
50 for Z=Y1 to 0 step-1 : screen copy
10,0,Y1,320,Y1+1 to logic,0,Z : next Z
60 for Z=Y2 to 200 step 1 : screen copy
10,0,Y2,320,Y2+1 to logic,0,Z : next Z
70 dec Y1 : inc Y2 : wait vbl : screen
swap : screen copy physic to logic
80 if Y1<>-1 then goto 50
90 repeat : until mouse key<>0
100 goto 30
```

■



Build up your collection of

...and learn more about STOS at the same time

Listed alongside you'll find a good selection of interesting programs with something for everyone. Each disc costs just £2 each – or £1 if you supply your own disc. They are supplied in single-sided format on double-sided discs unless stated otherwise. Three of the discs are shareware – the authors will receive payment from every disc that's ordered (deduct £1 if you supply your own disc). Each disc has a specially-printed STOS Public Domain coloured label produced by Mandarin Software to match the rest of your STOS master discs.

In most cases the programs require you to boot your copy of STOS first – that way you can have a good nosey at the listing!

Many of the Swedish demos are rather lacking in documentation – some of the instructions are written in Swedish! However you should be able to suss most of the games out anyway.

Ring Sandra on to find out about the latest public domain titles to be added to the library.

If you have any programs which may be suitable please send them along. You can also get something back for your efforts by making your submission shareware.

Send cheques, postal orders or stamps to: *Sandra Sharkey,*

Choose from the following:

SPD15: *Caves of Rigel* – a well-designed arcade-type game converted from the Atari 8-bit commercial release on Atlantis Software by the original author, Ralph Effemey. There's also another game from Ralph: *A Froggy Day in London* based on the classic arcade game Frogger.

SPD16: *STOS Demo* – The first demo created for use in shops – no great shakes, but includes a nice tune you can grab!

SPD17: *STOS Demo2* – The cycling demo you may have seen at the shows, using Ian Waugh's attention-grabbing tunes from the Accessories disc. (Double-sided only)

SPD18: *STOS Add-ons Demo* – The latest demo from Mandarin with sampled sound – you'll need the Maestro extension to run this.

PD10: *Xmas Demo* – This double-sided disc contains a well-executed and amusing spoof of the nativity.

PD14: *First Serve* – Wimbledon revisited in this one-player against the computer tennis game.

PD19: *STOS listings* – Programs and routines from Newsletter 4.

PD21: *STOS Basic Update* – Upgrade your copy of STOS to version 2.4. It also works with the new TOS 1.4 chip.

PD22: *Fun School 2 demo* – See two programs from each of the three packages (two 360k discs [£4] or one 720k disc).

SH5: *Brain Games* – A selection of thinking STOS games (Gridder, Minefield, Rotation, Solitary and Swopper) from Pete Gerrard and Sandra Sharkey. (£4.25)

SH7: *STOS Games* – A selection of simple STOS games by Mike Brown. (£3.00)

SH9: *Kids Games* – Five games for pre-school and primary children by David Alexander. (£3.00)

PD23: *Swedish One* – Two games submitted in the Swedish competition. Yatzy is a cleanly-designed version of the dice game Yahtzee with a good tune, and Virus is a flip screen maze game in which you are a cute-looking alien searching

STOS programs

for missing 3.5" discs! Some great sprites for you to grab.

PD24: *Swedish Two* – Fia is a version of Ludo controlled with the mouse pointer – good fun for kids. Mario is an infuriating platform game, and Rush is a 'shove-the-bricks' to solve the puzzle.

PD25: *Swedish Three* – Saga Castle is an ambitious multi-screen platform game with lots of puzzles to solve. Stratego is a simple war game.

PD26: *Swedish Four* – Frog Race is a complex game in which you bet on a number of excellently animated jumping frogs which race towards the finishing line. Upstart is a nicely put together shoot-'em-up

PD27: *Swedish Five* – Acid Remix is a sprousedup version of the old standard Snake game with some clever sampled sound at the beginning.

PD28: *Give Us a Break* – An excellent version of the popular pub game based on DLT's radio show. Try to pot all the snooker balls by answering multiple-choice questions with difficulty related to the value of each ball.

PD29: *Home Finance* – Quite a good financial program.

PD30: *Maestro Samples One* – A marvellous collection of samples put together by Martin Walker, author of a number of commercial C64 games including Chameleon, Hunter's Moon and Citadel. He also did the music for Armalyte – and the samples include effects used in some of these games plus some newly-created ones.

PD31: *Maestro Samples Two* – A wide selection of useful sounds for use in your games, recorded at 6Khz (12Khz if double density). Includes laser bolts, alarm, pulse, scream, warble, warp, takeoff and tinkle – 20 in all.

PD32: *Maestro Samples Three* – Eleven more samples including alarms, cocking, fountain, hum, shoot, tantrax and up/down.

PD33: *Thunderbirds* – An entertaining homage to the cult TV show with amusing animated sequences, great music and top class Maestro samples.

PD34: *Phantom of the Opera* – A superb music demo which combines Vidi-digitised animated clips from the video of the classic Iron Maiden tune you hear in the current Lucozade ad.

PD35: *TOME demo* – Try out for yourself this sophisticated multi-screen map editor with built-in machine-code scrolling routine.

PD36: *PCP* – The full-screen editor which allows you to forget about line numbers and add labels and procedures to STOS (see article on page 10). The disc also includes the rest of the listings from this issue.

PD37: *STOS Typing Tutor* – The winning entry in Issue 4's competition. Richard Gale's excellent program has 50+ exercises and a built-in typing game.

PD38: *ST Wizard* – Richard Gale (again) has put together a top-notch demo: Spiralling sprites casting shadows on a checkerboard landscape, a scrolling message and a long sample of the Top of the Pops theme tune. Just great! (Any ST with double-sided drive)

PD39: *STOS Paint* – A feature-packed art program written by Ralph Effemey which loads as an accessory so you can flip to it with ease. A unique feature allows you to paint with sprites from the sprite bank. Also on the disc is the dealer demo for STOS Maestro (requires extension). (£5.00)

PD40: *Blood Money* – Andrew Webb has crammed his adaptation of the theme music for this Amiga game (2.57Mb of music) into 350k with clever use of looping and repeat phrases. It plays for four minutes in all. (Requires 1Mb) ■

Coming soon: The Poltergeists (creators of the excellent Thunderbirds demo) will be bringing you their latest production – Batman! Available soon ... watch this space!

STOS at the Atari Show

Mandarin Software had tremendous success with its stand where Richard Vanner and Chris Payne spent the whole time demonstrating the range of software available – concentrating about 50% of the time showing off STOS products on a 37-inch monitor. They showed off the Gameswriter winner (see below) and highly commended titles such as Mouthtrap, Pukadu, Arthur of the Britons, Skate Tribe, Dice, Poker Dice, Wild River Run – and many more. Some of these will appear on a compilation – others may be available on the public domain or on an ST magazine's cover-mounted disc.

STOS on public domain

Many of the top PD libraries are building up an impressive collection of STOS programs for everyone to try out. And PD software house Budgie have got in on the act by releasing its first STOS game: Parabellum, which is only available from specially-licensed public domain libraries which send Budgie a small percentage of the money received.

More details from: Goodmans PDL, Longton, Stoke- on-Trent, Staffordshire

Also: The South West Software Library, Wimborne, Dorset and Softville, Waterlooville, Hants

STOS Plus in 1990

François will be starting work on a major update to STOS in the new year, once he's completed AMOS and AMOS Compiler. STOS Plus will include word processor-like environment, optional line numbers, procedures, two-joystick routine, multi-mode displays, multi-palette screen, STE enhancements, a completely new collection of games and much, much more. Registered STOS users will get a low-price upgrade – so any of your friends who buy STOS now will not lose out – and there's a good chance they'll benefit from buying now rather than later. Price will be around £49.95 in the late spring of next year. Write to Mandarin with your wish list.

AMOS developments

For those of you with Amigas, or friends with Amigas, Mandarin have produced a detailed spec sheet for *AMOS – The Game Creator*, which will now be released in January. Simply write to AMOS Information at Mandarin Software, Europa House, Adling- ton, Macclesfield

AMOS will be a structured language with labels and procedures, and with its word processor-like environment there's no need if you want for line numbers – but you can use them for computed GOTOs and GOSUBs if that's how you like to program. It will also be possible to port games from the ST to the Amiga. AMOS will have more than 400 commands and will support the copper chip and blitter, allowing 48 hardware sprites and 32 software sprites on screen at once. The price will be £49.95.

£5,000 Awards winner

Simon Cook, a university student from Manchester, took away a cheque for £5,000 at the Atari Show for winning the title of STOS Gameswriter of the Year. His game Cartoon Capers impressed Mandarin with its original ideas, clever graphics and good use of sampled sound.

The game features Karate Kat and Judo Jake locked in head-to-head combat in a variety of scenarios. You control Judo Jake and have a wide range of moves at your disposal: You can even pick up a hammer and clobber the cat over the head, throw a bomb at him, or even spin him round your head and propel him through a wall – all with impressive cartoon-style animation.

Simon's game was written entirely in STOS, compiled using the Compiler (what else) and includes sampled sounds using STOS Maestro. Cartoon Capers will be on sale in December for the Atari ST at £19.95. An Amiga version written in AMOS will be released at the same time (fingers crossed). ■