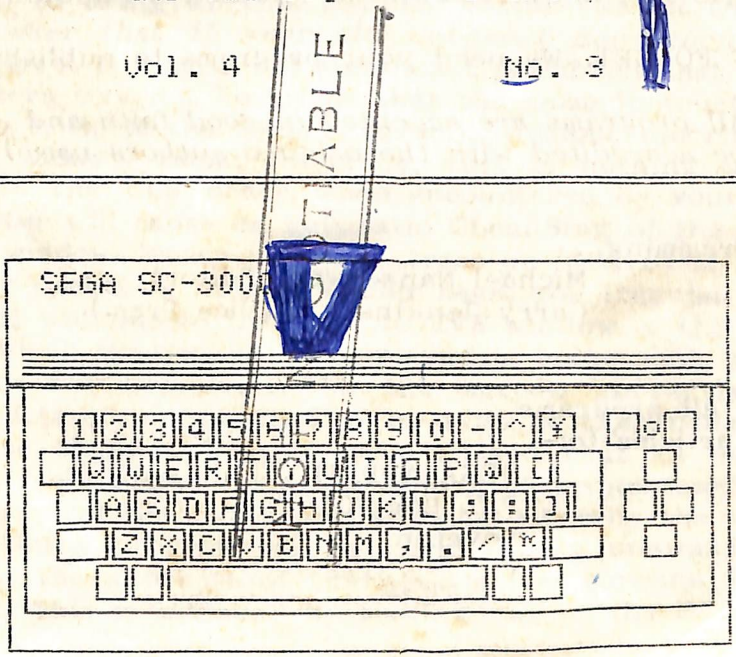
SEGA addict

The Official Magazine of
the Vic Sega Users Group

Vol. 4 No. 3



INKEY$:

Hello and welcome to the new improved model of ***SEGA ADDICT.*** "*Why does the magazine look different?*". I'm glad you asked that question! The club, in it's wisdom, has purchased a new printer for the production of the mag. This new device is an **EPSON LQ-500**, which is a 24 pin letter quality dot matrix printer which has lot's of wonderful features built into it (you'll see more of these as we learn how to use them!).

A short time ago the V.S.U.G. annual general meeting was held where our much loved treasurer, **Ray Finke**, decided to step down from his position. Ray has been a solid keystone for this group for a long time not only as our treasurer but also as public officer and co-ordinating the printing and distribution of the magazine (among other jobs). The committee would like to thank Ray on behalf of the club for his magnificent effort in sustaining and supporting the Victorian Sega Users Group.

PROGRAMS PROGRAMS PROGRAMS PROGRAMS PROGRAMS

DON'T FORGET...We need your programs to publish!

(NOTE: *All programs are accepted in good faith and should be accredited with the original authors name)*

Happy programming...

Michael Nanscawen (Editor)
Garry Jenkinson (Vice Pres.)

Please send all programs and articles to:-

SEGA ADDICT
P.O. Box 102
Doveton
Vic. 3177

....Life outside the CIRCLE....

..OR..

....co-ordinating your colour scheme....

Yes, folks!!! This month, we are going to spread our wings and travel to routines far and wide!!! Routines that reside within the bounds of your beloved SEGA's BASIC. So far, in fact, that we may find ourselves making a **BLINE** (pun intended) towards areas that are beyond the realm of the sane human mind to comprehend (did I say that???) (*do not adjust your screen, the fault is in this programmer...ED*).

....Making things a little Clearer....

Leaving the CIRCLE behind (yeh, I know it's difficult to say goodbye to old friends, but sometimes it's for the better...), we find ourselves confronted by the mysterious instruction **'CLS'** (at least that's what comes next in MY instruction manual). Some of you may wonder why I have bothered to bring up this subject matter (*was it the triple scotch after that 15 year old Cabernet Sauvignon??...ED*). Admittedly, it is a small but significant possibility that many of my readers may not be aware that the same instruction can apply to both of the SEGA's screens without discrimination (and in today's racist existance, this is nothing if not a miracle!!!). The **'CLS'** order, when encountered by your SEGA's interpreter will cause an automatic **CL**ean**S**ing of the current *ACTIVE* window. Please note the stress on 'active'. If you can recall my ramblings from an earlier saga, you (regular readers included) will realise that the *ACTIVE* window is the window toward which the data to be displayed is being sent, although this may not necessarily be the screen currently being displayed (are you confused? - *no more so than usual...ED*)!!! For those of you who may have missed that all important episode, I elucidated upon the ability of programmers to be able to prevent the undesirable situation whereby the observer of a **RUN**ning program can be made totally unaware of the fact that the alternate of the SEGA's two screens is being changed. This is achieved by making use of the **SCREEN 1,2**

or the **SCREEN 2,1** instruction to hide any such change. If you missed that item, you will have to wait for further explanation when this subject reappears in it's **AFFERBECK LAUDER**....so there!!! All of the foregoing notwithstanding, most of my itinerate followers should be able to grasp the fact that 'CLS' will simply CLear the current Screen being displayed (*say no more...nudge...nudge...wink...wink...ED*)!

....Only a pigment of your imagination....

Or is that MY imagination? What is reality anyway?? When I find out, rest assured folks, you will be the first to know!!(*get on with it!...ED*) Right, back to the matter in hand (*you'll go blind!!...ED*). Next on the list of instructions (unless I can be otherwise corrected) , is none other than **COLOR** (it doesn't take a Rhode's Scholar to tell that BASIC was written by an American, does it??). If you happen to use one of those marvellous SEGA 'Green Screen' monitors or a monochrome television, then I suggest that you cease reading from this point and find something better to do. *No, no,* come back....I was only kidding!!!!

So, what does this magical statement enable the BASIC programmer to do? Well, in the simplest possible terms, it allows the programmer to set or alter the colours of the screen. How this is achieved will depend upon which screen is being displayed, which will, in turn, affect the number and format of the parameters which apply to this instruction.

Before I delve into the intricacies of the aformentioned formats, it might be worth my while to spend some time explaining the one relevent aspect of the workings of the SEGA's Video Display Processor. This **VDP**, the Texas Instruments's TMS 9929A, is capable of generating up to sixteen different colours and it can display any two of these at any one time for any single BYTE of screen data. For those of you who may have forgotten (or those others not yet enlightened), a BYTE of screen data is any one of the 6144 groups of EIGHT pixels whose patterns constitute what we see upon the screen (*anyone requiring further clarification is directed towards SEGA ADDICT, vol. 2, No. 4, pp 29 to 38...ED*). As explained in GRAPHIC detail in that episode of WORKSHOP, the VDP performs this amazing feat by providing the 'ON' bits of any such BYTE with one of these colours - that which we BASIC users call the 'FOREGROUND' colour - and the 'OFF' bits of the BYTE with the colour we have christened the 'BACKGROUND' colour. It is IMPERATIVE that this concept be fully understood at this point, as a

failure to comprehend it's simple complexities (*is that possible??...ED*) will make what follows even more confusing than it already is !! All of which leads me very nicely to the first of the parameter formats which COLOR requires

....Changing pens....

When you first provide power (or when the **[RESET]** is activated) to the internals within the cold, inhuman plastic case that is your SEGA computer, you are confronted with (what I call) the 'Interpreter start-up message' which appears as black text upon a green background (that is, unless, like me, you have a modified version of disk BASIC - I look at white text on dark blue paper - but that's another story). However, you are in no way compelled to keep this colour combination if you do not like it! Alternatively, you may want to change these colours for whatever reason during a program run. This is where **COLOR** steps into the act.

At any time you want to change the colours of the **TEXT** screen, you simply provide the **COLOR** instruction with two parameters which, co-incidently, correspond to the colours which you wish to use, like this....

```
COLOR 15,4
```

....remembering that the first of these pararmeters is the one allocated to the 'writing' colour and the second for the 'paper' colour. It's as easy as that!!!! You have my permission, right this very instant, to enter as a direct input your own version of this statement (or you can use mine if you're feeling lazy)!! Should you not like the colours that you choose then you can simply change them again by providing other values to the statement. By the way, I feel that I should mention that there are certain limits to the values that you can put into this instruction. You see, the SEGA's VDP is only capable of producing sixteen different colours (as the accompanying chart shows). If you attempt to exceed the limits (0 to 15) then you will incur the wrath of the "Statement parameter" error message!!

What's that??? You only want to alter the writing

colour! This is not a problem for your clever little SEGA. All you have to do is to simply enter....

```
COLOR n
```

....where 'n' represents the value that you want to use. You'll notice that I have omitted the (all important) comma in this version. That's because it isn't needed on this occasion. However, leave out the comma for the next alternative for this statement and you will find yourself doing that which you have already done! Of course, this alternative is the one that you use to change the background colour. To do this, you make use of the second of the parameter values. Like this....

```
COLOR ,n
```

....and **DON'T** forget that comma.

I shouldn't have to mention this, of course, but should you accidently find that you provide the same colour for both the ink and the paper, you won't be able to read what you have written! If this situation does occur, you can rectify things by one of two ways. Firstly, you can press the **[RESET]** key. This will return you to the 'interpreter message' and your original screen colours. The other 'fix' is somewhat more involved, but by no means, difficult. What you have to do is to very carefully (so that you don't make a mistake), enter the direct instruction, **COLOR1**. This will immediately make all writing black. If your screen colour happens to be black too, then you should choose another colour instead. Nonetheless, the principle is the same.

....A Multitude of Colours....

It is at this point that this discussion becomes a little involved. Not neccessarily confusing or complicated (*thank heaven for that...ED*), but some of you will need to don your thinking caps!!

As much as I know that you have become accustumed to the ease with which you can now change the colour of your screen (why does that sound familiar??), we must move onto more engrossing and interesting pastures, vide licit, your SEGA's Graphic screen colour instructions.

It would be very easy for me lunge (and for you to plunge beyond your depth) into an explanation of how to use

this version of the **COLOR** statement, but I feel that I must explain some more of the inner operation of **SCREEN 2** before I take the plunge (or is that lunge??).

Most of you, I would hope all, will recall my words from SYNTAX Vol. 4, No. 1, where-in I discussed the nature of PIXELS and how 'text' was presented upon the Graphic Screen. Remember how I referred to the 6x8 blocks which display each character on the **TEXT** screen and how the **GRAPHIC** screen is comprised of those 49152 pixels? Some of you may have got the impression (though from where I do not know) that text is also presented in these same 6x8 blocks on **SCREEN 2.** This is not the case (trooly rooly)!!!! In fact, the **GRAPHIC** screen is actually comprised of a series of 'blocks', but they are not 6x8 pixels as you might think. They are, in fact, (are you ready for this...) 1x8!! Yes folks, your SEGA's venerable VDP requires it's GRAPHIC video display data in amounts that are 1x8 pixels. How many of you have realised that this is the same size as a single BYTE of information?? Can any of you remember back to the issue of SEGA ADDICT when I took over the writing of this article (*you mean you haven't been doing this forever...ED*)?? There-in I expounded upon the virtues of *BINARY NOTATION* and exposed the inner secrets of 'BITS and BYTES'.

Well, now that knowledge is going to come in very handy (you haven't forgotten any of it, I trust)!! Actually, only one aspect of the use of bits and bytes is relevant to this dialogue (*a soliloquy this aint...ED*). This is the reality that the VDP is programmed (it is only another silicon chip, afterall) to display all of the 'ON' (or '1') bits of the incoming byte of data as the 'foreground' or writing colour and the 'OFF' (or '0') bits as the 'background' colour.

What MUST be remembered from all of this, and this is FUNDAMENTAL, is that within any one of those 6144 individual blocks of GRAPHIC screen data there can only exist **TWO** colours. Perhaps this program will illustrate my point better (a picture IS worth a thousand words - *especially a thousand of yours...ED*)....

```
10 REM PROGRAM #1
20 SCREEN 2,2:CLS
30 COLOR 6:CURSOR 103,90
   :PRINT CHR$(229)
40 COLOR 3:CURSOR 114,90
   :PRINT CHR$(229)
50 COLOR 10:CURSOR 124,90
```

```
    :PRINT CHR$(229)
60 COLOR 15:CURSOR 137,90
    :PRINT CHR$(229)
70 COLOR 12:CURSOR 148,90
    :PRINT CHR$(229)
80 GOTO 80
```

....and although everyone can see that I have instructed my BASIC to place five 'boxes' upon **SCREEN 2**, using a different colour for each box, can anyone explain why it is that this has not occurred?? In fact, how many of you have seen the effect I have produced on previous occasions??? If so, you should now be able to explain what causes this!!

For the rest of you, I suppose I had better fill in the blank spaces so that you can avoid this problem in your own programs. Firstly, you must recount what I uttered earlier regarding the displacement of the **GRAPHIC** screen colour data. What I mean is this. Because the screen is divided into 'blocks' that are 1x8 bits, each area of individual colour must lie within one of these boundaries, id est, it will have a **CURSOR** value that is a multiple of eight (across the screen) or a multiple of one (down the screen). As you might appreciate, there are few problems encountered when drawing upon the screen vertically, however, care must be taken when drawing horizontally across the screen. Take my program (*please, as far away as possible!!...ED*), I am doomed to failure right from the very start. Line 20 has the first of my squares located at GRAPHIC SCREEN CURSOR X co-ordinate 103. A quick consultation with my calculator tells me that 103/8 = 12.875! This is clearly NOT a multiple of eight! So, although I have located my square in the middle of the **PATTERN** 'block' (which is perfectly legal and doesn't cause too many problems), the interpreter has automatically started the writing colour from the previous colour 'block' nearest this CURSOR X co-ordinate. Thus, this block will start at the co-ordinate X = 96.

Now, all of you can see that the co-ordinates concerned will cause the **PATTERN** of my character blocks (**CHR$(229)**) to overlap into the next 'block' of colour data and since (you'll have to take my word for this for now), such **PATTERN** data follows the same rules as the colour data and another

episode of SYNTAX wil be required to explain this.

So, at this point, we have our first block of colour data starting at co-ordinate 96, whilst the pattern data starts at co-ordinate 103. This is no problem, because the interpreter will automatically tell the VDP to alter the colour data for the following byte so that the whole character appears in the colour I have selected. This means that the colour data now covers two 'blocks'. However, all hell breaks loose when the program executes line 30.

Here, we tell the interpreter to place another square one pixel's distance from the previous one. No worries as far as the **PATTERN** data is concerned. The trouble is caused by the fact that I have now asked SEGA to **PRINT** this character in another colour. Given some thought, it is clearly obvious what will happen.

This **PATTERN** data has been located at a **CURSOR** co-ordinate such that it's colour data is overlapping the second of the 'blocks' which were previously allocated to that first square!!. Since I am now attempting to force the VDP to allocate two 'writing' colours within a single byte, a mammoth struggle takes place over which colour gets preference. In the end, it will always be the last colour value specified that will oust any previous value.

This adverse effect is known as '*COLOUR BLEEDING*' and, as a result of the predetermined scheme of things, all sorts of undesirable consequences can occur if the programmer is careless with the allocation of co-ordinates.

....Looking at it in GRAPHIC detail....

So, what has all of this to do with the matter under discussion. Read on, McDuff (*who's he?...ED*)!!! I wonder how many of you noticed that I used **'COLOR n'** in exactly the same way to set the (in this case) 'drawing' colour as I did to set the 'writing' colour on the **TEXT** screen?? Those who did may go to the top of the class, because this is the correct methodology required!! However, I suggest that you do not attempt to change the background colour in the same fashion. Unfortunately, **COLOR 1,15,** (when **SCREEN 2** is required) will only incur the wrath of Syntax error (*some would say a fate worse than death...ED*)!! Don't believe me??? go ahead and attempt this direct input....

```
SCREEN 2,2:COLOR 13,5
```

....and see what you get!!!!

As a result of all of the preceeding drivel, the interpreter

requires somewhat more precise information so as to accomodate it's ability to have the aforementioned two colour capability in each colour 'block'. In fact, it is because of this that the next SYNTAX requirement for **COLOR** is the way it is. So what is this requirement?? It is, quite simply, the start and end co-ordinates of the area to which the background colour is to be allocated.

As is the case with ALL other co-ordinate specifications, these must be encompassed within parentheses (*brackets, in other words...ED*). These brackets are used in the following format....

(x,y)-(xx,yy)

....where the first group contains the starting point (top right hand corner) of the area to be coloured and the second contains the co-ordinates for the ending point (bottom left hand corner) of this area. Side-stepping the issue for a moment, have any of my readers noticed that whenever the SEGA's BASIC interpreter is required to know a co-ordinate (or point) on the **GRAPHIC** screen, almost **WITHOUT EXCEPTION**, these points will be enclosed by brackets?? It proves a simple progression from this, that, once the requirements of an instruction or command are learnt, if such requirement is a set of co-ordinates, then the SYNTAX MUST require these brackets!!!

Returning to the more immediate present, why, you may well ask, did I take so long to get to this point and get so involved with the VDP for such a simple explanation? I did this so that you should avoid the problems that can occur (by having knowledge of them) in your programs. Now that we know how to instruct the interpreter as to where we want the area of background colour, it only becomes a small matter to tell it what colour to put there!! Let's look at program #2 and see how all of this works....

```
10 REM PROGRAM #2
20 SCREEN 2,2:CLS:COLOR ,15
   ,(0,0)-(255,191)
30 COLOR ,3,(50,20)-(100,50)
40 GOTO 40
```

....and immediately, the more observant of you will notice something different about the **COLOR** statement in line 30 of this program and that which has been offered before. Yes, I have omitted the specification of a 'drawing' colour (by the usual expedient of providing an appropriate number of commas and the ommision

of any value) and then following it with the...you guessed it...'background' colour specification! Lest you should think that this will cause a failure of the syntax, the BASIC interpreter will allocate the current drawing colour or the 'start up' colour (black) to any of the 'ON' bits that may be involved (though there are none here).

Further, some may perceive that the values for the X co-ordinates (arbitrary as they are) that I have chosen for the area of screen to be coloured are not in multiples of eight. I have done this deliberately to show how the programmer can 'come-a-cropper' without proper attention to detail or thought to what he/she really wants to see upon the screen!!

When you **RUN** this little program (if you haven't already), you will notice that it does so without any problems at all (*that's a novelty...ED*). That is until you attempt to do this....

```
35 COLOR ,10,(24,0)-(53,45)
```

....and with this addition, you will notice that the yellow area of background is not exactly where one might expect it to be!! Where it quite obviously (as was intended) overlaps the green area, it should do so but diligant observation will show that it in fact overlaps that area by the mammoth amount of a full eight pixels, five more than was required. Why is this so??

This apparent failure to carry out instructions is not a case of insubordination, but an attempt on the part of the issuing 'officer' to extract more from his valued recruit than that body is capable of providing. Let me explain. As I have stated already, though I may not have stressed it's importance, it is imperative to the PROPER organisation of the GRAPHIC screen that all areas of background colour are allocated starting points whose X co-ordinates are multiples of eight, and whose end X co-ordinates terminate at a value that is 1 pixel less than the co-ordinate of the starting

point for the next 'block' of eight colour bytes. It might not seem obvious to some, but, if some thought is given to this topic, should the preceeding matters be ignored, the result (upon the screen) will be nothing less than an absolute shambles.

Though I need not have to press the point any further, none of the aforementioned problems occur to the Y co-ordinate, and as such, programmers may feel free to locate their Y co-ordinates within the limits available, separating them a mere 1 pixel lower 'down' the screen.

....Regressing to where we started....

Uphold thy hands thine of my followers who hath seen that the aforesaid declaration doth contain within it's meagre content more than hath thus far been exposed (*I noticed a little bit on the end of your last direct input statement...ED*)! What does this hold in store for the intrepid traveller? Tis well that thee ask!!

Lest it be said that your erudite tutor should leave you, his trusted disciples, in a state of neglect, it is with pleasure that I plunge (yet again) into a discussion of the aforementioned addition!! This appendage to the apprently otherwise complete **COLOR** statement poses yet another possibilty for the presentation of the SEGA's **GRAPHIC** screen. You see, there is more to the TMS 9929A than many realise!! I am sure that nearly ALL of my autodidacts have noticed, on many occasions, that when they have set a BACKGROUND colour for the **GRAPHIC** screen, that there is an area surrounding this so-called screen, which is not within the jurisdiction of the regulation X and Y start and finish co-ordinates of the **COLOR** statement.

This area has one of two potential uses. It can be used to display the input of an external video source, such as a normal television transmission, or it can be programmed (via **COLOR**) to provide an additional contingency to the variation of the impact produced by the creative use of colour on **SCREEN 2.** Before I continue, I should point out that understanding of the External Video feature of this VDP chip is above and beyond the scope of this article (but I couldn't resist the temptation to whet your appetites)!!

This notwithstanding, it is simply a matter for the programmer to decide which colour he/she desires in this area and to then instruct the SEGA to place that colour there, by inserting the desired colour code as the last of the parameters for the COLOR statement (not forgetting the

omni-potent comma). As could easily be imagined, it would be possible to create the illusion that the operator at the other end of the program would be seeing a bi-coloured screen containing nothing more than text, and though they may believe they are looking at the text screen, they are , in fact, seeing the graphics screen. A change then from **SCREEN ,2** to **SCREEN ,1** would bring about an instantaneous change to the text screen to the amazement of such a person, but an explanation of that device must wait for another emanation from my illustrious pen (*that's an amazing Osmiroid you have there...ED!!*). See you next time we emanate....

COLOR CHART

Color Code	Color	Color Code	Color
0	Transparent	8	Red
1	Black	9	Light Red
2	Green	10	Dark Yellow
3	Light Green	11	Light Yellow
4	Dark Blue	12	Dark Green
5	Blue	13	Magenta
6	Dark Red	14	Grey
7	Light Blue	15	White

> >

TOP THE TOPS:

Look, up in the sky! Is it an ORGROID? Is it EXERION?? Maybe it's just SUPERMOUSE!!! Whatever it is, we want to see your best efforts at getting the best out of these (and other) GAMES. Any better effort will be acceptable, but those verified by a PHOTOGRAPH ***, will surpass all others.

GAME**	SCORE**	NAME**	
BASEBALL	67	Danielle Anderson	
BOMB JACK	150,150	Meaghan Jenkinson	
BORDERLINE	429,610	Amanda Lawford	***
CHAMPION BOXING	K.O./Level 5	Jason Puschenjak	
CHAMPION GOLF	-11	M. Meehan	***
CHAMPION TENNIS	6-0	Jason Anderson	
CHOPLIFTER	180,500	Marcus Thompson	
CONGO BONGO	235,360	Sean Jenkinson	***
DEMON GOBBLER	53,550	Neuzat Ismaili	
DRAGON WANG	796,000	Stephen Emmett	***
EXERION	3,263,300	Malcom Chin	
GIRL'S GARDEN	51,130	Carly Pickett	
G.P. WORLD	1,008,560	Stephen Emmett	***
HANG-ON II	2,269,500	Stewart Crowther	
HUSTLE CHUMY	117,500	Jarron Hall	***
HYPER SPORTS	961,520	Jason Anderson	
LODE RUNNER	1,059,300	Richard Bolt	
MONACO G.P.	999,999	Adam Bolt	
NINJA PRINCESS	249,750	Craig Flowers	***
N-SUB	119,264	Dean Lyons	
ORGUSS	535,000	Michael Van Den Heuvel	
PACAR	201,342	Shane Gardner	***
PITFALL II	300,000	Stephen Emmett	***
		Sean Jenkinson	***
POP FLAMER	937,360	Malcolm Chin	
SAFARI HUNTING	69,230	Jason Anderson	
SAFARI RACE	74,720	Jane Anderson	
SEGA FLIPPER	1,450,500	Neuzat Ismaili	
SEGA GALAGA	172,390	Craig Flowers	***
SINBAD MYSTERY	269,350	Richard Bolt	
STAR JACKER	239,443	Brett McCallum	***
VERMIN INVADER	26,150	Danielle Anderson	
VORTEX BLASTER	395,290	Nelson Chin	
YAMATO	220,000	David Skene	***

< < < < < < < < < < < < < < < < < < < < < < < < <

Victorian Sega Users Group Inc.

☞ *Group News* ☜

As you would be aware, our AGM was held on 10th July , and at this meeting the following members were elected to be your committee for the coming year.

President: Stewart Crowther V.President: Garry Jenkinson
Secretary: Debbie Crowther Treasurer: Kris Flowers
Ordinary Committee: Ray Emmett Richard Moore
Nandi Juhasz Ray Finck
Doug Thornton
Hon. Editor: Michael Nanscawen

WHAT'S NEW:

- Discussions were held at AGM for possible change of our monthly meetings to Saturday afternoons, this has not yet been resolved and we will advise members should our current arrangements alter.
-Our editorial staff have a new printer, (an Epsom LQ500) which will enhance the quality of our magazine.
- New joysticks for the Mega System are suitable for our computers. - available at K-Mart.
-'Osi-Soft' have the Sega Mega System distributorship, and will check with them to see if any hardware/software can be obtained through them.
-CF2 Maxell disks are available at $63-00 a box from a stationery shop next to Dick Smith's in Elizabeth Street.
-A Bulletin board is being established for modem users - further info next magazine.
-Second tutorial course was held sucessfully at Frankston and another will organised soon. - Contact Stewart if interested.
-We believe Issue 3 of MEGACOMP will be issued soon - kkep an eye on the newstands.
-Total membership for year ending 30/6/89 reached 134.

- LAST MAGAZINE FOR 88/89 MEMBERS # # # # # # # # # # #

Application forms are enclosed for those members who have not renewed to date.

Until further advised - the following meeting dates have been arranged.
-7th August -4th September
-9th October -6th November
-4th December

Meetings are held at YWCA, 489 Elizabeth St Melbourne. 3rd floor - Rooms 7 & 8.
Time: 7-30pm to 10-00pm.

UTILITY:

....Mutatis mutandis....

Despite the apparently awful sounding nature of the title (*I had that once - but I got better...ED*), the contents of this month's *UTILITY* will not have you visiting your local GP!

After a protacted (perhaps too long) interlude, we have yet another assembly listing for all of the Machine Code junkies out there in SEGAland. This time around, the program on offer is a very short routine that allows the programmer to simultaneously store TWO (yes, TWO!!!) complete **SCREEN 2**s in memory and to interchange them at will by means of a simple **CALL**!! How can this be done????

The first of these graphic screens is stored in it's normal VRAM location. The alternate one is kept in RAM from address &HD000 ~ &HFFFF. With this little Utility immediately preceeding it. When this routine is **CALL**ed, the two areas of memory are transposed.

Some details of it's operation are worthy of note. This program is intended only for DISK BASIC. The assembly listing should be entered into memory using your ASSEMBLER and assembled at the indicated **ORiG**in address. This is also it's BASIC **CALL** address (though we should point out that the **ORG** address and the **CALL** address of an assembly listing are not necessarily synonymous). The BASIC statement **LIMIT** should be applied at the **ORG** address to protect this UTILITY and your valuable screen data. Likewise, the BASIC instructions **SAVEM** and **LOADM** can be used with great effect to preserve your precious screen data to and from disk almost instantaneously. Don't forget the proper SYNTAX for these statements which in this instance should be....

```
SAVEM"filename.DAT",&HD000,&HFFFF
```

or

```
LOADM"filename.DAT"
```

....where you should refer to your manual for further information. It is irrelevent which of the two of the SEGA's screens (**TEXT** or **GRAPHIC**) is the current display screen, this routine ALWAYS operates upon **SCREEN 2**. Next issue we will attempt to publish a BASIC version of this routine for Diskies and IIIB Cartridge

users (IIIA cartridges do not have sufficient FREE memory to accomodate this program - sorry!) so don't panic if you do not possess an assembler to enter this version.

Before we leave you to the program, we invite any Machine Coders to share with all of us any small programs or utilities that they may have devised and thus receive their well earned ($5.00 - *inflation has caught up at last...ED*) publishing fee. Now you can get to work....

```
                0002    TITLE   Screen 2 Swap
                0003
CFD4            0004    ORG     #CFD4
                0005
CFD4 F3         0006            DI              ;disable interupts
CFD5 210000     0007            LD      HL,#0000;start address of screen
                0008                            ;in VRAM
CFD8 1100D0     0009            LD      DE,#D000;RAM store start address
CFDB 010018     0010            LD      BC,6144 ;length of screen
CFDE CBAC       0011    LOOP:   RES     5,H     ;point to PATTERN data
CFE0 CDF0CF     0012            CALL    SWAP    ;swap data bytes
CFE3 CBEC       0013            SET     5,H     ;point to COLOR data
CFE5 CDF0CF     0014            CALL    SWAP    ;swap COLOR data
CFE8 23         0015            INC     HL      ;next screen address
CFE9 0B         0016            DEC     BC      ;decrement byte counter
CFEA 78         0017            LD      A,B     ;test byte counter
CFEB B1         0018            OR      C       ;is it zero?
CFEC 20F0       0019            JR      NZ,LOOP ;keep going if it ISN'T
CFEE FB         0020            EI              ;enable interupts
CFEF C9         0021            RET             ;back to BASIC
CFF0 CD2865     0022    SWAP:   CALL    #6528   ;set VDP read address
CFF3 DBBE       0023            IN      A,(#BE) ;read VDP data
CFF5 08         0024            EX      AF,AF'  ;save data in alternate
                0025                            ;accumulator
CFF6 1A         0026            LD      A,(DE)  ;get data from RAM
CFF7 CD3A65     0027            CALL    #653A   ;set VDP write address
CFFA D3BE       0028            OUT     (#BE),A ;write VDP data
CFFC 08         0029            EX      AF,AF'  ;retrieve VDP data byte
CFFD 12         0030            LD      (DE),A  ;store it in RAM
CFFE 13         0031            INC     DE      ;next RAM address
CFFF C9         0032            RET             ;back to main routine
D000            0033    END
```

PROGRAM:

Here is a small contribution to the multitude of programs thus far published for our delightful machine. A program, we feel that you can **RUN** whilst YOU contemplate Life, the Universe and Everything (your computer will be too busy to do anything else). All this notwithstanding, SYNTAX disciples will discover more creative ways to use **INT(** (already discussed) and **PSET** (yet to appear), for future reference....

```
10 REM "AC08A.BAS"
20 REM      A colourful bauble.
30 REM
40 REM              by
50 REM
60 REM          J.Guest
70 REM
80 SCREEN 2,2:COLOR 1,15,,15:CLS
90 CURSOR 75,95:COLOR 4:PRINT "A ";
100 COLOR 6:PRINT "COLORFUL ";:COLOR 2
:PRINT "BAUBLE."
110 FOR Z=1 TO 500:NEXT
120 COLOR 1,1,,1:CLS
130 POSITION (128,96),0,1
140 FOR A=1 TO 15
150 FOR X=0 TO 90
160 C=INT(2+RND(1)*14)
170 F=INT(95*RND(1))
180 B=SQR(8100-X*X)
190 IF F>B THEN 240
200 X1=0.7*X
210 PSET (X1,F),C:PSET (-X1,F),C
220 PSET (X1,-F),C:PSET (-X1,-F),C
230 IF F>B THEN 240
240 NEXT X
250 NEXT A
260 BEEP 2:BEEP 2
270 GOTO 270
```

```
10 REM  ****************
20 REM  *  FANCY WORK  *
30 REM  *      By      *
40 REM  *   Gwen Wood  *
50 REM  ****************
60 SCREEN 2,2:COLOR 1,15,,15:CLS
70 C=INT(RND(1)*12)+3
80 X=90:Y=50
90 DX=INT(RND(1)*3-1)
100 DY=INT(RND(1)*3-1)
110 Y1=Y:X1=X:GOSUB 290
120 X1=254-X:GOSUB 290
130 Y1=190-Y:GOSUB 290
140 X1=X:GOSUB 290
150 Y1=Y:GOSUB 290
160 X1=X*2:Y1=Y*2:GOSUB 290
170 X1=254-X1:Y1=190-Y1:GOSUB 290
180 X=X+DX:Y=Y+DY:IF X<0 OR X>126 THEN
 DX=-DX:GOTO 180
190 IF Y<0 OR Y>94 THEN DY=-DY:GOTO 18
0
200 A$=INKEY$
210 IF A$="?" THEN CONSOLE,,,,2:HCOPY
2
220 IF A$=" " THEN GOTO 60
230 IF A$="C" THEN C=C+1:IF C=15 THEN
C=2:GOSUB 290
240 IF RND(1)>6 THEN DX=INT(RND(1)*6-1
)
250 IF RND(1)>6 THEN DY=INT(RND(1)*6-1
)
260 DX=INT(RND(1)*3-1)
270 DY=INT(RND(1)*3-1)
280 GOTO 110
290 PSET (X1,Y1),C
300 RETURN
```

KEYBOARD MAESTRO:

....Cosmetic Micro Surgery....

Remember, (long ago though it was), how in the first instalment of this series, we promised we would provide you with ability to correct some mistakes that were made by the good fellows at MITEC? You don't!! No matter! You see, this month is where we are actually going to keep our word and correct this oversight.

"*But my disk BASIC works perfectly well!*", we here you all say! THAT'S what YOU think!!!! A question....

How many of you have tried to invoke the CONTROL CODE "N" in order to obtain the 'DIERESIS' character output from the keyboard? We appreciate that there would not be a great demand for this activity, but at least one or two of our readers must have tried. As it happens, ALL of those people have failed in this endeavour!!

Yes, readers, the table that appears on page 23 of the SF-7000 users manual, catagorically states that **[CTRL] + [N]** or **PRINT CHR$(14)** allows the operator to shift between the normal English characters and the Dieresis characters at will. That is, until you try to use it (yes, we know the shift works using the **ENG/DIER'S** key - but have you tried using that in a program, not to mention that pressure must be kept on this key for this to work??!!)!!!! Our varied trips through SEGA BASIC have discovered the reason why this is so and since fixing it would involve little more than a **POKE** here and a **POKE** there, we decided that we would make the required alterations. They appear here as part of this program.

So as not to have you feeling dissatisfied, we have also included an additional CONTROL CODE for this issue. The accompanying program will provide you with the facility to invoke a program **LIST**ing by simply combining a press upon the **CTRL** key, simultaneously with a press of the ']' key. Well may you wonder why we chose this most distant of keys, but regular **[FUNC]** users will immediately observe that this is the key that they use to invoke BASIC's Function print-out **LIST**. You may also wonder why we have provided a feature that already seems to be there. In fact, the main difference betwixt the **[FUNC] + [ ]** and our **[CTRL] + [ ]** is that BASIC will wait for you to enter either the line number of the program (which you wish to **LIST**) or the **[CR]** key (should you desire a complete list). On the other hand, our utility is intended to make a complete **LIST**ing more accessible and also provided for the invokation of a program **LIST**ing (e.g. **PRINT CHR$(27)**)

from within the program (though you should be warned that because of the routines used by the BASIC interpreter to provide the LISTing, you will not return to your program after the list is compltete (Ah, well. You can't have everything!!)

As with all of our previous *KEYBOARD MAESTRO*s, you should ensure that you enter the program correctly, reproducing the DATA there-in faithfully. You should ensure that you attempt to alter your BASIC disk ONLY (unlike we did whilst testing this program for you). All being well, this should now give your keyboard a little extra versatility and you can wait with bated breathe for the next issue of *SEGA ADDICT* where-in you will find the final episode of this series. Now for the program....

```
10 REM CTRL+(N)=CHR$(14)=ENG/DIER'S
20 REM
30 REM CTRL+(])=CHR$(27)=LIST
40 REM
50 REM    Disk BASIC ONLY!!!
60 REM
70 CLS:PRINT "Standby...":PRINT :C=0
80 READ D$:L=LEN(D$):IF D$="*" THEN 12
0
90 IF L=4 THEN AD=VAL("&H"+D$):C=C+AD:
GOTO 80
100 IF L=2 THEN DT=VAL("&H"+D$):POKE A
D,DT:AD=AD+1:C=C+DT:GOTO 80
110 PRINT "BAD DATA!!!!":BEEP 2:STOP
120 IF C<>132597 THEN 110
130 DSKI$ 0,1;A$,0,15
140 IF A$<>"SYS: disk BASIC" THEN PRIN
T "This is NOT a BASIC system disk!!":
BEEP 2:STOP
150 CALL &HF000
160 REM
170 DATA 518B,AA,518C,56,5691,1F
180 DATA 569A,B1,56AA,11,63,56,C3
190 DATA DD,04,5C98,1B,F000,CD,19
200 DATA 37,DA,44,2F,1E,00,21,22
210 DATA F0,01,06,03,C5,46,23,56
220 DATA 23,CD,85,36,DA,47,2F,C1
230 DATA 10,F2,CD,3E,38,C3,EF,03
240 DATA 02,51,07,56,0D,5C,*
250 REM
260 REM Please check all data
270 REM   very carefully!!!
```

NOW IN STOCK!!!

SQUATTER

The Great Australian board game is now available for your SEGA computer!

It doesn't matter if ewe like sheep, it doesn't matter wether you lose or win, this game is shear delight. Join the flock and fleece us for your copy today!!

SQUATTER SF-7000 ... $30.00 (disk)

SQUATTER SC-3000 ... $20.00 (tape)

SQUATTER

Send cheque or money order to:
Marketing Co-ordinator
P.O. Box 589
Mordialloc
Vic. 3195

```
10 REM ** The Revenge of the **
20 REM ** Toothless Vampire! **
30 AA=0:AB=0:AC=0:AD=0:AE=0:AF=0:AG=0:
AH=0:AI=0:AJ=0:COLOR 1,11:PATTERN C#14
4,"0020508850200000"
40 CONSOLE 0,23,1,0:CLS:P=2:X$="":PRIN
T CHR$(138);:FOR X=1 TO 36:PRINT CHR$(
129);:NEXT:PRINT CHR$(140);CHR$(128);"
The Revenge of the Toothless Vampire";
CHR$(128);CHR$(139);:FOR X=1 TO 36:PRI
NT CHR$(129);:NEXT:PRINT CHR$(141)
50 CONSOLE 4,19,1,0:DIM Q$(45),S(45,4)
,G$(20),B(20),N$(20),N(20),V$(5)
60 FOR X=1 TO 44:READ Q$(X):FOR Y=1 TO
 4:READ S(X,Y):NEXT Y,X
70 FOR X=1 TO 20:READ G$(X),B(X),N$(X)
:N(X)=X:NEXT:LN$=CHR$(10)+CHR$(42):FOR
 X=1 TO 36:LN$=LN$+CHR$(129):NEXT:LN$=
LN$+CHR$(42)
80 CLS
90 IF P=45 THEN 500
100 J$="I am "+Q$(P):GOSUB 1560:A$=""
110 IF S(P,1)<>0 THEN A$="North"
120 IF S(P,2)<>0 AND LEN(A$)>0 THEN A$
=A$+",South"
130 IF S(P,2)<>0 AND LEN(A$)=0 THEN A$
="South"
140 IF S(P,3)<>0 AND LEN(A$)>0 THEN A$
=A$+",East"
150 IF S(P,3)<>0 AND LEN(A$)=0 THEN A$
="East"
160 IF S(P,4)<>0 AND LEN(A$)>0 THEN A$
=A$+",West"
170 IF S(P,4)<>0 AND LEN(A$)=0 THEN A$
="West"
180 IF P=7 THEN A$="Down"
190 J$="I can go "+A$:GOSUB 1560
200 E=0:FOR T=1 TO 20:PP=0:IF B(T)=P T
HEN PP=1
210 IF PP=1 THEN 230
```

```
220 NEXT:GOTO 250
230 IF E=0 THEN J$="I can see "+G$(T):
GOSUB 1560:GOTO 220
240 J$=G$(T):GOSUB 1560:E=E+1:GOTO 220
250 PRINT "What shall I do now ";:INPU
T Z$:PRINT LN$
260 B$=LEFT$(Z$,2):C$=LEFT$(Z$,3):D$=L
EFT$(Z$,4)
270 IF B$="N" AND S(P,1)>0 THEN P=S(P,
1):GOTO 90
280 IF B$="S" AND S(P,2)>0 THEN P=S(P,
2):GOTO 90
290 IF B$="E" AND S(P,3)>0 THEN P=S(P,
3):GOTO 90
300 IF B$="W" AND S(P,4)>0 THEN P=S(P,
4):GOTO 90
310 IF B$="N" OR B$="S" OR B$="E" OR B
$="W" THEN PRINT "I can't go that way"
:GOTO 90
320 IF C$="CLI" OR B$="UP" THEN GOSUB
670
330 IF C$="DOW" THEN GOSUB 710
340 IF C$="IN" THEN GOSUB 800
350 IF C$="GET" OR C$="TAK" THEN GOSUB
 830
360 IF C$="SCO" THEN PRINT "This isn't
 a game you know!"
370 IF C$="PRA" THEN PRINT "That made
me feel better!"
380 IF C$="INV" THEN GOSUB 1080
390 IF C$="SPR" THEN GOSUB 1520
400 IF C$="HEL" THEN PRINT "I haven't
got a clue!"
410 IF C$="GIV" OR C$="DRO" OR C$="LEA
" THEN GOSUB 1120
420 IF C$="WAI" THEN GOSUB 1280
430 IF C$="PUL" THEN GOSUB 1330
440 IF C$="WEA" THEN GOSUB 1360
450 IF C$="REM" AND AI=1THEN PRINT "I
can't get them off my hands":GOTO 90
```

```
460 IF C$="REM" THEN PRINT "Don't be a
bsurd!"
470 IF C$="HIT" OR C$="SMA" THEN GOSUB
 1410
480 IF C$="UNL" THEN GOSUB 1470
490 IF P<>45 THEN 90
500 IF AD<1 THEN PRINT "I just remembe
red. I forgot my teeth!":PRINT "I run
back!":P=2:GOTO 90
510 CLS:J$="Well Done! You have helped
 poor old Cedric to solve this adventu
re":GOSUB 1560
520 END
530 DATA in the dentists surgery. Ther
e is a large chair in the middle,0,0,2
,0,in a dingy waiting room. A pile of
old magazines is piled on a chair in
the corner,0,9,3,1,in a small corridor
,0,0,0,2
540 DATA in a disused surgery. A tatty
 dentists chair stands in one corner,0
,0,5,3,in a narrow passage,0,6,0,4,at
the bottom of a steep flight of steps,
5,0,0,0,in a small sparsely furnished
room. Steps lead down,0,0,0,0
550 DATA in an overgrown garden,0,0,0,
9,in a doorway. A plaque on the wall r
eads 'Dr TOOTH...Dentist',2,15,8,0
560 DATA by a well stocked bar,0,0,0,1
1,in an elegant cafe,12,13,10,0,by a p
ile of tables,0,11,0,0,outside a small
 cafe. There's a bouncer on the door,0
,0,14,0,on the pavement at the side of
 a main road,0,0,15,13
570 DATA by a zebra crossing. The traf
fic is very heavy here,9,0,0,14,in the
 town centre. The heavy traffic stops
me crossing the road,0,21,17,0,outside
 a gunsmiths. It is closed,0,22,0,16,a
t the top of a steep cliff,0,0,19,0
```

```
580 DATA on a bracken covered hillside
,0,24,20,18,on a small plateau,0,0,0,1
9,outside the village blacksmiths. It'
s locked,16,0,22,0,by the side of a wi
de river. There is a drawbridge here,1
7,0,0,21
590 DATA on the banks of a river. Ther
e is a drawbridge here,0,28,0,0,in a t
hick fog,19,24,24,24,by a magnificent
alter,0,0,26,0,in an eerie crypt,0,31,
27,25,at the entrance to a magnificent
 temple,0,0,28,26
600 DATA walking between two rows of v
ery tall plants which look like triffi
ds,23,0,0,27,on a misty mountain top,0
,0,30,0,outside a strange castle. A so
ldier is to be seen on the battlements
,0,0,0,29
610 DATA at the top of some steps,26,3
4,0,0,at the bottom of some steps,0,35
,0,0,in a gloomy courtyard. Peasants a
re pushing coffins around.,0,38,0,0,in
 an antechamber,31,0,0,0
620 DATA in a dark cavern. Blood drips
 down the cobweb covered walls
630 DATA 32,0,0,0,at the bottom of the
 west tower,0,0,37,0,in a quadrangle.
There are vultures overhead,0,42,38,36
,in a passage. The walls are lined wit
h ancient sepulcres,33,43,39,37,in a p
assage lit by torches in the walls,0,0
,40,38
640 DATA by a large oak door. A wreath
 of garlic is fastened to the door,0,0
,0,39,in a large bedroom. There is a f
our poster bed in the corner,0,0,42,0,
in a library. There's a coat of arms
on the wall,37,0,43,41
650 DATA by a large tomb,38,0,0,42,in
a secret chamber,42,0,0,0
660 REM ** GO UP **
670 IF P=6 THEN P=7:RETURN
680 IF P=32 THEN P=31:RETURN
```

```
690 PRINT "I can't do that here!":RETU
RN
700 REM ** GO DOWN **
710 IF P=7 THEN P=6:RETURN
720 IF P=31 THEN P=32:RETURN
730 PRINT "I can't do that here!":RETU
RN
740 DATA an old copy of the 'Dentist's
 Gazette',2,GAZETTE,a glass of 'Bull's
 Blood' wine,10,WINE,a clove of garlic
,8,GARLIC
750 DATA a golden candlestick,25,CANDL
E,a crucifix,35,CRUCIFIX,a set of fals
e teeth,1,TEETH
760 DATA a pair of rubber gloves,7,GLO
VES,a large metal lever,22,LEVER,a lar
ge rock,18,ROCK,a large dancing skelet
on that I can't get past,40,SKELETON
770 DATA an angry dentist,3,DENTIST,a
giant lizard,42,LIZARD,a disco ticket,
5,TICKET
780 DATA a jar of lizard repellent,12,
REPELLENT,a large silver key,44,KEY,a
priest holding a giant crucifix,30,PRE
IST
790 DATA a large bible,35,BIBLE,a jar
of jam,11,JAM,a drill,4,DRILL,a paint
pot,37,POT,a peasant,37,PEASANT
800 IF P=13 AND AA=1 THEN P=11:RETURN
810 IF P=13 THEN J$="The bouncer says
'Where's your ticket ?'":GOSUB 1560:RE
TURN
820 PRINT "I can't do that just yet":R
ETURN
830 GOSUB 1030:IF L<1 THEN RETURN
840 E=0:FOR H=1 TO 20:IF B(H)=P AND B(
N(R))=P THEN E=1
850 NEXT:IF E=0 THEN RETURN
860 IF R=13 THEN AA=1
870 IF R=1 THEN AB=1
880 IF R=3 THEN PRINT "Vampire's can't
 carry garlic!":RETURN
```

```
890 IF R=4 THEN J$="A secret panel mov
es aside and I walk through it!":GOSUB
 1560:P=24:RETURN
900 IF R=5 THEN PRINT "I can't go near
 to a crucifix!":RETURN
910 IF R=8 OR R=10 OR R=12 OR R=11 OR
R=16 THEN PRINT "Don't be absurd!":RET
URN
920 IF R=2 THEN AC=1
930 IF R=6 THEN AD=1
940 IF R=7 THEN AE=1
950 IF R=9 THEN AF=1
960 IF R=14 THEN AG=1
970 IF R=15 THEN AH=1
980 IF R=17 THEN AJ=1
990 E=0:FOR D=1 TO 5
1000 IF V$(D)="" THEN V$(D)=G$(N(R)):E
=1:D=6
1010 NEXT:IF E=0 THEN PRINT "My hands
are full!":RETURN
1020 B(N(R))=0:RETURN
1030 L$="":FOR H=1 TO LEN(Z$)
1040 IF MID$(Z$,H,1)=CHR$(32) THEN L$=
RIGHT$(Z$,(LEN(Z$)-H)):H=80
1050 NEXT:R=0:L=0:IF LEN(L$)<2 THEN RE
TURN
1060 FOR H=1 TO 20:IF LEFT$(N$(H),LEN(
L$))=L$ THEN L=1:R=H
1070 NEXT:RETURN
1080 E=0:J$="I am carrying "+X$:GOSUB
1560:PRINT CHR$(30);:FOR H=1 TO 5
1090 IF V$(H)<>"" THEN J$=V$(H):GOSUB
1560:PRINT CHR$(30);:E=1
1100 NEXT:IF E=0 THEN PRINT TAB(15);"n
othing at all!"
1110 PRINT :RETURN
1120 GOSUB 1030:IF L<1 THEN J$="I can'
t see a "+L$:GOSUB 1560:RETURN
1130 E=0:FOR D=1 TO 5
1140 IF V$(D)=G$(N(R)) THEN V$(D)="":E
=1
```

```
1150 NEXT:IF E=0 THEN J$="I'm not carr
ying a "+L$:GOSUB 1560:RETURN
1160 B(N(R))=P
1170 IF P=3 AND R=1 THEN S(3,3)=4:PRIN
T "The dentist thanks me and lets me p
ass":B(1)=0:B(11)=0:RETURN
1180 IF R=1 THEN AB=0
1190 IF R=2 THEN AC=0
1200 IF R=6 THEN AD=0
1210 IF R=7 THEN AE=0
1220 IF R=13 THEN AA=0
1230 IF R=14 THEN AG=0
1240 IF P=30 AND R=17 AND C$="GIV" THE
N PRINT "The priest thanks me and lets
 me pass":G$(16)="a smiling priest":G$
(17)="":S(30,2)=33
1250 IF R=15 THEN AH=0
1260 IF R=17 THEN AJ=0
1270 RETURN
1280 PRINT "O.K.";:FOR X=1 TO 34:FOR H
=1 TO 50:NEXT:PRINT CHR$(46);:NEXT:CLS
1290 IF P=15 THEN PRINT "The lights ch
ange and I cross the road":P=16:RETURN
1300 IF P=16 THEN PRINT "The lights ch
ange and I cross the road":P=15:RETURN
1310 IF P=24 THEN PRINT "The fog lifts
":Q$(24)="on a misty hillside. There i
s a narrow path to the south":S(24,2)=
29:S(24,3)=0:S(24,4)=0
1320 RETURN
1330 IF P<>22 THEN PRINT "Not here!":R
ETURN
1340 IF AI<>1 THEN PRINT "AAAGGGHH. I
get an electric shock!":RETURN
1350 PRINT "The drawbridge comes down!
":S(22,3)=23:RETURN
1360 IF AE<>1 THEN PRINT "I haven't go
t anything to wear!":RETURN
1370 IF AI=1 THEN PRINT "I'm already w
earing them!":RETURN
```

```
1380 AI=1:FOR X=1 TO 5:IF V$(X)=G$(7)
THEN V$(X)=""
1390 NEXT:X$="A pair of rubber gloves
... worn! PLUS "
1400 PRINT "O.K.":RETURN
1410 IF P<>40 THEN PRINT "I can't do t
hat here!":RETURN
1420 IF AF<>1 THEN PRINT "I have nothi
ng to do that with!":RETURN
1430 PRINT "The skeleton falls in a pi
le of bones!"
1440 FOR X=1 TO 5:IF V$(X)=G$(9) THEN
V$(X)=""
1450 NEXT
1460 AF=3:G$(10)="a pile of bones":RET
URN
1470 IF P<>40 THEN PRINT "Don't be rid
iculous!":RETURN
1480 IF AF<3 THEN PRINT "I can't get p
ast the skeleton":RETURN
1490 IF AH<1 THEN PRINT "I haven't got
 the key!":RETURN
1500 PRINT "I open the door":Q$(40)="b
y an open door":S(40,2)=45
1510 RETURN
1520 IF AG<1 THEN PRINT "I haven't got
 any repellent!":RETURN
1530 IF P<42 THEN PRINT "There's not m
uch point in that here!":RETURN
1540 PRINT "I spray the repellent and
it moves     aside":S(42,2)=44
1550 RETURN
1560 CO=38
1570 IF LEN(J$)<38 THEN CO=LEN(J$)+1:G
OTO 1590
1580 IF MID$(J$,CO,1)<>CHR$(32) THEN C
O=CO-1:GOTO 1580
1590 PRINT LEFT$(J$,CO-1)
1600 IF LEN(J$)<38 THEN J$="":PRINT:RE
TURN
1610 J$=RIGHT$(J$,LEN(J$)-CO):GOTO 156
0
```

REVIEW:

Girl's Garden

Don't the girl's just love a bit!!! A bit of time spent playing at this delightful non-violent, peace loving (perhaps the one and only) computer game, I mean (*what else did you have in mind...ED*)!!. Why is it that the more popular computer games (especially when one considers that most computers [and the programs written for them] are operated (read 'played') by males) are the ones that exhibit the most macho and masochistic tendencies??? I don't mind admitting that I enjoy an occasional respite from the hectic, let's keep up the image, show no emotions pretence required by the general tendency of (male) society, and on such occasions, I indulge in a round or two (or three) engulfed in the stress dispersing ecstacies of this (need I say) delightful game.

So what is it that has me in raptures (well, near paroxisms) of bliss? It is nothing more than one of the more recently released games for the SEGA that has managed to find it's way into my possesion! This issue, the review column has decided to throw all semblence of masculinity to the four winds and admit that (just maybe) it isn't as virile as the image it presents to it's readers! Yes, I really enjoy (I don't mind saying) playing this marvellous game!! Ooops, I haven't told you which one it is yet, have I?? Well, to keep you out of suspense, I am talking about **GIRL'S GARDEN**!!!

Don't get me wrong, my wife hasn't threatened to leave me yet (at least not because of any preference I show for this game - she likes it too)!! Not only that, and don't you DARE misinterpret my seemingly condescending attitude to imply some measure of chauvanism (*heaven forbid...ED!!*). I will say that the thought that has been put into this program has resulted in a game which can be played by a multitude of persons regardless of race, creed, colour or SEX! By this stage you must all be waiting with bated breath for the usual dissertation upon the playing procedure. Who (whom??) am I to stand in your way???

So, down to the nitty gritty (*they were a good band...ED!*)! This is a game where the player pits his/her wits against the might of the all powerful **TEDDY BEAR**! I bet you all thought that they were meek little timid creatures who occupied their time having picnics in the woods! Needless to say, everyone has been labouring under a gross misapprehension for all these years and this program is certain to destroy any patronising image you may have had of these creatures!! It is within the confines of this program (*fortunately...ED*) that these apparently hapless creatures become a force with which to be

reckoned. I shall ellucidate (*why don't you explain yourself somewhat more clearly, instead?...ED*)....

Girl's Garden is set within the confines of the (rather large) garden which belongs to the **BOYFRIEND** of the **GIRL** who is the character representing the player. In the middle of this garden is his house. The object of the game is to get your **BOYFRIEND** to come out of this house. In order to do this, you must entice him to leave with the offer an enormous bunch of **FLOWERS** (*a reversal of thousands of years of social development...ED*). It is collecting these **FLOWERs** which occupies much of the time needed to play this game. This all sounds extremely easy, however, there are a number of obstacles put in the player's path (pun intented) to make this task more difficult. Firstly, the **FLOWERs** have four stages of growth. They start out as shoots, then grow into buds, after which they bloom into full flower and then eventually die and dissappear. The player can collect the **FLOWERs** at any time, however it is only if they are collected whilst they are blooming that they will be added to the required bunch. This effort also collects you 100 points. If you should collect the **FLOWERs** before this time, you will only obtain 10 points for each of them. Should you collect a **FLOWER** that has expired, you will suffer the penalty of the loss of 4 **FLOWERs** from the bunch which you have already collected. Thus, it will take you longer to collect the 10 which you need to entice your **BOYFRIEND** out of his house. This is fine as long as you do not take longer than the time it takes for your **BOYFRIEND** to move across the top of the screen towards the other **GIRLFRIEND** in his life. If he reaches her, *YOU* die of a **BROKEN HEART!!**. Bad Luck, sunshine!!!!

The player suffers the same fate should you find yourself needing more than the prescribed 'lives' which you are granted at the commencement of play. Well, I use the term 'lives' lightly, in actual fact, the player doesn't have 'lives' in this game. He/she has 'loves', three of them, no less (*were real life so good...ED*). The loss of these is depicted as the loss of a **HEART** as the game progresses.

Besides these impediments, numerous snags are strewn throughout the **GARDEN.** These make matters more difficult. They include such things as the **ROCKS**, around which you must search, to find the **FLOWERs**, a number of large **POOLS** (and wouldn't you know it, SHE can't swim) and not to mention the numerous **TEDDY BEARS** sent to confuse your best efforts. These multiply at an incredible rate depending upon the level of play which is reached (I love the little wiggle the **GIRL** gives to shake off the water after she gets wet).

There is, however, one small advantage available to

the player! The **TEDDY BEARS** are susceptible to the contents of the 5 **HONEY POTS** which the player is granted at the inauguration of the game. This advantage is soon depleted, unless the player is fortunate to ascertain that the (almost) benevolent **BEE** has landed upon a **FLOWER**, and thus made possible an increase in that advantage. Mind you, this will only occur if the player should collect this **FLOWER** whilst the **BEE** is still devouring the nectar thereon!!

This leads me to discuss a number of other advantages which this **BEE** has to offer to the player. Throughout the duration of the play, the **BEE** will deposit certain items randomly within the **GARDEN.** These include a **CLOCK** (*have scientists investigated this aspect of the species...ED?*), a bunch of **GRAPES**, a bunch of **CHERRIES**, an **APPLE**, a **HEART** and a **SKULL.**

The **CLOCK**, should it be collected, gets the player more time to explore the garden and obtain the required number of **FLOWERS**. Collecting the **GRAPES** obtains for the player 500 points. Should the player procure the **APPLE**, the score will increase by 1500 points. If the player should be fortunate to have the **BEE** drop the **CHERRIES** within their grasp, the score will be increased by the addition of 2500 points. The most difficult to obtain is the **HEART**, which will procure for the player an EXTRA LIFE (read **HEART**). Which only leaves the **BEE** to deposit the **SKULL** and cause the player to lose a **HEART** ,and thus a life, should that item be stumbled upon during the play. This should be avoided at all costs!!

Once the full compliment of 10 **FLOWERS** have been gathered, the player must make 'her' way to 'her' **BOYFRIEND**'s house, where 'she' will find him awaiting 'her' arrival. The only obstacle here is getting into the confines of his 'property'. Once this has been achieved, the next round is attempted.

In this round, (and the third round for that matter), the player must traverse the garden collecting the requisite number of **ROSES** (**DANDELIONS** in round 3). When this has been achieved, the player must undertake the **CHALLENGING ROUND**, which involves the vaulting of a number of the everpresent **TEDDY BEARS.** The more of these that can be leaped, will increase the bonus obtainable at this time. The player then continues playing the game from the 'start' with a comensurate increase in the number of obstacles which must be overcome. I reckon that that's enough to tell you without giving away too many secrets and spoiling the enjoyment that can be extracted from this 'fun' game.

Throughout all of this there is a most delightful tune playing to keep the player in the right frame of mind. RECOMMENDED for all....

WORKSHOP:

....The SN76489AN S.G.C...

OR

....*another way to slice your chips*....

The Texas Instrument's **SN76489AN** Sound Generation Controller is a data based I/O peripheral (*go and wash your mouth out!!...ED*)!!! A what??!!!! you all ask!!! This is the official name and description of the little sixteen legged beastie which runs about inside the SEGA making lots and lots of noise!!

Frivolity aside (well, not too far aside), our dissertation for this month will involve, as if you might not have gathered already, an in depth confrontation with our SEGA's sound chip. This marvel of technological engineering (NOTE this Texas Instruments), is a device which includes three programmable tone generators (allowing simultaneous sounds), a programmable white noise generator and programmable attenuation (*I did that at school once...ED*) which runs at 3.58 MHz (as far as we are concerned). This monstrous animal is connected to the insufferable Z80 microprocessor through that chip's output port **&H7F**. It is only possible to talk to it through this port (it doesn't 'talk' back). This chip (the **SN7689AN**) is tied to the Z80's 'WAIT' line which effectively means that the Z80 is forced to do nothing for 32 clock cycles until the data for the **S.G.C.** has been latched by the sound chip.

Each of the three tone generators has within it's jurisdiction, a frequency synthesis section (*is that anywhere near the string section?...ED*) and an attenuation section (which provides for a variable volume control). These frequency synthesis sections require a data input comprising a 10 bit binary value which corresponds to a value that is exactly one half of the interval of the required wavelength. This binary number is placed into an interval counter which is decremented at a rate equivalent to the SEGA's clock frequency (3.58 MHz) divided by 16. When this counter reaches zero, an internal mechanism reverses the polarity of the signal to the speaker, at which point, the tone counter is refreshed (with the original binary number) and thus the frequency produced is twice the value of the input provided (*two for the price of one???...ED*). Are you

still with us, or have you gone to make a cup of tea?? If you haven't we suggest that you try a Scotch instead, you ain't read nothin' yet!!!

As wonderful as this device may sound, there are a few limitations to it's abilities. Not the least of which are the upper and lower frequency range limits. these proportions reside between approximately 110 Hertz and up to over 100,000 Hertz. For the uninitiated, 1 Hertz is the value assigned to a frequency of EXACTLY one cycle per second. This desired frequency can be obtained using the following formula....

INT(3580000/(32*f))

....where 'f' is the desired frequency expressed exclusively in HERTZ (that didn't *hertz*, did it??)! This resultant decimal value should now be converted into a ten digit binary number. This, of course, should not exceed the hexadecimal values 0 to &H3FF (0 to 1023) which provide the above mentioned Hertz capablities.

....Turn that bloody thing down!!....

The volume of the frequency generated by this 10 bit binary count is controlled by a 4 bit attenuator. Using 4 bits enables the programmer to achieve 16 combinations (&H00 to &H0F) of output volume level, where 0 is the loudest possible and 15 is off (were it that all these modern musicians worked this way?? - *are you having a dig at me??...ED*). Perceptive colleagues will notice that this arrangement is contrary to our normal thought processes. Whereas, the smaller value (to our minds) represents a softer sound, in fact, in the case of the **SN76489AN**, it corresponds to a lower ATTENUATION and thus, a louder output, and vice versa!!

...."I'm Dreaming of a...." ....

The third of the generators within this little instrument, which is available to the programmer/ess, is the periodic/white noise generator. As is the case with 'white light' (which is an amalgamation of all the frequencies of the visible spectrum), white noise is, similarly, a combination of all of the audible frequencies within the range of this P.S.G. (for the uninitiated, this is the sound of frying eggs - or that more familiar hiss created by the television when you wake up at 0400 Hrs and every-one at the ABC have gone to bed)!!! Technically, this generator

is the Noise Feedback Controller! What this means is that you, the programmer, can control the output of the S.G.C. by the simple expedient of setting the FB (or FeedBack) bit. Reset to zero, the output from this generator will be in the form of 'periodic' noise, whilst setting this bit to one, will cause the noise generator to produce 'white' noise. Three distinguishable 'Periodic' noises are available to the programmer of this device. These are noises produced by dividing the SEGA's clock cycle with the values 512, 1024 or 2048, depending upon the desired result. Purely and simply, the 'periodic' noise can be considered as 'rythmic' noise, whilst 'white' noise may be imagined as 'random' noise. This table may make things a little clearer (then maybe it won't - *as long as there's food on it I'm not fussy...ED*)....

TABLE 1

Noise frequency Bit 1	Bit 0	Shift rate
0	0	clock/512
0	1	clock/1024
1	0	clock/2048
1	1	*

* when this setting is used, the Shift Rate is calculated according the frequency that is set by the output to Channel 3.

For our purposes, however, the output frequencies must remain within the range of the chip. More on this later (*How's that Scotch lasting???...ED*)!!!.

Now that you know how it all works, we feel that it is time to show you how to make use of this new-found knowledge.

....Theory put into Practice....

The S.G.C. (not to be confused with the S.C.G.) contains a total of 8 dedicated registers that are used to control the 3 tone generators and the solitary 'noise' generator. To update a frequency to any one of these 3 tone generators, requires two bytes of data presented to the chip in the correct format (like all distinguished persons, it must be addressed properly). Bit 7 of the first byte MUST set, i.e. '1' (or 'ON'). The following

3 bits (Nos. 6, 5 and 4) determines the register to which the data will be despatched (one of the 8 we mentioned earlier). The least significant nibble of the byte contains the lowest four bits (bits 0 through 3) of the 10 bit binary number which represents the chosen frequency. For the second byte of data sent to the chip, the first (or most significant) bits (bits 7 and 6) SHOULD be 'OFF' (zero). The remaining six bits make up the balance of the most significant bits (bits 4 through 9) of the ten bits of the selected frequency calculation. We expect you'll want a demonstration, so we came up with this....

TABLE 2

BYTE 1

Bit No.	7	6	5	4	3	2	1	0
	1	R0	R1	R2	D3	D2	D1	D0

BYTE 2

Bit No.	7	6	5	4	3	2	1	0
	0	0	D9	D8	D7	D6	D5	D4

....where D0 ~ D9 are the 10 bits of the binary data to determine the desired frequency and bits R0 ~ R2 are allocated according to this table....

TABLE 3

R0	R1	R2	Destination Register
0	0	0	Tone 1 freq.
0	0	1	Tone 1 Volume
0	1	0	Tone 2 freq.
0	1	1	Tone 2 Volume
1	0	0	Tone 3 freq.
1	0	1	Tone 3 Volume
1	1	0	Noise control
1	1	1	Noise Volume

To control the noise source requires a slightly different beat on the drum!! This (you'll be pleased to learn) only neccesitates

the output of only ONE byte!! (yeah!!!). Again, bit 7 of this byte must be set ('ON'), the next 3 bits can be set in accordance with the above table. Observant readers will immediately realise that this nibble MUST be 1110 (in binary) or &H0E (in hex). The lower nibble of this byte must have it's most significant bit reset, the remaining three bits are allocated as bit 2, for the FB bit and bits 1 and 0 according to the Noise Frequency bits ('NF' bits - as per table 1). Exempli gratia....

TABLE 4

Bit No.	7	6	5	4	3	2	1	0
	1	R0	R1	R2	0	FB	F1	F0

Now, we move onto updating the attenuation register (*I took one of them out once...wasn't very exciting...ED*). This single byte of data follows the pattern set by the above table (table 4), as far as the most significant nibble is concerned. The least significant nibble (bits 3~0) contains the four bits of attenuation value (volume) as we discussed earlier, not forgetting, that the higher the value provided there (&H00~&H0F), the softer will be the output from the sound chip.

There is a neat little trick that the **SN76489AN** keeps up it's sleeve to make things a wee bit easier for the programmer. This is the small but significant ability, whereby the chip will 'latch' the 'channel' (or register) provided by the first byte of frequency data, causing that register to remain set thus allowing the second byte (the six Most Significant Bits - MSB - of the binary value of the frequency) to be continuously (read 'rapidly') updated without further reference to that register.

....Getting down to BASICs....

As we mentioned at the start, this sound device (*does this imply that it is reliable??...ED*) is connected to the Z80 microprocessor via that chip's port **&H7F**. Fortunately, it is not neccessary for any of our disciples to be fully conversant with Machine Code to be able to program the **S.G.C.** You see, this chip can be fully accessed directly through BASIC by use of the BASIC statement **OUT**port (the only difference between this access and M/C is one of speed)!! One of the good things about this is that one can experiment with values to this chip without damage of any kind to any BASIC program or the chip itself (except to your mum's ears)!! If you don't like

the noises that your SEGA makes, **SOUND 0** will kill it all in a flash!!! It is worth noting that this technique is similar to using **VPOKE** to control sprites (as opposed to the **SPRITE** statement), giving the programmer greater versatility than the corresponding BASIC instructions. In order to appease the programmers amongst you, we offer this small demonstration....

```
10 OUT (&H7F),&H8B:OUT (&H7F),&
H1A:OUT (&H7F),&H90
20 FOR Z = 1 TO 1000:NEXT Z
30 OUT (&H7F),&H9F:END
```

....wherein, line 10 sets up channel 1 to 'sound' middle C at maximum volume. Line 20 causes a simple delay. Line 30 then, and this is worth noting, does nothing more than turn off the volume (the frequency is still set). This would be the equivalent to BASIC's statement....

```
SOUND 1,262,15
```

We leave you to have lots of fun playing with this playful contrivance. May the **FORTISSIMO** be with you!!!!

ƒ≈

FILES:

****

Sega Help Line:

If you have any urgent problems with your computer, or wish to find other Sega users in your area then call one of these members:

Garry 309-7130
Stewart Crowther 783-7328
They will be happy to help you.

Marketing Co-ordinator
P.O. Box 589
Mordialloc Vic. 3195

****