# Sam Coupé

# ARCADE DEVELOPMENT SYSTEM

## GLENCO SOFTWARE

## IMPORTANT NOTICE

# TABLE OF CONTENTS


Table of contents
General Introduction....1
Designer....3
Supervisor....4
Requirements for Running SCADs....5
Working with Masterdos....6
Games Designing Theory....7
Coordinate System....10
Introducing Sprite Windows....11
Introducing Animation....13
More about Scenery....15
Introducing Masking....17
Introducing Sprite Planes....19
Introducing Nodes....21
Introducing Missiles....34
Introducing Character Sets....37
Introducing Sound Effects....38
Using the Designer Program....42
Loading the Designer....42
Keyboard....42
Sam Mouse....42
Blue Alpha Mouse....43
Joy-stick....43
MAIN MENU....44
IMAGE....46
Image / create....46
Design / abort....48
Design / wipe....49
Design / move....49
Shift / up....49
Shift / down....49
Shift / left....49
Shift / right....49
Design / save-quit....49
Design / dimensions....50
Design / effects....50
Effects / mirror....50
Effects / invert....51
Design / copy....51
Image / view....52
Image / ani - test....52
Movement / up - down - left - right - still....53
Image / imp - exp....54
I-O / import....54
Colour selection / imported colours....55
Colour selection / current colours....55
Graphics grabber / standard grabber....56
Graphics grabber / grab then clear....56

Graphics grabber / disc operation .....56
Graphics grabber / main menu.....56
I-O / export.....57
Export / return.....58
Export / abort.....58
Export / save.....58
SCENERY.....60
Scenery / create.....60
Scenery / view.....62
Scenery / i - o.....62
I-O / import .....62
I-O / export.....62
ROOM DESIGN.....63
Screen option / initial settings .....63
Screen settings / screen dimensions.....64
Screen setup / x start.....64
Screen setup / x finish .....64
Screen setup / y start.....64
Screen setup / y finish .....64
Screen settings / border selection.....65
Screen settings / screen filename.....65
Screen filename / choose filename.....65
Screen filename / cancel and clear.....66
Screen option / screen edit.....66
Screen / edit.....66
Scenery / place.....66
Scenery / move.....68
Scenery / erase.....68
Scenery / highlight.....69
Scenery / view.....69
Scenery / destroy.....69
Room Design:Hints.....70
Screen / view.....71
Screen option / node edit .....71
Nodes / place.....71
Define node / clear data.....73
Define node / jump.....73
Define node / fire.....73
Define node / speed.....73
Define node / remove.....73
Define node / place.....74
Nodes / move.....74
Nodes / erase.....74
Nodes / alter.....75
Nodes / view.....75
View / jump.....75
View / fire .....75
View / speed.....76
View / remove.....76
View / place.....76
COLOURS.....77
FILE.....79

Disc / load ... 79
Disc / save ... 80
ANIMATION ... 82
Sequence no / direction ... 82
Range / clear ... 83
Range / alter ... 83
Sequence no / test ... 83
Sequence no / save ... 84
Sequence no / abort ... 84
JUMP OFFSETS ... 85
Jump stage / enter ... 86
Jump stage / abort ... 87
Jump stage / save ... 87
Jump stage / edit ... 87
Stage edit / delete ... 88
Stage edit / change ... 88
KEY DEFINITION ... 89
CHARACTER SET 1 ... 90
Text1 / create ... 90
Text 1 / view ... 91
Text 1 / type-test ... 91
Text 1 / imp-exp ... 91
SOUND EFFECTS ... 92
Sound effects / tone envelopes ... 92
Tone env / enter ... 93
Tone env / edit ... 94
Edit stage / alter ... 94
Edit stage / delete ... 94
Tone env / save ... 94
Tone env / abort ... 94
Sound effects / volume envelope ... 95
Sound effects / sound commands ... 95
Sound edit / chan ... 95
Sound command / octave ... 96
Sound command / tone ... 96
Sound command / length ... 97
Time options / fixed time ... 97
Time options / repeat rate ... 97
Sound command / volume ... 97
Sound command / noise ... 97
Sound command / vol env ... 98
Sound command / tone env ... 99
Tone / repeat ... 99
Tone / fixed ... 100
Playing / Saving a note ... 100
Sound options / save sound ... 100
Sound options / erase sound ... 101
Sound options / play sound ... 101
Sound options / play scale ... 102
CHARACTER SET 2 ... 103
Text2 / create ... 103
Text 2 / view ... 103

Text 2 / type-test....103
Text 2 / imp-exp....103
MISSILES....104
Missile firer menu....104
Missile image menu....104
Missile offset selection....105
Direc menu....105
Direc / x speed....105
Direc / y speed....105
Missile / save....106
Missile / erase....106
Missile / edit....106
Using the Supervisor program....107
Loading the Supervisor....107
Entering SCADs Commands....108
Entry errors....108
Run time errors....108
Getting Started....109
Writing a Program....112
Permanent Sprites....118
Platform Sprites....119
Door Sprites....122
Missile Sprites....123
Joy-stick / Keyboard Sprites....127
Collision Detection....130
Printing Text and Numbers....132
Printing text....132
Printing numbers....132
Using Sound Effects....134
Autosound....134
Scorn....134
Note....134
AMMADD....135
AMMO....135
ANIMATE....136
ANIMOFF....136
AUTOSOUND....138
BOUNCE....138
CLINK....139
COLOUR....139
COPYSCREEN....140
CPLANE....140
CTEXT....141
DJCLEAR....141
DOOR....142
DROP....143
FEET....144
FILE....144
FIRE....145
FULLCLEAR....146
FULLWIPE....146
GETINP....146

HIT..... 147
INIT..... 148
INKBLACK..... 149
JUMP..... 149
LINK..... 150
MAINSCR..... 150
MISSHIT..... 151
MISSILE..... 152
MISSOUND..... 153
MISSRANGE..... 154
MOVEALL..... 154
MOVER..... 155
MOVES..... 155
NEXTHIT..... 156
NODEOFF..... 157
NODEON..... 157
NOTE..... 158
PANEL..... 160
PARTCLEAR..... 160
PARTWIPE..... 161
PCLS..... 161
PLATFORM..... 162
PLATSPEED..... 162
PNUMA..... 163
PNUMB..... 164
PSPUT..... 164
RAMMO..... 165
RIMAGE..... 166
RESTDJ..... 166
ROOM..... 167
SATTR..... 167
SCOM..... 168
SETINP..... 168
SMERGE..... 170
SPEED..... 171
SPUT..... 172
SWINDOW..... 173
SXPOS..... 174
SXSPD..... 174
SYPOS..... 174
SYSPD..... 175
TEXTA..... 175
TEXTB..... 176
TOGGLE..... 176
VIEW..... 177
XEDGE..... 178
YEDGE..... 178
Command Summary..... 180
Technical Information ..... 182
WHAT IS NEXT ? ..... 193
Error Message Codes..... 194

# GENERAL INTRODUCTION

Congratulations on purchasing the Sam Coupe Arcade Development system. I know you will not be disappointed with the extra power and ability this suite of programs will bring to you and your computer.

Why the Sam Coupe ?

The Sam Coupe is supplied with an excellent hardware specification, large memory, fast processor and high resolution graphics, an ideal computer for both serious and entertainment software.

Sam Basic is both extremely fast and very comprehensive, the graphics commands can zip along at a fair old pace. However, there is only so much that can be achieved within the 32k that is available in ROM. All of the BASIC programming language and graphics commands have to be written into the ROM, there is simply no space left to add advanced sprite commands.

The Sam Coupe Arcade Development System (SCADs) will extend the BASIC with over 100 new, powerful commands. These commands will allow the user to manipulate graphics and sound to a far greater extent than was ever possible using BASIC. Indeed you should be able to write arcade quality games within a few weeks of using this product.

## A Brief History !

The games development system has taken a number of years to get to the stage it is at today. The original concept programs were written in 1983 on a computer called the Dragon 32. As I developed the routines and improved upon them, I decided I needed a faster computer, and so Sprites Alive was written for the Amstrad CPC6128. The program has continued to develop into the form you see here.

If you compare this version of the program to the last version (Sprites Alive) you will hardly recognise it. The Sam Coupe, with its extra power and vastly superior memory has allowed me to add all of the features that I longed to, but couldn't because of lack of memory. I believe this is the most advanced games designing system on the market and I hope you will agree.

## How does the SCADs system work ?

The whole of the SCADs system is based on 'Sprites'. A sprite is a character, a shape that can be placed onto the screen. Unlike the normal characters ( A,a,Z,z ) that are printed onto the screen in a single colour and a single size (8 x 8), the sprite characters can be any size you require (up to a maximum of 32 x 32) and can contain up to 16 colours. But, I hear you say, "you can do that with the Sam BASIC GET and PUT commands", and indeed you can. However that is just the start of what a sprite can do. You can program Sprites to move around the screen, with animation dependent upon direction. The Sprites can bounce, disappear and explode. The collision detection can tell you exactly which Sprites are in collision. The Sprites can be programmed to fire missiles at each other, they can also be instructed to chase or run from each other.

They can walk in front of, or behind scenery. They can even be programmed to make their way around complex mazes, in fact, they can do anything you want them to.

You may be thinking you will need a degree in computer programming in order to achieve all of the things that only professional machine code games programmers could normally do.

**YOU ARE WRONG**.

Glenco Software has spent many years writing, re-writing and improving all of the difficult machine code routines to make them as fast and as smooth as possible. The machine code has been written so that you are not required to do much programming, everything is done as automatically as possible.

After writing all of the machine code routines, I then wrote the link routines that would allow you to access all of the lightning fast routines from simple BASIC. This means you require no knowledge of machine code at all. In fact the only language you need to understand is simple BASIC.

You may be getting a bit impatient to find out just how fast and powerful SCADs BASIC really is, well wait no longer. Supplied on main program disc are a number of demonstrations of the capabilities of this package.

To run the demonstration programs, turn your computer on. Insert the SCADs master disc and press the **f9** key. Wait momentarily and the DESIGNER/SUPERVISOR options screen will be displayed.

Select option 2, 'Supervisor'. The SCADs Supervisor will now load into memory. Once the Supervisor has loaded, the computer will reset. Don't be alarmed, this is perfectly normal.

Get a directory of the disc, all of the basic demonstration programs start with the filename "DEMO". ie DEMO1, DEMO2 etc. At the time of printing this manual it is not known how many demonstrations will be supplied on the disc. Looking at the directory listing will tell you how many demonstrations we have supplied.

To load your first demo program type LOAD "DEMO1". After pressing return, the first SCADs BASIC demo will load into memory.

The demonstration programs have been saved so that they will automatically run once they have been loaded.

You may break into the demonstrations at any time by pressing the **ESC** key. You are now able to list the programs. All of the SCADs commands are prefixed by a small solid square.

Once you have looked at one demonstration, you may load the next demonstration by simply typing LOAD "DEMO2".

The SCAD system consists of two main programs, the Designer and the Supervisor.

## 2 Introducing SCADs

# DESIGNER

The Designer program will allow you to create all of the image and scenery graphics that you will require.

You can design up to 255 different images for use with Sprites, and you can design up to 255 drawings for use as scenery.

You can create up to 64 different animation sequences. Animation sequences will allow you to animate all of the sprites that are printed on the screen. What's more, each sprite can have a different set of animation images for every different direction the sprite can move. If you animate a man walking around the screen, as soon as the man changes direction, the animation will change to follow suit.

The Designer allows you to select 16 colours from the full palette of 128 (all 128 colours are displayed on the screen at the same time to allow easier selection).

You can program each of the sprite images to fire missiles, each missile can be programmed to fire in a certain direction, depending on the direction in which the sprite is facing. The missiles can be programmed to move at varying speeds and for varying distances. The missile can be programmed to explode on impact, or to simply be removed from the screen.

You can link together the various pieces of scenery together to create a Room.

You may design up to 255 different rooms for use within your programs. Rooms may consist of up to three types of scenery. Sprites will be able to pass over, move under and bounce off items of scenery, all fully automatically.

Within the rooms you have designed you can either create a series of paths for the sprites to follow, or you may insert hidden instructions which will cause the sprites to perform certain actions. ie fire a missile. The paths or instructions are called screen nodes.

You can program a full range of sound effects. The Designer's sound option allows the novice user to access the powerful Sam sound chip with absolute ease. You will be able to create complex volume and tone envelopes, and then link the envelopes together to create exciting sound effects. The sound effects are all handled fully automatically by the Supervisor using interrupts. This means that the sounds play even when the computer is doing another task.

If you have created graphics using Flash or any other Sam art package, you can import them into the SCADs Designer using the graphics grabber facility.

There is a special program supplied on the system disc called SCRCONV. This program will convert any flash files into a format that the designer can understand.

You may also design two different character sets for use within your programs, the characters are special, in that, each character may be up to 32 x 32 pixels. The characters will be printed to the screen proportionally.

# SUPERVISOR

The Supervisor program, once loaded into the Sam Coupe, will extend Sam BASIC with over 60 new commands.

With the SCADs Supervisor loaded into memory you will be able to write your own games using both Sam BASIC and the new extended SCADs BASIC.

The SCADs Supervisor will then allow you to load in you the data you created within the Designer and to utilise this data to create full action arcade games. All of the information you have created within the Designer program (sprite images, rooms, nodes, animation, text, sound effects and missiles) can be easily accessed via simple SCADs BASIC commands.

SCADs BASIC is both easy to use, and also extremely powerful. One simple command will allow all of the Sprites that are placed onto the screen to move around, bouncing off walls and scenery, travelling behind or over the top of objects, all completely automatically. You have total control of every sprite that is displayed on the screen, and there may be up to 64 of them H

Complete screens can be displayed from one keyword. You will not believe the power of the SCADs Supervisor until you have used it.

## News Flash

If we have added any extra features to the SCADs package after this manual has been printed, then this information will be stored on the master disc in the form of a program called "readme". To see the very latest information on the SCADs package, simply type RUN "README", this program will then display on the screen any new information that may be missing from the manual. Don't worry if there is no readme file. This simply means that there is no new information to read.

Glenco Software have a continual upgrade policy, if we write any new features into the SCADs package, then you may upgrade to the new version for a small fee. In order to operate this service we will need to keep a register of all the people that are currently using SCADs. Please return your registration form with your name and address, and we will keep you up to date with the very latest news on SCADs.

# REQUIREMENTS FOR RUNNING SCADS

1 The Sam Coupe Arcade Development System will work on all Sam Coupe's providing they have Sam Rom 2.00 or above. In order to check the version of the roms you have fitted, type the following command.

PRINT PEEK (15)

If the number that is printed onto the screen is less than 20, then you have an old ROM, please contact Sam Computers for an upgrade ROM. The latest version of the ROM is version 3, peek(15) returns a value of 30.

2) The programs will work on either a 256k or a 512k machine.

3) I would advise people to purchase the 256k memory expansion card from Sam Computers to make full use of this program.

4) The programs will recognise the keyboard, joy-stick, Sam mouse or Blue Alpha input devices. I recommend you use the sam mouse option. The mouse option will only work if you have purchased the Sam Mouse from Samco. The Blue Alpha mouse, which is actually a joy-stick emulator, will work with SCADs as long as you do NOT select the mouse option.

5) The program will only work on a disc based system. Both SamDOS and MasterDOS may be used. There are limitations to note when you are using MasterDOS, please read the information below.

6) The programs should always be run from a 'clean' machine, this means that the computer should be turned off and then back on again and then the SCADs disc should be inserted into the drive and the computer booted. No other programs can be loaded into memory before running the Designer or the Supervisor programs. This eliminates all other programs without exception.

7) This package is NOT compatible with MASTERBASIC, or any other program that loads into memory when the computer is booted up. An error message will be displayed if any of the SCADs programs finds a 'alien' program in memory.

8) Due to the high piracy rate of modern day software, Glenco software have made the SCADs master disc, key protected. You may make as many copies of the SCADs master disc as you wish, however, whenever you run the Designer program, you will be prompted to 'INSERT KEY DISC' before the Designer will work correctly. The key disc is the original SCADS master disc. After inserting the master disc and pressing the SPACE BAR, the Designer will check to ensure the correct disc has been inserted. Once the disc has been verified, you may remove the disc and use a copy of the master disc to store you data.

Please keep your master disc safe. do not save any data directly onto the master disc.

# WORKING WITH MASTERDOS

1) You may only create 80 directory entries when formatting a disc.

2) You should not use subdirectories for graphics files.

3) Ramdiscs are not recognised.

4) Only drive 1 is recognised.

The 256k Sam Coupe has 49151 bytes of memory free for designing graphics and screens, and 41158 bytes of memory free for your BASIC programs.

The 512k Sam Coupe has 311295 bytes of memory free for designing your graphics and screens, and 41158 bytes of memory free for your BASIC programs.

You will notice that no matter what size of memory you have, you still have the same amount for your BASIC programs. Do not worry about this. It is extremely unusual to write a game that contains more than 40k of actual programming. The Supervisor program makes most of the decisions for you.

All arcade games use large amounts of memory in storing the graphical images, the drawings. I believe the ratio of programming memory (40k) and graphics memory ( 303k) to be a good balance for most games. If you own a 256k Coupe then please purchase the memory expansion, it will make a world of difference to your games.

The SCADs master disc is supplied with a copy of SamDOS 2. If you wish to use masterDOS, then first boot the computer with a disc containing a copy of masterDOS and then insert the SCADs disc to boot the SCADs package.

Please note, the masterDOS disc should not have a copy of MASTERBASIC on it. MASTERBASIC is not compatible with SCADs.

# GAMES DESIGNING THEORY

It is very important that you understand all of the concepts used by this games development system. If you do not understand any of the points made, please re-read the information again.

Before we go into the theory of designing your games, I would like you to consider a television cartoon. A cartoon is made up of a number of different characters ie Tom and Jerry, Bugs Bunny. Each of the characters within a cartoon is made up of hundreds of different images. If you could look at each of the images the cartoon was made from, you would see the character is drawn in a slightly different position in comparison to the previous image. When these images are displayed one after another in succession, the character appears to be moving and animated. This is just an optical illusion, the images are being swapped at such a rate that we cannot tell that the character is a series of images. A walking man would consist of a number of images of a man with his limbs in a slightly different position to the previous image. The first image would consist of the man standing upright, the next image would consist of the man standing with one leg stepping forward. The next image would have the man's leg even higher, and so on. When all of these different images are displayed one after another, the man appears to walk. This cartoon technique used to create animation is the same technique used to animate computer graphics. By drawing a number of computer images onto the screen one after another, we achieve the animation effect. This effect is very difficult to achieve when using BASIC as each image has to be drawn onto the screen 25 times every second and BASIC is simply not fast enough to achieve this effect.

We call every character you see on the screen, a SPRITE. In the Tom and Jerry cartoon, Tom is a sprite and Jerry is another sprite. The rake that Tom always seems to run into is also a sprite. Every object within a cartoon that moves or interacts is a sprite.

A SPRITE is the series of computer graphics that you see animated on the screen. A sprite can be a man walking, an alien life form, a spaceship or even a stationary object with no animation at all.

In a cartoon, as well as the main characters (Sprites), we also have the scenery. This is made up of even more drawings, drawings that do not move or change. The scenery graphics are simply there to enhance the appearance of the cartoon.

Games are very similar to cartoons, the same techniques that are used to make a cartoon, are used to create a game. Every object that plays an active part in a game is called a SPRITE. We also use scenery within a game to enhance the games appeal. Let us look at a typical game and see how it works.

Consider the game of BREAKOUT. In this game you control a bat at the bottom of the screen. Across the top of the screen are a number of rows of stationary blocks. You try to keep a bouncing ball in play by deflecting the ball off the bat. Every time the ball hits a block at the top of the screen, the ball bounces off the block, the block disappears and your score is increased. The game I have just described contains all of the ingredients for most arcade games. All of the graphics on the screen can be described as either sprite graphics or scenery graphics. The scenery graphics consist of all of the non participating graphics that you may see on the screen. The border of the playing area

will consist of scenery graphics, their only reason for being displayed is to highlight the edge of the playing area. If the floor of the Breakout game consisted of a pattern, the floor patterns would also be made out of scenery graphics. The scenery plays no part in the game. Scenery is there to brighten up the appearance of your game. The Sprites consist of all the objects that are taking an active part in the game. The sprite characters consist of the blocks at the top of the screen, the ball and the bat.

As mentioned above, the sprites are made up of a number of single frame images. The images on their own can not be moved around the screen or interact with the game in any way. We have to create a sprite character. A sprite character can be linked to a number of images, a number of sprites can be linked to the same image.As a sprite is moved around the screen, the image that is currently linked to that sprite is displayed.

You can give the sprite characters certain rules on how they will behave. As an example of some of the rules, the sprites can be instructed to disappear, bounce or stop when they hit another sprite or a piece of scenery. The sprites can be instructed to follow a predefined route. The sprites can fire missiles at each other, they can explode on command. They can also be under program control, joy-stick control or keyboard control.

You get full collision information. If a sprite collides with another sprite, your program is informed of the fact, and your program can then take the relevant action.

In the game of Breakout mentioned on the previous page, all of the blocks at the top of the screen, the bat and the ball are all separate sprites. They all have rules which determine how they will react to each other and the situations that may occur.

Here is a simple example of the rules that may have been set for each of the sprites used within a basic game of Breakout.

The block sprites will be stationary, with collision detection turned on, they will disappear when they collide with another sprite. ie the ball. The ball sprite will bounce whenever it collides with a piece of scenery or with another sprite. The bat sprite will be only be allowed to move horizontally, under control of the keyboard and it will stop if it collides with a piece of scenery.

AN **IMAGE** TAKES NO PART IN A GAME WHATSOEVER. AN IMAGE IS SIMPLY THERE TO BE COPIED **ONTO** THE SCREEN AS PART OF A SPRITE.

A sprite is basically an invisible square. This square can be programmed to act in different ways. The problem with a sprite is that it is INVISIBLE, the computer sees a sprite as a set of coordinates. It would be no good writing a game using invisible sprites, as nobody would be able to see them, this is why we have sprite images. In order to see the sprite moving around the screen, we attach an image to it. This image is printed onto the screen on top of the invisible square, the image will then reflect where the sprite is currently on the screen.

Consider the game of Breakout again. All the blocks at the top of the screen are the same shape, colour and size. There are 10 rows of blocks, each containing 8 blocks. We therefore have a total of 80 blocks at the top of the screen. Each of these 80 blocks is a SEPARATE sprite. We therefore have a total of 82 sprites (blocks, bat and ball) displayed on the screen at the start of a game. However, we only need to design 3

images, that of the block, the bat and the ball.Even though we have 80 block sprites at the top of the screen, they are all identical and so we can link all 80 sprites to 1 block image.

I hope this makes sense. There is no point in storing all 80 blocks in memory and linking each sprite to a different, but identical, image. All of the images are identical and so we can link all of the sprites to the same image.

I am trying to emphasize this point because it is very important for you to understand, any sprite may use any image at any time, the image is displayed purely as an image for the sprite. You can link 64 sprites to 1 image, it makes not the slightest bit of difference to the Supervisor.

## WHAT IS AN IMAGE ?

As you may already know, all computer graphics are created from pixels. A pixel is a spot of colour that can be displayed on the computer screen. The Sam Coupe, in screen mode 4, can display 256 pixels horizontally ( X dimension) and 192 pixels vertically ( Y dimension). A single pixel can be any of 16 colours from a palette of 128.

An image is a rectangular group of pixels. A block or image can be up to 32 pixels wide by 32 pixels high. You may select a pixel to be any of 16 different colours.

The images we have been describing on the previous pages are created by using the SCADs drawing Designer program.

The drawing designer within the Designer program will allow you to create all of your sprite images with ease. The Designer will allow you to manipulate your drawings, to spin them and to change their dimensions until you have created your desired image.

The Designer package will also allow you to add masking to your drawings. This will allow you to create windows within your sprite images (windows will allow the scenery graphics the sprite is travelling over to show through). You will also be able to define borders around your images to highlight the image. More information about image masking can be found in the *introducing masking* section of the manual.

The scenery graphics are created in the same way as you would define your sprite images. You have exactly the same drawing tools at your disposal to define scenery graphics, as you have to create sprite images.

# COORDINATE SYSTEM

The coordinate system used by the Sam Coupe is described in detail in the Sam Coupe Users Guide. It is very important that you understand how the Sam graphics system works before continuing with this manual.

When the Coupe is first turned on, the computer automatically defines two windows, the Graphics window and the Editing window.

The Editing window is placed at the very bottom of the screen, and uses 18 horizontal lines. All of the text you type at the keyboard appears in the editing window.

The Graphics window is defined to use the rest of the screen, from a point 13 lines from the bottom of the screen, to the very top of the screen, 174 lines in total. Try typing PLOT 0,0 a white dot will appear at the bottom left hand side of the screen, just above the editing area. The distance between the very bottom of the screen and the point you have just plotted is called a graphical offset. The top right hand side of the screen, using the offset system, is 255,173. There is a command within Sam BASIC that will allow you to alter the graphical offsets, the YOS command, setting the YOS variable to different values will alter the graphical offset starting position.

The coordinate system the SCADs package uses is very similar to the native Sam coordinate system. However, the SCADs package does not recognise the graphical offsets. The Supervisor treats the complete screen as a graphics window. This may cause a few problems. If you are going to use only SCADs graphics commands you will not notice any problems. However if you mix Sam BASIC graphics commands and SCADs commands you would find the coordinate systems do not match correctly, because SCADs command ignore the graphics offsets, whilst Sam BASIC commands do not. In order to overcome this problem, the Supervisor cancels any graphics offsets when the Supervisor is first initialised.

Once the graphical offset system have been cancelled, coordinate 0,0 is at the bottom left hand corner of the screen and coordinates 255,191 is at the top right hand corner of the screen. Try plotting these two points to verify the screen boundaries.

All of the sprites that are displayed on the screen have their own individual coordinates. To demonstrate how the sprite coordinate system works we will look at a typical SCADs BASIC command. It is not necessary for you to understand how this command works, it is only being used as a demonstration of the coordinate system. When we want to display a sprite on the screen we use the command.

```
SPUT 45,121,75
```

This command will print a sprite onto the screen. The first number (45) is the sprite number. The second number is the X coordinate (121). The third number is the Y coordinate (75). This command is informing the Supervisor program to place sprite 45 at coordinates 121,75. The sprite number is of no relevance in this example. The coordinate 121,75 is a position near the middle of the screen. The sprites are always referenced by their top left hand corners. In the above example the sprite will be displayed on the screen with its TOP LEFT hand corner at position 121,75.

# INTRODUCING SPRITE WINDOWS

A sprite window is a defined area of the screen in which the sprite can move, you may place a sprite outside of its own window but it will not be able to move.

There are actually two different types of sprite window that can be defined within the Designer program.

## MASTER WINDOW

The first window is called the master window. The edges of this window are defined within the Designer program and remain fixed. When you first load the Designer program, the master window is set to the full size of the screen. All of the rooms you create must have the scenery placed within the master window.

## SPRITE WINDOW

The second window is called the sprite window. The edges of this window are defined within the Supervisor program. When you first define a sprite, the sprite window is set to the same size as the master window. You may program each sprite that is to be placed onto the screen to have a different sized window. However, the sprites own individual window can not be larger than the master window.

You may place sprites onto the screen outside of their windows, however the sprites will not move or do anything at all, they will remain static. Trying to move a sprite that is displayed outside of it's own window will result in an error message being displayed.

## WHY TWO DIFFERENT WINDOWS ?

If you look at most computer games, you will see that a large majority of them have a static area on the screen to display items such as score, lives left, etc. This section of the screen never moves or alters in any way, except the score getting updated, etc. This section of the screen may also have a stationary picture displayed. This section of the screen is known as the panel. In most cases the panel is either running along the bottom of the screen, or it is running down the right hand side of the screen. Some games have a number of panels , and the playing area is displayed in the centre of the screen.

The master window should be defined as the area that is actually updated, the playing area. You may opt to have the playing area as the full size screen, that is entirely up to you. You should define your master window before you start to design your rooms. Remember the scenery graphics that make up a room can only be placed within the master window.

When you are using the Supervisor program and you issue a room command, only the section of the screen that has been defined as the master window is cleared. If you have loaded some graphics onto the screen to act as a 'panel', they will not be cleared.

On issuing the room commands, any sprites that are currently displayed on the screen will be removed. Any sprites that are placed outside of the master window will then be redisplayed once the room has been printed. This may be useful if you wish to graphically represent the number of lives you have left, etc.

Remember sprites placed outside of a window can not move.

The sprites may also have their own individual windows. The sprites own windows must be placed within the master window. The sprites windows could be used, for example, to allow of sprite to move backwards and forwards within a small section of the screen. Each sprite may have its own separate window.

If we have set a sprite to bounce around inside its window, the sprite will bounce whenever the forward facing edge of the sprite hits the window edge.

Example.

We have created an image that is 10 pixels by 10 pixels, the image has been linked to the sprite and the sprite is placed inside its sprite window which is set to the full size of the screen (256 x 192). If the sprite is placed in the middle of the screen and the direction is set for the sprite to move to the right, the sprite will bounce when it reaches ( x) coordinate 246.

## USING PANELS

If you are going to use a panel within your game, then you should first decide on the size the playing area is going to take. After deciding the size of the playing area, you will have an idea of the size of your panel. You can design your panel using any Sam art package.

The art package should be capable of printing the x and y coordinates of the pointers position. This will allow you to position items such as score text with ease.

When you are creating your panel graphics, you should bear in mind the start and finish coordinates of the panel. If you make the panel to large, then it may overlap into the playing area of your game. This is not a problem as the Supervisor will clear any section that overlaps into the playing area, but it may look untidy.

When you are creating a panel within your art package, it is important that you match the colours you are using within the Designer to the colours you are using within the art package. An easy may of matching the colours is to export a couple of drawings from the Designer program to a screen file. You may then import the drawings into the art package. You may then erase the drawings from the screen. The colours you imported into the art package should then match the colours within the Designer program.

The Designer program will allow you to store a panel in memory, so that whenever you wish to create a room, the panel is displayed. In order to use this feature, you must have at least 24576 bytes of memory free within the Designer program.

It should be noted that the panel must be loaded separately into the supervisor using the PANEL command.

# INTRODUCING ANIMATION

Taking the BREAKOUT game one stage further, the ball is the shape of a rugby ball, ie. oval shaped. As the ball bounces around the screen it rotates slowly. In order to make the ball appear to rotate as it moves, we should have defined a number of images each showing the oval shaped ball in a slightly different position. All we need to do now is alter the image number for the ball sprite as it moves, and the ball will appear to rotate. This is handled automatically by the Supervisor program.

There are certain rules that need to be remembered before you start to design your animation images.

1) All of the animated images must be the same X and Y dimensions.

2) All of the animated images for a given range must be in sequence.

3) There can be no gaps in image numbers for a given animation image range.

An animated sprite consists of a number of images that are displayed one after another as the sprite moves across the screen. Each of the individual images is created using the image option from within the Designer program.

A sprite can have up to 9 different animation ranges, dependant on which direction the sprite is travelling. It is possible to have a different animation range for all of the following directions.

| RANGE NO. | DIRECTIONS |
|---|---|
| 1 | UP |
| 2 | UP - RIGHT |
| 3 | RIGHT |
| 4 | DOWN - RIGHT |
| 5 | DOWN |
| 6 | DOWN - LEFT |
| 7 | LEFT |
| 8 | UP - LEFT |
| 9 | STATIONARY |

A completely defined set of nine animation ranges is called a SEQUENCE. The same animation sequence can be applied to a number of sprites.

When you have allocated an animation sequence to a sprite, all of the animation is handled automatically by the sprite Supervisor. The Supervisor monitors the direction the sprite is travelling and automatically adjusts the animation to suit the direction of travel.

To define an image range, you select the start image number and the finish image number. Both of the image numbers can be set to the same value, in this case the sprite will not be animated for that particular direction.

You do not need to set all of the direction ranges for a particular sequence, if a sprite changes its direction to a range that has not been defined, the previously defined range will continue to operate.

All of the animation ranges are defined using the drawing Designer program. You may create up to 64 different animation sequences. You can select any number of sprites to use a particular sequence.

Please see the table below for an example of a defined sequence,

| RANGE NO | IMAGE NO | DIRECTION |
|---|---|---|
| 1 | 0,1,2,3 | UP |
| 2 | 0,1,2,3 | UP - RIGHT |
| 3 | 4,5,6,7 | RIGHT |
| 4 | 8,9,10,11 | DOWN - RIGHT |
| 5 | 8,9,10,11 | DOWN |
| 6 | 8,9,10,11 | DOWN - LEFT |
| 7 | 12,13,14,15 | LEFT |
| 8 | Not Defined | UP - LEFT |
| 9 | 16 | STATIONARY |

If a sprite is placed onto the screen and starts to move to the right, the Supervisor will animate the sprite with images 4 - 5 - 6 - 7 - 4 - 5 etc. If the sprite then stops moving the sprite will change to image 16. Once the sprite starts to move up the screen the sprite will be animated with image numbers 0 - 1- 2 - 3 - 0 - 1 - 2 - 3 - 0 - etc. If the sprite then changed direction again to move diagonally UP - LEFT, this range is not defined and so the sprite would use the last range used which is 0 - 1 - 2 - 3 - 0 - etc.

This is all handled by the Supervisor, I have explained it so you can get your animations working with the minimum of effort.

Animating images is one of the most difficult effects to achieve. Trying to get a man to walk in a straight line without making him look like he has a limp is extremely frustrating. However patience should eventually pay off. Taking your time, and getting the animation correct within your programs is one of the biggest hurdles you will face in producing your mastergame. Take your time and get it right and it will look fantastic.

If you have no artistic talents, and I haven't, then do not despair, there are plenty of good graphics supplied with this package (I didn't do them, thanks Gary) which you can use quite freely. We will also be selling a number of graphics discs to complement this package. Please see the supplier of this program for more details.

Any graphics artists out there reading this, please send us your efforts for possible inclusion in one of our graphics discs. Who knows? You may even make some money out of it.

To find out how to animate your images please check the ANIMATION sections in the Designer section of the manual.

# MORE ABOUT SCENERY

We have so far only skimmed over scenery graphics. Scenery graphics are designed in a separate section of the Designer program, they have nothing at all to do with the sprite image graphics.

As I have mentioned in the previous pages, a game and its graphics can be broken down into 'useful' and 'non useful' graphics. Consider a game now, to consist of the sprites that move around the screen, and the game 'backdrops'. The backdrops consist of all the background graphics. Graphics which do not actually do anything except improve the presentation of the game, we call 'non useful'. The 'useful' graphics are the sprites. A game could not be played without the sprites. We have already looked at the sprites in great detail.

We are now going to look at the different types of graphics backdrops you may use within your programs. The graphics backdrops are made up of a number of drawings, similar to the drawings the sprites use. We call the drawings that go to make up the graphical backdrops 'scenery graphics'. You may design up to 255 different scenery drawings within the Designer program.

There are three distinct types of scenery graphics:- Foreground, Centreground and Background. Each of these types of scenery has a specific job to perform within your programs. The following examples will attempt to explain the role of each type of scenery.

Example a.

The first type of scenery we will describe is called BACKGROUND scenery. This will probably be the most popular type of scenery used within your games. Background scenery is scenery that has absolutely no use within your games, it is used purely as a cosmetic feature. This type of scenery can be considered as 'carpet'. The scenery is used for all of the sprites to travel over. The scenery will not restrict a sprite in any way, a sprite can never collide with background scenery. If the game of Breakout we have been mentioning throughout the manual had a patterned floor, the floor would be made out of background scenery.

Example b.

The next type of scenery you may use is called FOREGROUND scenery. This has the same collision characteristics as the background scenery. ie NONE. A sprite will never collide with a piece of foreground scenery. This type of scenery could be considered as the roof graphics. Every sprite moving around the screen will pass UNDER this type of scenery. As the sprite passes under this type of scenery the sprite will be hidden from view. You could build tunnels and bridges out of this type of scenery, the sprite will pass straight under them.

Example c.

Consider any computer platform game. The main character jumps around the screen landing on the various platforms that are scattered around the playing area. The platforms that the character lands on and jumps from serve no real purpose in the game, they don't do anything. They are there to allow the man to land on them and that's it. The man can never jump through a platform. This type of scenery will not allow any sprite to cross over it. This type of scenery is also used in maze games. The walls of the maze are created out of this type of scenery, no sprite can then cross over the sides of the maze. This type of scenery is called CENTREGROUND. This is a strange name and we had to think long and hard about. Centreground seems to aptly describe a type of graphical backdrop that is halfway between foreground and background.

The Designer program will allow you to piece together the three different types of scenery to create a complete 'backdrop', or ROOM. A group of scenery graphics placed onto a single screen is called a ROOM. The Designer program will allow you to create up to 255 different rooms for use within your games. A complete room can be printed to the screen in a fraction of a second, the sprites will then be able to move around the room under the control of your program. The sprites will not be able to cross over any centreground scenery. The sprites will pass over any background scenery and the sprites will always move under the foreground scenery.

The best way of understanding how the scenery works is to play the demonstration games that are supplied with this package. You should take note of how the sprites react to the various types of scenery that are placed onto the screen.

I advise you to load the drawing data into the designer program, select the ROOM option and experiment with the graphics. The drawing Designer program has an extensive range of commands that allow you to manipulate the scenery. As well as placing the three different types of scenery onto the screen you can move scenery, erase scenery, highlight the different types of scenery and view your handiwork.

In a number of games the playing area consists of a rectangle in the centre of the screen. Around the edge of the rectangle is a border. The Designer will allow you to load this border into memory whenever you edit the rooms. The border graphics can be designed using FLASH or any other art package. The border graphics usually contain such information as your score, hi score and the number of lives left. This 'border' is described in one of the previous sections under the heading *'Introducing sprite windows'.*

A room can consist of up to 255 background drawings, 255 centreground drawings and 32 foreground drawings. A room can also consist of up to 256 path points or NODES. We will describe the actions of Nodes a little later in the manual.

You will note that you can only use 32 foreground drawings as opposed to 255 background/centreground drawings, there is a simple reason for this, SPEED.

Foreground drawings have to be printed onto the screen everytime a sprite is moved. The more drawings that have to be printed, the longer it will take to update a single screen. The screens can not be displayed until they have been completely updated and so the more items to update on the screen the slower the graphics will become.

# INTRODUCING MASKING

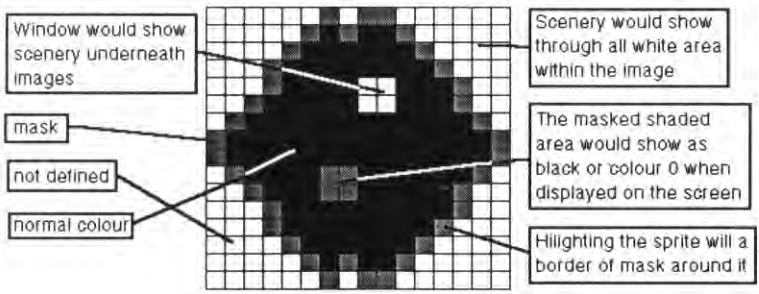
You have to understand how sprites/images are displayed on the screen in order to make full use of masking. When an image moves over a piece of scenery, the sprite remembers any scenery it passes over and then replaces the scenery once it has passed. If the sprite did not replace the scenery, the scenery would soon become obliterated. A sprite image is placed onto the screen by firstly saving the scenery underneath the sprite (as detailed above) and then overwriting the colours that were on the screen with the sprite image colours. However if we literally copied all of the sprite colours onto the screen we would have a major problem.

All images are created within a box. There would not be a problem if the sprite image was completely square and filled the box, but in reality only a number of sprites are actually square. In most cases there are a number of blank areas around the edge of the image, but still inside the box. Even though you have not defined these colours around the image, they are still stored as part of the image ( the images are stored as boxes within memory). If you simply overwrote the screen colours with the sprite image colours, then the area around the sprite that had not been defined would be printed to the screen. This means that all of the sprites displayed on the screen would have a black box around them, this is clearly unacceptable.

The solution to the problem, is to only print to the screen the pixels that make up an image that are NOT set to colour 0. This is a very simple solution. However it does cause one more problem. This technique will not allow you to print colour 0 anywhere within your image. Anywhere that contained colour 0 would not be written to the screen, and so whatever is originally on the screen at that point (the scenery) will show through the sprite image.

**ANY PART OF A SPRITE IMAGE THAT IS SET TO COLOUR 0 WILL ACT LIKE A WINDOW WITHIN THE IMAGE, WHATEVER IS UNDERNEATH THE SPRITE IMAGE WILL SHOW THROUGH WHEN THE SPRITE IS DISPLAYED ON THE SCREEN.**

To get around this second problem is also simple, we will create an extra logical colour. A logical colour that does not actually exist. This colour is called a mask colour, it acts as though it is a colour, set to the same colour as the background ( colour 0 ), but it will not allow the graphics underneath to show through the sprite image.




Using masking will allow you to achieve some special effects once you have

understood the principles.

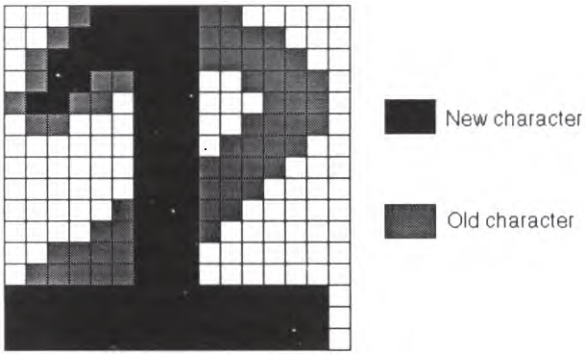
You can of course not use the mask at all and let the screen colours shine through the image, this looks like the sprite has a window.

You may completely surround the sprite image with a layer of mask 1 pixel wide, this effect can highlight a sprite when it is displayed on the screen and make it stand out.

You can use the mask within your drawing to achieve a darkened window effect.

Experimenting with the mask can produce some interesting results. The masking effect can be used when you create a sprite image, foreground scenery and when defining your character sets.

It is important to note that character sets do not store the backgrounds when they are printed onto the screen. This may cause some problems when you are overwriting existing text with even more text. As the backgrounds are not stored, printing a new character on top of an existing character will allow the old character to show through.




The diagram above shows the problem that may occur when you overwrite a character that is already placed on the screen. The old 2 has been overwritten by a 1, however both of the characters will be displayed, one on top of the other. To get over this problem, we need to use masking. All of the characters that will overwrite other characters should have all there blank spaces filled with a mask, by doing this the old character will be completely overwritten.

Please note that if you export drawings, the mask information can not be exported with them. This is because a mask is a logical colour, and logical colours can not be displayed outside of the Designer and Supervisor programs.

Similarly, if you import drawings, you can not import mask information. However once the drawings have been imported, you may edit them and add the mask information.

# INTRODUCING SPRITE PLANES

In most games it would be advantageous if we could allow some sprites to pass over other sprites without a collision being detected.

The system the SCADs Supervisor uses is called collision planes. When each sprite is placed onto the screen, it is placed on one of eight collision planes. Imagine eight separate screens stacked one on top of another, the sprite is then placed on one of these eight screens.

You can not tell by looking at the sprites moving around the screen which of the eight planes the sprite is travelling, however the sprites will only detect collisions with other sprites that are travelling in the same plane. This system will allow a number of sprites to be moving around the screen all in different collision planes. Even though sprites are passing over each other, only the sprites within the same collision plane will be reported as having collided. This is clearly a very useful collision system, however there is a disadvantage in that all of the sprites that you wish to have collision detection reporting have to travel in the same plane. This is not true. We have also introduced multi-plane collision reporting.

Multi-plane collision reporting works like this.

When each sprite is placed onto the screen, it is placed in one of the eight collision planes. You may then instruct the sprite to detect collisions in a number of planes. I think an example is called for. We shall number the collision planes 1 - 8.

We place a sprite on the screen and instruct it to travel in collision plane 5. We can then instruct the sprite to detect collisions in collision planes 1 and 4. The sprite will only report collisions when the sprite overlaps a sprite that is travelling in collision plane 1 or collision plane 4. It should be noted that in this example the sprite will not detect a collision with another sprite travelling in the same plane as itself (plane 5).

When a sprite is initially defined, the Supervisor sets the sprite to detect collisions within all sprite planes. To alter the collision detection characteristics of the sprite, you must use the Supervisors CPLANE command.

We have stated earlier in the manual that a sprite can not pass over centreground scenery, this was not strictly true. Since we introduced collision planes, we thought it may be nice if you were given the option of allowing the sprites to pass over centreground scenery. All centreground scenery is placed in collision plane 8, if you turn off the collision detection for collision plane 8, then the sprite will pass over centreground scenery as if it was not there.

It should be noted that it does not make any difference in which collision plane the sprite is placed to detect collisions, the planes the sprites detect collisions in is set by the CPLANE command within the Supervisor, as stated above when the sprite is first defined, the Supervisor sets the sprite to detect collisions on all planes. You may alter the collision detection plane status after the sprite has been defined.

As you are aware there are eight separate collision planes for the sprites to travel in, in the examples above the sprite planes are numbered from 1 to 8, this was used purely as a way of simply describing the examples. In reality the sprite planes are numbered slightly differently. The following table shows how the planes are numbered.

| Plane number | Sprite plane |
|---|---|
| 1 | 1st sprite plane |
| 2 | 2nd sprite plane |
| 4 | 3rd sprite plane |
| 8 | 4th sprite plane |
| 16 | 5th sprite plane |
| 32 | 6th sprite plane |
| 64 | 7th sprite plane : Nodes |
| 128 | 8th sprite plane : Centreground scenery |

The sprite planes are actually numbered from 1 to 128. The reason for this is simple, it is far easier to select multiple sprite planes using this numbering system. Remember a sprite can only be placed in a SINGLE sprite plane. However the sprite can detect collisions within multiple sprite planes.

To select a number of sprite planes, we simply select the planes in which we wish to detect collisions from the right hand column and then we total the numbers from the left hand column.

Example.

To detect collisions within planes 1,4,5,8 we would add the following values together 1+8+16+128 which gives the total of 153. We would use this value in the CPLANE command to inform the sprite of which collision planes in would detect collisions.

If you do not include plane 128 in your calculations, then the sprite will pass straight over the top of centreground scenery.

You will notice that plane 64 is used by the nodes. If you wish a sprite to detect nodes, then the sprite must have its collision status set to detect collisions within plane 64. Nodes are described in the next section of the manual under the heading, *introducing nodes.*

If you are defining a sprite to fire missiles, then it may be a good idea to set the missiles so that they do not detect collisions in the plane that the sprite firing the missiles is placed. In some circumstances the sprite firing the missile may be blown up by the missile it is firing. By altering the plane detection on the missile this can be avoided.

It should be noted that all sprites will detect collisions with doors, no matter what collision plane detection has been set. We have not looked at door sprites yet, but I have included the information here for reference.

# INTRODUCING NODES

If you watch some arcade games you will notice that some of the sprites seem to follow a specific path around the screen doing various actions, all automatically. You can program the sprites within your games to follow distinct paths. A Node path is an invisible path that only the sprites can see.

To define this sprite path we need to set up a number of node points. A sprite can only change its direction when it hits a node point. You can program the directions that the sprites can leave this point. You need three pieces of information for every node point you define. You will need to program the node point's position on the screen (X and Y) and you will need to program the directions the sprites can leave that node.

As a very simple example we will explain how a sprite can be programmed to move backwards and forwards between two nodes.

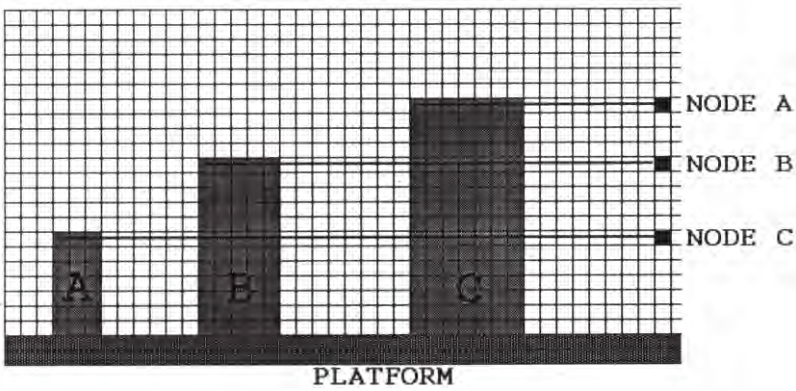
We will place the first node at position 40,100, the second node we will place at 180,100. Both of these node points are on the same horizontal line (100), The first node is placed towards the left hand side of the screen and the second node is placed towards the right hand side of the screen. If we set the allowable leaving directions for the first node to be right only and we set the allowable leaving directions for the second node to be left only, we have set up a node path. If we place a sprite at location 40,100 and set it moving, it will move to the right until it hits screen position 180, 100. The sprite will then change direction travelling back toward the first node again. The second node will only allow the sprite to leave in one direction and so the sprite has no choice. We now add another node to the screen at position 180,10 and we set the allowable leaving direction to up. We then change the allowable leaving direction of the second node to left or down. We have created a choice for the sprite. If we program the sprite to move randomly along the node path the sprite will move from the first node heading towards the right of the screen. When the sprite reaches screen position 180,100 it has a choice. The sprite can either move down the screen or the sprite can move to the left again. This is a very basic example of how a node path works. In practice a room is made up of a large number of nodes and complex paths are set up for the sprites to follow.

It is important for you to remember that the sprites can only change direction when they hit a node. If the sprite was travelling between the first node and the second node it could not suddenly decide to change direction to move upwards. There is one occasion when a sprite could change direction when it was not placed over a node, and that is when the sprite has collided with either another sprite, providing the bounce was turned on, or a piece of centreground scenery. If the sprite hits another object the sprite will simply go into reverse and head back to the node it had just left.

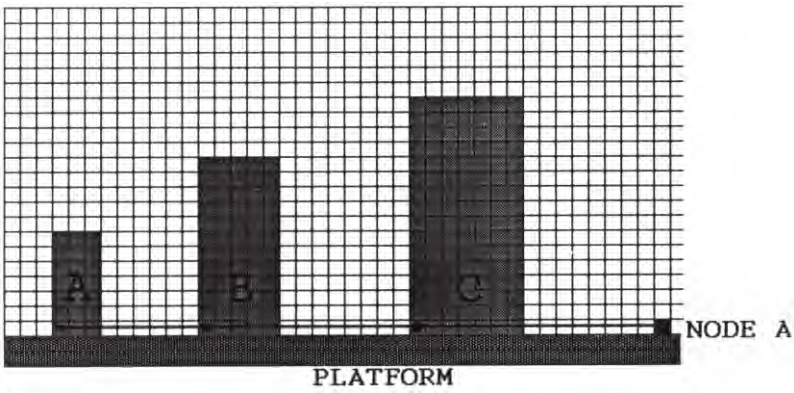
The sprites will follow the node paths only if you have instructed them to. You may have a number of sprites on the screen that are following the node paths and you may also have a number of sprites on the screen that are totally ignoring the paths.

If we make the sprite hit a node when the top left hand corner of the sprite has the same coordinates as the node point, we have a problem. If you are writing a platform game, most of the sprites will be travelling along the platforms. We have a number of

different sprites detecting nodes. The sprites are all different heights. If we wanted ALL of the sprites to perform a certain action at a particular point on the screen, then we would have to place a number of nodes into the room, all of them at the same x coordinate, but at slightly different heights. This is because the sprites x and y coordinates have to match the nodes x and y coordinates, the sprites are different heights, and so we have to have a number of nodes to match all of the different heights. Remember, in a platform game the sprite will be travelling along the platform, and so there y coordinates will all be different for the same x coordinate. I think an example will explain this a little better.




In the picture displayed above are three sprites, all travelling along the same platform. We wish the sprites to reverse direction when they reach a certain point on the platform. The three sprites are three different heights, sprite A is 7 pixels high, sprite B is 12 pixels high and sprite C is 16 pixels high. In order for all of the above sprites to change direction, we would have to use three separate nodes, node A,B and C. This is clearly a problem. In order to simplify matters, the Supervisor detects node collisions when the sprites BOTTOM LEFT corner has the same coordinates as the node.




As we can see from the above diagram, making the sprites detect nodes when their

BOTTOM LEFT corners have the same coordinates as the nodes will allow you to only place a single node instead of three. Another advantage in using this system is that it is far easier to place nodes within the Designer program. Instead of having to calculate the position of the platform, and then adding the sprite height to calculate the node position, all you have to do is place the node one pixel above the platform. All of the sprites travelling along the platform will then hit the node.

I have stated that the sprites must hit a node before the sprite can change direction. I define hit as meaning that the sprites BOTTOM LEFT corner and the node must have the same screen positions to the exact pixel. We have placed a sprite (12 pixels high) onto the screen at position 97,111 (the sprites bottom left hand corner has the coordinates 97,100) and we have placed a node at position 120,100. The sprite is programmed to move to the right at a speed of 2 pixels every move. This sprite will never hit the node. The sprite will be in position 119,111 and then 121,111, totally missing the node at 120,100. You can use this system to your advantage, you can allow certain sprites to hit a specific node, whilst other sprites moving along the same axis can be made to miss the node.

You may program the nodes to do far more than allow the sprites to follow a path. You may program the nodes so that the sprites fire missiles, jump, remove the sprite from the screen and remove the sprite and place it in another position on the screen. The Designer program will allow you to select all of these features with ease.

Nodes are normally used to determine the directions a sprite can travel when the sprite hits a node. If you ignore all of the other features of the nodes for a minute and just concentrate on the directions a sprite can take it will be so much easier for you to understand.

When a sprites bottom left hand corner match those of a node, the sprite is said to have hit the node. The sprite will then look at the node and see what directions are available for the sprite to leave the node. If there is only 1 direction, then the sprite has no choice, and the sprite will leave the node in that direction. If there is more than 1 direction the sprite has a choice. Firstly the sprite will discount the direction it has just come from (if there are two directions then the sprite will never simply reverse direction). If there are still more than two directions after eliminating the entry direction, the sprite will make a random choice and then move in the desired direction is has chosen.

The situation becomes more cloudy when we introduce other commands, such as jump, fire and remove. These commands are allocated to a sprite direction, even though the sprite will never actually move in that direction. If, for example you specified the up direction to fire a missile and the sprite decides it will go up, the sprite will not move up, but it will fire a missile. The sprite will not move, the next time the sprite is called to move by the Supervisor, the sprite is still hitting the node (as it has not moved) it will then go onto pick a different direction (but not up, as this has been temporarily discounted because the sprite thinks it was already moving up). I told you it was a bit complicated.

If a sprite hit a node with only 1 command defined within the node, then that command will be executed. If there are two commands defined within the node, the sprite will temporarily ignore the command for the direction it has just come from and so it will execute the other command. If there are more than 2 commands defined within the node, the sprite will forget the direction it came from and then choose randomly from

the commands that are left.

I think we had better look at some examples in order to make using nodes a little bit clearer.

Example. A node has direction (0-up) defined to speed. The sprite hits the node moving to the left. There is only 1 option for the sprite so therefore it will now move up the screen.

Example. The same sprite hits the same node, again moving from the left, this time direction (0 -up) is defined to fire a missile. When the sprite hits the node, again it only has 1 choice (direction 0) so the sprite fires a missile but does not move. The next time the sprite comes to move (it is still on the node) it will start to move again to the left (it has eliminated option 0 and therefore has no options left and so reverts back to its old speed)

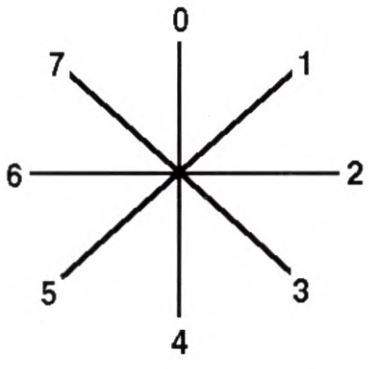
Example. The node now has direction (1-up right) defined as fire, and direction (2-right) defined as speed. When the sprite hits the node moving towards the right, the sprite has two choices.

If the sprite chooses direction 2 then the sprite will keep moving to the right. If the sprite chooses direction 1, the sprite will fire a missile (this direction will then be eliminated) and then it will move to the right (this is the only option left).

You may program all or any of the non joy-stick/keyboard controlled sprites to follow nodes. Each sprite can have node detection turned on or off, this is selectable with a simple Supervisor command. It should be noted that any sprite that is under keyboard or joy-stick control will ignore all but two nodes commands. The joy-stick/keyboard sprites will only act on speed and remove/replace commands. The joy-stick/keyboard sprites will ignore the automatically jump, fire missile and removed commands.

A sprite can not 'hit' a node if the sprite is currently jumping or dropping.

In order for you to make the best use of nodes, I will try and explain exactly how the Supervisor will deal with a sprite when it comes into collision with a node.




Firstly it should be noted, the action a sprite takes when it collides with a node depends on the type of sprite that is in collision. There are basically two different types of sprites. Normal sprites and joy-stick / keyboard controlled sprites. We shall look at the two types of sprites separately.

The diagram on the left is displaying all of the different directions a sprite can travel in.

In the following descriptions, we shall refer to this diagram.

Firstly we shall look at the actions a normal sprite will take when it collides with a node.

## NORMAL SPRITES

We will assume that the normal sprite (not controlled by the joy-stick or keyboard) was travelling on the screen from left to right in a straight line (direction 2) when the sprite hit a node.

The first action the Supervisor will take is to make a list of all of the directions that have been programmed within a node. It should be noted that the directions (0 - 7 ) may contain more than just speeds. A node can contain instructions to fire missiles etc.

The Supervisor will then establish which direction the sprite was travelling. In the above example the sprite was travelling in direction 2. The Supervisor will then temporarily cancel the opposite direction in which the sprite was travelling. In this example the Supervisor temporarily cancels any instructions that may have been programmed into direction 6. The Supervisor then checks all of the other directions to see which directions have instructions stored in them. If there are directions, then the Supervisor will randomly select a direction and execute the instruction that was stored within that direction. Please see the following list of possible instructions and the actions the Supervisor will take.

**SPEED:The** Supervisor will alter the speed of the sprite to the speeds that were stored within the node. The sprite will then start to move at the new speeds, thus leaving the node.

**REMOVE:The** Supervisor will remove the sprite from the screen. No further actions will take place for that particular sprite.

**REPLACE:The** Supervisor will remove the sprite from the screen, and replace the sprite at the point specified within the node. The sprite will then start to move from the new position on the screen.

**FIRE:The** Supervisor will instruct the sprite to fire a missile. If should be noted that the image that is currently being displayed for the particular sprite should have a missile definition. If the image does not have a missile definition the sprite will not fire a missile. After the sprite has fired a missile, the sprite will still be placed on the node, and the Supervisor will then select another direction from the directions that are left, and execute the instruction for the new direction. The Supervisor will not select the same direction twice. Once a missile has been fired for a particular direction, that direction is then cancelled.

**JUMP:The Supervisor** will instruct the sprite to jump. The .sprite will then leave the node, following the speeds defined within the jump sequence that has been specified.

We stated at the start of the text that the Supervisor will temporarily forget the opposite direction to the current direction of travel. If, after doing this, there are no directions left that contain instructions, the Supervisor will then look at the opposite direction and execute any instruction that may be stored. The opposite direction is always executed as a last resort.

If there are no instructions left for the Supervisor to execute, the sprite will then continue to move past the node in the direction it entered the node.

## JOY-STICK / KEYBOARD CONTROLLED SPRITES

Joy-stick / keyboard controlled sprites do not act in the same way as normal sprites. When a joy-stick / keyboard controlled sprites collides with a node, the Supervisor creates a list of all of the directions that have instructions defined within the node. The Supervisor will only look for speed and replace instructions, all other instructions within a node are ignored. If, after the Supervisor has created the list, there are no instructions, the sprite will ignore the node and continue to travel past the node.

Once the list has been created, the Supervisor will test the joy-stick or keyboard to see in which direction you wish the sprite to travel. Once you have selected a direction, the Supervisor will look at the instruction stored within that direction and execute that instruction. If the direction selected with the joy-stick/keyboard has not been defined within the node, the Supervisor will look at the directions on either side of the joy-stick/keyboard selected direction to see if one or both of the directions on each side of the joy-stick/keyboard selected direction have been defined. As an example the node had directions 0,2 and 4 defined, and you have selected direction 3 with the joy-stick/keyboard, the sprite can not act on the instruction within direction 3 as no instruction has been defined. The Supervisor will then look at the directions on each side of direction 3. ie directions 2 and 4. If both of the alternative directions have been defined, then the sprite will stop and wait for another direction from the joy-stick or keyboard. If there is only one direction defined, out of the directions on either side of the joy-stick/keyboard direction, the Supervisor will execute the command stored within that direction. This may seem a little complex, but please read on, there are a number of examples a little later in the manual.

The Supervisor will only act on 2 node instructions.

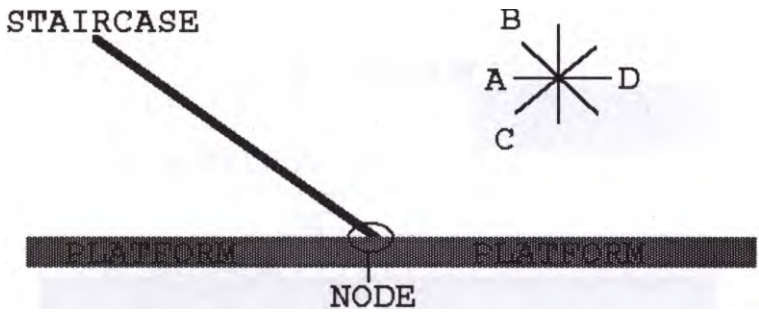
**SPEED:The sprite** will take the speeds that are stored within the node and the sprite will start to travel at the new speed in the specified direction. Once the sprite has left the node, the allowable travel directions for the sprite will be altered so that the sprite can only travel in two directions. The allowable directions will be set to the current direction of travel, and the opposite direction. As an example, the sprite starts to move in direction 3 after leaving the node, the allowable travel directions will be altered so that the sprite can only travel in direction 3 and direction 7. The sprite will only be allowed to travel in these two directions until the sprite hits another node.

**REPLACE:The** sprite will be removed from the screen and replaced at the new point on the screen. The new point on the screen must contain another node, informing the sprite of the directions it can leave the new position.

When a sprite is under node control, the sprite drop is calculated in a different manner. Under normal conditions (not under node control) the sprite will drop whenever there is nothing under the sprites feet. This system is fine under normal circumstances, however, if we wish to program ladders or stairs then this system is obviously not acceptable. When a sprite is under node control, the sprite will only drop if it is moving

horizontally. A sprite will NOT drop if, the sprite is currently hitting a node, or the sprite is moving up or down, or the sprite is moving diagonally. This will allow the sprites to travel up and down ladders, or up and down stairs, without the sprite dropping.

I think an example of programming stairs is needed.




The staircase in the above diagram is created using Background scenery. If the staircase was created out of Centreground scenery, the sprites would simply bounce of the stairs. At the bottom of the stairs is a node. The node has been defined to allow the sprite to move in two directions, B and D. A sprite is moving along the platform from right to left. The joy-stick is being held to the left (not diagonally). The sprite moves to the left until the sprite hits the node. Once the sprite has hit the node, the Supervisor creates a list of all of the directions that the sprite may leave the node, in this case directions B and D. The joy-stick is still being held to the left. The Supervisor will check the direction of the joy-stick. The node will not allow the sprite to travel to the left, and so the Supervisor will check the directions on either side of the joy-stick direction. In this case, there is only one direction, direction B, and so the sprite will start to move up and left. The sprite is moving diagonally and so the sprite does not drop.

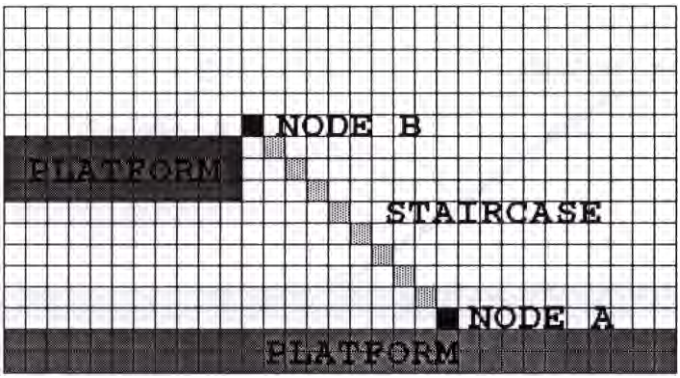
Once the sprite has left the node, the allowable travel directions will change so the sprite will only be allowed to travel in two directions.ie. Up/Left (direction 7) or Down/Right (direction 3).

The sprite will continue to travel in one of the two allowable directions until the sprite hits another node. The process would then be repeated.

Lets take the example one stage further. The node at the bottom of the staircase now has three directions defined, directions B, A and D. The sprite is travelling to the left. The joy-stick is being held to the left. The sprite hits the node and the Supervisor again creates a list of directions that the sprite may leave the node, in this case directions B, A and D. The joy-stick is still being held to the left. The Supervisor will check the direction of the joy-stick. The node will allow the sprite to exit to the left, and so the sprite travels straight past the staircase. The only way the sprite could travel up the staircase, in this particular case, is if the joy-stick direction was changed to the upwards direction whilst the sprite was hitting the node.

It should be noted that there should be another node at the top of the staircase that will allow the sprite to travel horizontally again. You should exercise extreme care when

placing this sprite, remember, as soon as the sprite starts to move horizontally again, the drop detection will be turned back on again. If the sprite is not standing over a piece of platform when the sprite starts to travel left/right, the sprite will drop.




The diagram shows how the nodes should be placed when you are defining a staircase. Node A at the bottom of the stairs has two directions defined, directions 7 (Up-left) and 2 (Right) . Node B at the top of the stairs also has two directions defined, directions 6 ( Left) and 3 (Down-right). Node B is placed at a position 1 pixel to the right of the platform and 1 pixel above the platform. A sprite is travelling from right to left along the bottom platform. Throughout this example the joy-stick will be held to the left (not diagonally).

The sprite travels along the platform until it hits node A, at this point the Supervisor creates a list of directions that the sprite may leave the node. The joy-stick is held to the left, but the sprite may not leave the node travelling left. The Supervisor will then check the alternative directions ( 5 and 7 ). There is only one alternative direction available ( direction 7) and so the sprite will start to travel in that direction (Up-left). The sprite then leaves the node, the allowable travel directions are altered so the sprite may only travel in directions 7, and the reverse direction, direction 3.

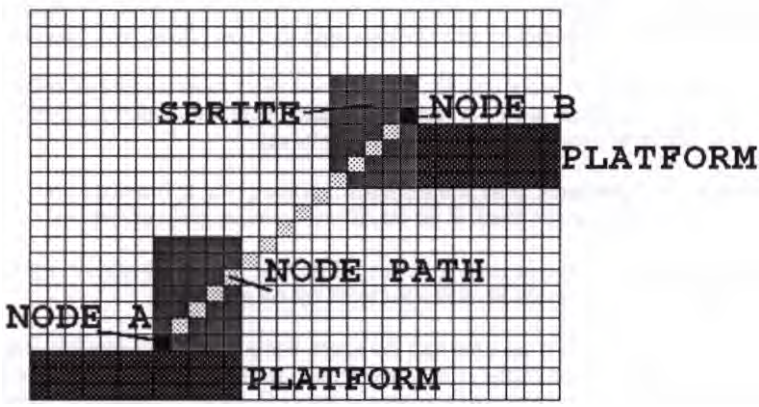
If the joy-stick is continuously held to the left, the sprite will continue travelling up the staircase. The sprite will not drop because the sprite is travelling diagonally. Eventually the sprite will hit the node at the top of the staircase, node B. The Supervisor creates a list of directions for the sprite to leave the node. In the case of node B, the directions are 6 and 3. The joy-stick is being held in direction 6 (left) and so the sprite leaves the node, travelling to the left. As the sprite leaves the node, the allowable sprite directions change so that the sprite will only be able to travel in directions 6, and the reverse direction, direction 2.

The sprite is now travelling horizontally (left) and so the sprite is tested to see if the sprite will drop. This is where the correct positioning of the node is vital. The sprite drop test is only executed after the sprite has left the node. In this case, the sprite will not drop as the sprite is already over the platform. If the node had been further away from the platform, the sprite would have dropped, as the sprite would not have been over the platform. The sprite will now travel along the top platform.

We shall now see what actions the sprite will take if you now reverse the sprite direction so that the sprite is travelling back towards the node.

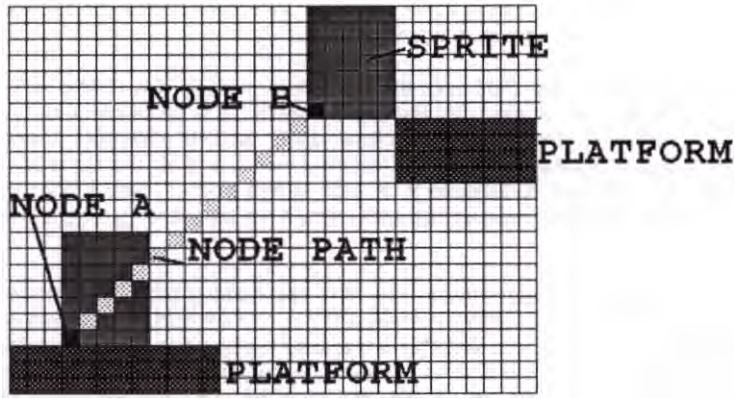
The sprite will travel to the right until the sprite hits node B. On hitting node B, the Supervisor creates a list of allowable leaving directions. The sprite can not travel right anymore (direction 2) and so the directions either side of direction 2 are tested ( directions 1 and 3). There is only one direction defined, and so the sprite will start to travel down the staircase in direction 3. If node B had been 1 pixel further right, the sprite would have dropped because the sprite would not be standing on the platform or a node.

Now, if you thought that was difficult, you ain't see nothing yet. The stairs we have described so far have been simple, it is very easy to make a sprite travel up a staircase when it is travelling from right to left. This is because the sprite is detected as hitting the node at the sprites left hand side. This is called leading edge detection. As soon as the sprite hits the node, the sprite can start to move up the staircase. However, when the sprite is travelling from left to right, the sprite will have to move near enough completely past the node before the sprite hits the node. This is called trailing edge detection. There is a demonstration on the disc called NODE1 that shows the problems that can occur.




The diagram above demonstrates the problem of defining staircases that will allow sprites to travel from left to right. In this example to sprite was travelling from left to right along the bottom platform. The sprite hits the node, and starts to travel diagonally along the node path. As the sprite reaches the top position show in the diagram, the sprite collides with the platform and so it can not travel any further.

If the sprite had been travelling along the top of the platform from right to left, the same problem would occur. The sprite would hit node B and then try to move diagonally down the node path. As soon as the sprite started to travel down the node path, the sprite would collide with the platform and stop. In order to overcome this problem, the nodes have to be placed further away from the platform. The following diagram demonstrates how to place the sprite so that they can travel correctly up and down the staircase.




In this diagram the nodes have been placed a small distance away from the platform. This will allow the nodes to travel up and down the staircase with ease. The positioning of the nodes in this instance is critical. If you place the nodes too close to the platform, the sprites will collide with the platform, and so they will not be able to complete their journey along the node path. If you place the nodes too far away from the platform, the sprites will reach the node, but as soon as they start to travel horizontally they will drop.

In order to calculate the exact position of the node, you must know the width of the sprite. The node should be placed a sprite width away from the platform. Please refer to the diagram above to understand what I am trying to say.

This technique will only work, if all of the sprites that are going to travel up and down the staircase are the same width, there is no way of overcoming this problem.

It should now be apparent that placing nodes within your rooms will require a great deal of careful planning. The results of all of your work will be worth it.

All of the above descriptions assume that the sprite has a speed of 1 pixel. In practice 1 pixel/move is a little bit slow. Most or all of the sprites will be travelling at a faster speed. The most difficult part of programming sprites is trying to get the correct sprite to always hit the correct node. If the sprites are moving at speeds of greater than 1 pixel/move every care will need to be taken. Please refer to the following tips.

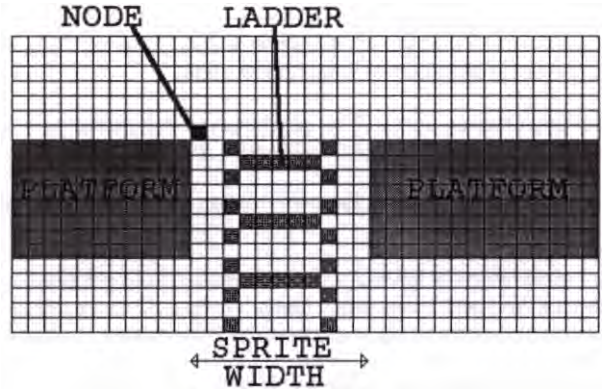
1) If the sprites are to travel at a speed of 2 pixels/move (most popular sprite speed) then ensure that all of the nodes are positioned on either odd or even coordinates ( X dimension only) and the sprites are also positioned in the same odd/even coordinate position. Please refer to the example at the start of the nodes section of the manual. All of the sprite movement commands (jump) must then contain only even movement speeds. ie. All of the movements defined within the sprites jump sequence must be either 0,2,4,6 or 8 pixels/move. If you have any odd pixels/move speeds, the sprite will not hit all of the nodes you require.

2) Ensure any nodes placed on platforms are 1 pixel above the platform.

3) If you are creating stairs using the room designer, it is a good idea if you place the node at the top of the stairs first. Remember the discussion on sprite travelling from left to right, and place the nodes at exactly the right pixel. After you have placed the node at the top of the stairs, go to the VIEW/SPEEDS option. The path the sprite will take is displayed as a series of dots. The bottom of the staircase should have a dot 1 pixel above the platform. Move the arrow over this pixel and note down the arrows coordinates. If you then go to the PLACE option and move the arrow so that the displayed coordinates match the coordinates that you have noted down and place the node at that point.

4) Carefully plan every room. It is a good idea if you use graph paper to plan the room before you transfer your ideas onto the computer. Remember some of the nodes have to be placed a small, but exact, distance away from the platform. With careful planning this should become a fairly simple process.

We have so far only looked at defining staircases using nodes, we shall now look at defining ladders.




The diagram above shows a typical ladder set up. The node at the top of the ladder is defined so that the sprite can leave in three directions, left, right and down. The ladder graphics themselves are created using background scenery. If the ladder was created using centreground scenery the sprites would not be able to travel up and down the ladder, they would bounce off.

The node has to be placed in the position shown, one pixel line above the platform, and one pixel to the right of the left platform. One very important point to note about this particular set up is the gap between the two platforms. The gap must be the same width as the sprite itself. Lets discuss why.

The sprite is travelling up the ladder. When the sprite hits the node, the Supervisor creates a list of all of the possible leaving directions. In this case the directions are left ( 6), right (2) and down (4). The sprite can not travel upwards anymore. The Supervisor will then wait for a direction from the joy-stick. Lets look at the possible directions the sprite can take.

DOWN:Selecting the down direction, the sprite would simply move back down the ladder, the allowable movement directions would be changed so that, once the sprite has left the node, the sprite could then only travel up (0) and down (4). The directions would be set until the sprite hits another node (placed at the top and bottom of the ladder).

LEFT:Selecting the left direction, the sprite would leave the node, travelling to the left. The sprite is now travelling horizontally and so the sprite is tested to see if it will drop. As the sprite has already left the node, the sprite will be at least one pixel over the left hand side platform, and so the sprite will not drop. The sprites allowable travel directions will be altered so that the sprite can now only travel left (6) and right (2). The directions would be set until the sprite hits another node.

RIGHT:Selecting the right direction, the sprite would leave the node travelling to the right. The sprite is now travelling horizontally and so the sprite is tested to see if it will drop. This is where the importance of making the gap between the platforms the same width as the sprite is vital. The sprite has left the node, the sprite will be at least one pixel over the right hand side platform, and so the sprite will not drop. If the gap between the platforms had been any larger, then the sprite might not have been over the platform, and the sprite may have started to drop. The sprites allowable travel directions will be altered so that the sprite can now only travel left (6) and right (2). The directions would be set until the sprite hits another node.

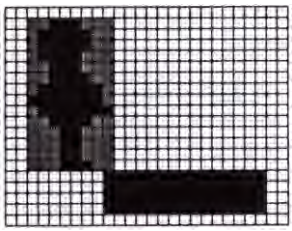
I hope this description makes sense. The gap between the platforms is very important. A gap which is larger than the sprite width will cause the sprite to drop.

As with the stairs examples, careful planning should overcome any problems that may occur.

There is another factor that we have not yet considered, I did not want to introduce this idea at the start as you will have been totally confused, as opposed to only semi-confused. The hardest part of nodes to grasp is the Supervisors FEET command.

The FEET command will allow you to effectively reduce the collision width of the sprite. This command is discussed in detail a little later in the manual, but I will describe the basics of the command here so that you can see the relevance in designing your node paths.

It was found, while writing the early SCAD demonstrations, that the sprites could appear to be walking on air, if the sprite was standing on the very edge of the platform, possibly by one pixel.

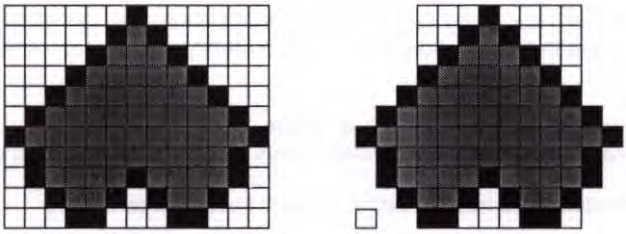


This can appear to be impossible, the main character appears to be walking on air. The sprite has not dropped because the sprite box is still over the platform. It was decided to introduce a command that would reduce the effective width of the sprite, thus the FEET command was created.

The command will allow you to reduce the collision width of the sprite. The command format is as follows

FEET *sprite number, left foot, right foot*

This command will reduce the collision width of the sprite by the defined number of pixels. The left foot variable will reduce the left side of the sprite, the right foot variable will reduce the right side of the sprite. The FEET command, not only reduces the collision width, it also reduces the drop width of the sprite. In the diagram on the previous page, if you had defined a feet command for the sprite standing on the platform of 1,1 then the sprite would start to drop



The diagram above shows a sprite within its collision box on the left, on the right is the same sprite after the feet command 3,2 has been issued. The sprite will not drop, only when the central section of the sprite is over a platform.

Nodes are still detected from the sprites bottom left hand corner, shown as the small square in the right hand diagram.

If you are going to use the feet command, you should treat the sprite as if the sprite is narrower than it actually is. In the ladders example, the gap between the platforms should be equal to the new effective sprite width, in this case 8 pixels. The nodes will need to be offset by the amount of left hand feet value.

May I suggest you do not use the FEET command until you are fully familiar with node operations. After you fully understand how the nodes work, then experiment with the feet variables. The demonstration NODES2 shows how to program feet more effectively. Look through this demonstration and then load the drawing data into the designer and view the nodes.

The only way of fully understanding this fairly complex subject is to experiment. It may be frustrating, but, I hope, it will also be a lot of fun.

# INTRODUCING MISSILES

The SCADs package allows the user to define missiles. Once a missile is defined it can then be fired from any on screen sprite. A missile is a special sprite, it is a sprite that is placed onto the screen by another sprite.

There were a number of problems associated with missiles, but I believe the system we have created, although it may seem a little complex at first, will allow the user to create a sophisticated missile firing system for all the sprites on the screen.

The first problem we encountered was that of missile direction. In which direction should the missiles travel once they have been fired ? The simplest solution would be to fire a missile in the direction the sprite was moving, this solution seems fine for very simple move and shoot games, but there are a great number of games where this technique would simply not work. Space invaders has the main firing sprite moving left and right, whilst the missiles always travel up the screen. Asteroids type games have a sprite that may not be moving at all, but still firing missiles in the direction it is pointing. Lastly, you may have defined a stationary sprite that has been animated to spin slowly, you require the missiles to fire in the direction the sprite is currently pointing to, this too is clearly a problem.

Our solution to this dilemma is to link a missile, not to a sprite at all, but to the sprite IMAGE. We can than program every sprite image with the various missile attributes.

The missile attributes we need to define for each image are as follows.

**Missile image.** This is the image of the bullet that the sprite will fire.

**X offset.This** is the +1- offset for the x coordinate, from the top left hand corner of the missile firing image to the top left hand corner of the missile image.

**Y offset.** This is the +/- offset for the y coordinate, from the top left hand corner of the missile firing image to the top left hand corner of the missile image.

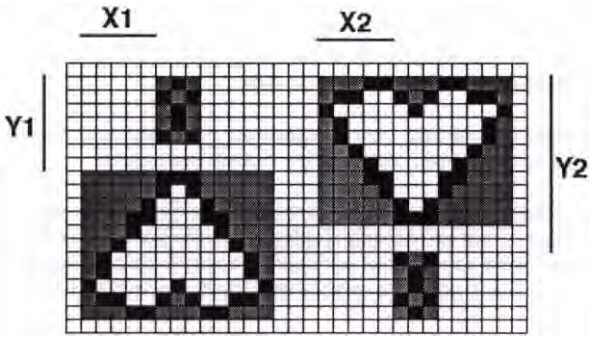
**X speed.** This is the +/- speed that the missile will move on the x axis.

**Y speed. This** is the +1- speed that the missile will move on the y axis.

We can define a different set of missile attributes for each image. In an animated sprite, the sprite number remains constant, it is only the image number that changes. The sprite for the main ship in Asteroids has the same sprite number throughout the whole of the game, however the image number is constantly changing. This image number can be used to determine in which direction the sprite is pointing. Therefore defining a separate missile attribute for every image used by the sprite will allow us to rotate the ship and fire the missiles in the correct direction.

To demonstrate the principle behind missiles please see the following diagram. The diagram shows two alien spaceships firing missiles. The shaded area around the ship is the sprite box. The first ship is pointing towards the top of the page and the second ship is pointing towards the bottom of the page. Both of these sprite images are linked to the

same sprite through the ANIMATION command.




In order to allow the missiles to fire in the correct directions we have to define two separate missile attributes.

Firstly we have to define the missile X and Y offsets. The offsets describe where the missile will be initially placed onto the screen.

The offset for the first image (left ship) would be X1 = +5 pixels, Y1 = +7 pixels

The offset for the second image (right ship) would be X2 = +5 pixels, Y2= -13 pixels

The offsets described the distances from the top left hand corner of the image that is going to fire the missile to the top left hand corner of the missile image.

We would then describe the X and Y missile speeds. The ship on the left is going to fire its missile upwards so therefore the speed setting could be X speed = 0, Y speed = +6. The ship on the right is going to fire the missile downwards so therefore the speed setting could be X speed = 0, Y speed = -6.

You will notice that the image of the actual missile is different for each ship, we would therefore have to define a different missile image number for each missile firing image. If the missiles had been circular than this would not have mattered as both missile firing images would use the same missile image.

The example we have shown here is very simple, but it does explain the theory behind missiles. Most of your games will probably have more than two animation directions, and so you will then have to define missile attributes for image that are moving to the left, right or even diagonally.

The design program allows you to create all of your missile information with ease. You will need to specify the image of the sprite that will fire the missile, the missile image, missile x and y offsets and the missiles x and y speeds.

There is one drawback to the missile firing method , but it is only a small one. A sprite image that has been defined to fire a missile will always use the same missile image

when firing. No matter how many sprites are using the missile firing image, the missile image will always be the same. ( I bet that has totally confused you, it confused me !). This is such a difficult subject to explain. It will all sink in eventually, honest ! There are a number of demos dedicated to showing you how the missiles work, working your way through the demos and looking at how the missiles are created in the Designer will enable you to understand missiles to a better extent

To give you a better insight into how to program your missiles, the following text will describe how the Supervisor will handle a fire missile instruction.

Once a sprite has been instructed to fire a missile, the Supervisor will look at the sprite and see which image is currently being displayed for the sprite. The Supervisor will then search for a missile definition for that particular image. If no missile definition has been created, then the sprite will not fire a missile and the fire command will be ignored.

If the Supervisor finds a missile definition, the information you defined for that image within the Designer program (speed, image and offset) will be transferred to a missile sprite. The missile sprite will then take the information supplied by Supervisor (speed and image) and place itself onto the screen at the offsets stated by the missile definition. The missile will then act in a similar way to a normal sprite, moving on the screen using the speeds that have been transferred from the missile definition.

The missile will ignore any nodes it passes over. If the missile hits the screen or window edge, it will disappear, ready to be fired again. If the missile strikes another sprite which is on a collision plane that the missile can detect, the missile will stop moving but it will not be removed from the screen. The sprite the missile collided with will also stop moving (if it was moving) and the collision will be reported to you. It will then be up to your program to decide what action to take. Once the missile collision has been reported you can then opt to animate the missile off the screen, possibly using an explosion animation. We use this technique in the full game demonstration on the demo disc. All of the commands on how to use missiles are described in detail in the Supervisor section of the manual.

When you are defining the missile commands within the Supervisor, you will also need to specify the collision plane the missile will be placed in and the collision planes the sprites will detect collisions in.

It may be a good idea to turn off the missile detection for the plane the sprite firing the missiles is placed in. This would eliminate any chance of the missile the sprite is firing detecting a collision with sprite that has just fired the missile. If you have not defined your missile offsets to clear the collision box of the sprite firing the missiles, then a collision between the sprite firing the missile and the missile that has just been placed onto the screen will occur as soon as the missile is placed onto the screen. There is also the possibility that the sprite firing the missile will move into the missile before the missile has moved away from the sprite. This could be a bit disastrous.

With careful planning you should have no problems with your missiles.

If you do not select the missiles to detect collisions with plane 128, then the missile sprites will pass straight through the centreground scenery.

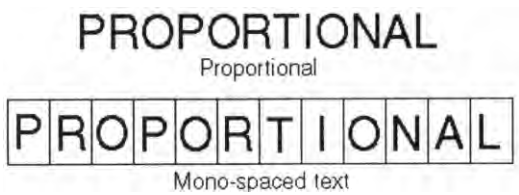
# INTRODUCING CHARACTER SETS

The Designer program will allow you to create up to 2 different character sets for use with the Supervisor. The Designers character set covers 96 characters, from space ( character 32) to character code 127 This range covers almost all of the usable characters including all numbers, punctuation marks, upper case letters, lower case letters and various symbols.

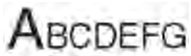
The Designers character sets are proportional, this means that the characters can be of various widths.

Normal character sets, as used by most computers, are mono-spaced, each character no matter how wide it is, takes up the same amount of room, an 'i' would take up the same amount of space as an 'm'. This can leave ugly spaces between the letters.

A proportional character set allows the letters to be of varying widths. This manual is printed using a proportional character set. You should notice that there is only a very small fixed width gap between each letter.




Text referenced from top left hand corner



Text referenced from bottom left hand corner

The diagram above shows the difference between mono-spaced text and proportional text.

As I have already mentioned in the previous pages, all sprites are referenced from there top left hand corners. If you look at the two examples of text above you will see that this method will not work very effectively with text.

Text, therefore, is referenced from its BOTTOM left hand corner. This will allow you to mix characters of any size on the same line. If you are going to use descenders within your characters, you should take special care to ensure the rest of the characters you design will line up properly.

When you are defining your character sets, please bear in mind the discussion on masking in the *introducing masking* section of the manual. Any characters that are going to be displayed as part of the score, or any character that is going to overwrite existing characters on the screen should have all of the blank areas around the character filled in with a mask colour.

# INTRODUCING SOUND EFFECTS

The Sam Coupe is supplied with an excellent sound effects chip, but unfortunately this is not backed up by good BASIC support. There is very little information supplied within the BASIC manual about the sound chip. The technical manual does supply a good deal of information, but it is a bit, Er, technical. Even when you understand how the sound effects can be programmed, you are still left with the awesome task of writing your own sound processing routines. When programming SCADs, we spent a great deal of time in trying to make using the sound chip as simple as possible to program.

The Designer program will allow you to create up to 32 different fixed sound effects for use with the sprite automatic effects ie. bouncing, firing, exploding, etc. The Supervisor will allow you to create an infinite number of sound effects.

All sounds can be broken down into two distinct areas, both of these areas are separate from each other but need to be combined in order to create life-like sounds. The different parts are called Tone and Volume.

The volume of the sound is very easy to understand. There are not many sounds that maintain a constant volume throughout the length of the sound. Even when you have you stereo set at a constant volume the level of sound coming from your speakers is constantly varying. The volume control on your stereo simply sets the average volume of the sound that will be played through the stereo speakers.

The tone of the sound is also very simple to understand. The tone or pitch of the sound determines how high or low the frequency of the sound will be. Tapping a wine glass *gently* with a pencil will produce a high pitched sound, thumping a full suitcase will produce a very low pitched sound (thud). As with the volume, there aren't many tones that remain constant throughout the length of the sound. If you looked at an *envelope* of a sound through an oscilloscope you would find that all but the purest of sounds have a varying tone.

The Sam Coupe sound chip will allow up to 16 levels of volume and 2048 levels of tone.

We can simulate these true-to-life sounds using a technique called 'enveloping'.

Enveloping consists of a table of data that specifies by what degree the tone/volume will vary throughout the length of the note. Look at the table below as an example of how an volume envelope is defined.

| No of Steps | Step Size | Delay Time |
|---|---|---|
| 5 | +3 | 2 |
| 40 | 0 | 2 |
| 15 | -1 | 2 |

Each row within the table represents an envelope stage, this example of a volume envelope has 3 stages. Each stage consists of three pieces of data; steps,size and delay. The first stage in the envelope instructs the sound chip to increase the volume by a value of three and then wait for 2/100ths of a second, this stage will be repeated 5 times (including the initial setting). So after 1/10th of a second (2/100ths x 5) the volume

will be at a value of 15 (maximum volume), assuming the volume started at zero. The volume will then remain constant for 80/100ths of a second (stage 2). The volume will then slowly become quieter as stage 3 is processed, until 1.2 seconds (120/100ths) after the sound started, the volume level will be back to zero (no sound).

The envelope described above is a typical example of a simple instrument sound. An instrument sound has three distinct stages, Attack, Sustain and Decay. The attack occurs when the note is first played, the volume builds up very quickly until it reaches the highest point of the note. The volume then goes into the sustain section of the envelope, here the volume stays at a constant level. Finally the note enters the decay stage as the note starts to fade away to nothing. All instruments follow these three stages. The difference between a piano and a violin are the values used within the stages. A piano has a much longer decay period than a violin. The Designer allows you to experiment with the different values within the stages to create an unlimited number of effects.

The example above shows a sound having three different stages, the Designer will allow you to design up to 8 different stages per envelope and you may create up to 32 different envelopes.

The tone envelope is designed in exactly the same way as the volume envelope, you may create up to 8 different tone stages and up to 32 different tone envelopes.

Now you understand how to define a tone and volume envelope we will now need to link them together to create a complete sound. As well as the tone and volume envelopes there are a number of other variables that link together in order for you to play a note.

The Sam Coupe's sound chip can produce sounds on 6 different channels, this means the Sam can play up to 6 notes at the same time. There is a slight drawback if you wish to use noise, but this will be explained later in the manual. If a note is already playing on a channel and you issue another sound command that wishes to use the same channel, the new sound will not be played until the previous sound has been completed. The new sound will be stored in the sound queue

A sound queue's advantage is that the Sam will never forget to play a sound, however there is one big disadvantage. If you set a sprite to fire a missile and the channel the missile sound will use is currently playing a sound, the fire sound will not be played until the sound that is currently playing has finished. This may introduce an unacceptable delay in your fire sound being played. To get over this problem, when setting the channel number, you may select for the sound you are defining to clear the sound queue for that particular channel. This would then cancel any sound that is currently playing and play the new sound instead. You may, of course, plan your sound effects to use different channels and this may alleviate the problem without resorting to cancelling the sound queue (if there is one).

The Designer program will allow you to define which of the six channels the sound will be played from.

As you may know, the Sam Coupe can play sounds in stereo. You may specify the side of the stereo field you wish your sound to play. You can opt for either left,right or both. This data is entered with the channel data. As well as the channel number you will also

need to define the stereo position of the sound.

The next variable that needs to be defined is the initial octave and tone values. As the name suggests these values are the starting variables used by the sound command. If you specify a tone envelope to use, then these variables will obviously change whilst the sound is being played. The octave value can be any number between 0 and 7, 0 being the lowest possible octave, producing the lowest tone (bass) and 7 being the highest value, producing the highest tone (treble). The tone value can be any number between 0 and 255, 0 being the lowest tone value for a given octave and 255 being the highest.

The next variable that needs to be defined is the sound duration. This variable determines how long the sound will be played for. You may express the length of the sound in two different ways. The simplest way is to specify the exact length of the sound in seconds and hundredths of a second. If this method is used then the sound will be played for a fixed length of time and then stop. If you have defined a volume and/or tone envelope this may not have finished before the sound comes to an end.

The second method of specifying the length of sound is to link the sound to a volume envelope. When using this method, the sound will finish when the volume envelope has been completed. If you are not specifying a volume envelope then this method can not be used. This option will also allow you to repeat the volume envelope a number of times before the sound is completed. Please note that using this option links the duration of the note to the VOLUME envelope and NOT to the TONE envelope.

The next stage in defining your sound command is to specify the initial volume. This is the starting value for the volume, it can be anywhere in the range 0 (silent) to 15 (full blast). The value of the volume will change if you have defined a volume envelope.

Next comes the noise variable, this variable will allow you to mix noise into your sound. Noise is the sound you hear when you switch on a radio and tune the radio so that it is not receiving any station. This type of sound is used to simulate explosions. There are 4 different types of noise you may use within your program, there are three fixed frequency noises and one variable frequency noise. You may select to play just noise, or just tone or a mixture of noise and tone. This is all described further into the manual under the sound command heading.

There is one complication when using noise. There are only two noise generators built into the Sam Coupe's sound chip and this has to be shared by all 6 channels. This means that if you issue three sound commands on three different channels then there will obviously be a problem, as not all of the channels will be able to use the noise registers (3 channels trying to use 2 noise registers). To overcome this problem you should limit using noise to 2 channels; channel 1 and channel 4.

The next variable used by the sound command is the volume envelope. You do not need to specify any envelope at all, but this will make the note sound lifeless and boring. You may specify any volume envelope between 1 and 32. Remember that you may link the duration of the sound to the volume envelope.

The last variable to be specified is the tone envelope. You do not need to specify a tone envelope at all, but the note normally sounds better if you do use an envelope. We have discussed defining a tone envelope in the previous chapters.

After you have selected a tone envelope, you will need to stipulate whether the tone envelope will play just once or be repeated. If you select repeat, then the tone envelope will repeat until the note has finished playing.

If you define a tone envelope that will take 10 seconds to complete and you have set the duration of the sound to last only 5 seconds; the tone envelope will not be completed. On the other hand if you define a tone envelope that will take 10 seconds to complete (without repeat) and you have set the duration of the sound to last for 15 seconds; the last 5 seconds of sound will have a constant tone.

If, while a tone or volume envelope is being played, the maximum value for the tone/volume is exceeded, the value will loop. ie if the volume is currently set to 14 and you increase the volume by a value of 3, the new volume value will be 1. You can use this technique to create special effects.

The best way of learning about the sound effects is to experiment with the sound effects generator within the Designer program. Have some noisy fun.

**Important Note**

The sounds will be updated every 1/100th of a second. When you are using the Supervisor and you start to place sprite onto the screen, the sound effects will start to slow down, that is they will take longer to play. The Supervisor pushes the processor inside the Sam to the absolute limit, moving sprites takes the highest priority and so the sound effects may start to slow down. This is not a problem when there are no sprites on the screen.

The slow down in the sound effects is only noticeable when the sounds have a long duration or large changes in tone/volume. Try to make your sound effects as short as possible.

By experimenting with the sounds and moving sprites around the screen, you will be able to gauge how to make your sound effects sound great and not experience any detrimental effect.

# USING THE DESIGNER PROGRAM

## *LOADING THE DESIGNER*

To enter the Designer program, turn the computer off, wait 5 seconds and turn the computer back on again. Insert the SCADs system disc and press the f9 key. Select the Designer option and wait while the program loads into memory.

If the screen displays an error message saying ALIEN CODE FOUND IN MEMORY the program has found that somebody else's program has been loaded into memory ie MASTERBASIC. This should be removed and you should try again.

Immediately after the Designer has loaded into memory, the message

**INSERT KEY DISC**

will be displayed. You should now insert your original SCADs system disc and press the SPACE BAR. The Designer will then verify that the original disc has been inserted the drive. Once the checks are completed, the designer will display the main design menu.

If the disc you inserted into the disc drive is not the original disc supplied by Glenco Software, you will not be able to proceed with using the Designer program.

The design program uses a system called WIMP, which stands for Windows, Icons, Menus and Pointers. This system is very easy to use. You control an arrow that can be moved around the screen. To select an option, simply point the arrow at the option and press the select key, the option will then be selected.

There are ways of moving the pointer around the screen, the keyboard and the mouse. The default method is to use the keyboard.

## *KEYBOARD*

Pressing the arrow keys on the right hand side of the keyboard will move the pointer in the relevant direction. To select an item, place the pointer over the option you require and press the space-bar. To escape from any option press the ESC key

## *SAM MOUSE*

The Sam mouse is the preferred device for using this program. Moving the mouse around the table will move the pointer around the screen. To select an item, place the pointer over the option you require and press the left hand mouse button. To escape from any option press the right hand mouse button.

To select the Sam mouse option you should press SYMBOL M. Selecting this option will allow you to use the mouse and the keyboard. If you do not have a mouse attached to your computer and you select the SYMBOL M option the pointer will not work correctly. To select the keyboard, the default setting, you should select SYMBOL K.

## *BLUE ALPHA MOUSE*

The Blue Alpha mouse works in a slightly different way to the Sam mouse. The Blue Alpha mouse works fine with the Designer program, but the response is not a accurate as the Sam mouse. Selecting an item with this mouse is exactly the same as selecting an item with the Sam mouse.

To select the Blue Alpha mouse option you should press SYMBOL K. Selecting the SYMBOL M option will not work with this mouse. The SYMBOL K option is the default input setting and so you do not need to select this option unless you have selected the mouse option using SYMBOL M.

## *JOY-STICK*

Moving the joy-stick in any direction, will move the pointer on the screen in the same direction. The joy-stick button acts as the select key. You should press the ESC key to step back a menu.

There is a short cut to selecting the various menu options. If you type the first letter of option you require, the pointer will immediately move to the selected option. If there is more than one option with the same starting letter, the pointer will rotate through the options one after another. To select the option the pointer is pointing to, press the select button.

As I have already stated you select an option by moving the pointer over the option and pressing the select button. If you are moving the pointer over a menu, the option that the pointer is placed over will be highlighted. This highlighted part of the menu is the option you will select if you press the select button.

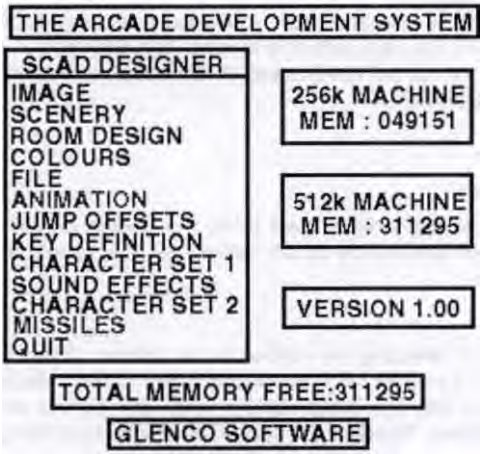
You may notice that some of the menu options are displayed in a different colour and that the option will not highlight when you move the pointer over them. You cannot select that option at that time. ie You will not be able to select VIEW if there are no drawings defined to view.

If you mistakenly select an option and an event occurs that you don't require you can usually press the ESC key to abort the last choice. Pressing the ESC key is the best way of back tracking through the maze of menus this program uses. If you are using the mouse you can achieve the same effect by pressing the right hand mouse button.

There are certain sections within the Designer program where you must select an option, the ESC key will not work. In these circumstances, one of the options will usually cancel the menu and step you back to a previous menu.

# MAIN MENU

Whenever you load the Designer program into memory, the first page that is displayed is called the Main Menu screen. This screen will allow you to access all of the functions within the Designer package.




This is how the Main Menu screen would be displayed on Sam having 512k of memory and no drawings loaded There are a few points to note about the screen

The ANIMATION and MISSILE options are displayed in a different colour to the rest of the options within the Main Menu. The options are a different colour because you cannot select them at present. You will need to create at least 1 sprite image before they may be selected. This technique is used throughout the Designer program, if an option is displayed as a different colour it can not be selected.

On the right hand side of the screen is displayed the memory status. There are two boxes displaying the memory free for both the 256k and 512k coupe's. Running along the bottom of the screen is another menu, TOTAL MEMORY FREE:000000. The menu will display the current memory free for your particular machine. If you have 256k fitted, the memory free displayed in the bottom menu will mirror the memory free displayed on the 256K MACHINE menu. If you have a 512k machine the memory free displayed in the bottom menu will mirror the memory free displayed on the 512K MACHINE menu.

These menus are informing you how much memory you have available for designing all of your graphics. You do not need to leave any memory free for writing your program, this is already reserved, you may use all off the memory displayed in the bottom menu for your graphics.

If you own a 512k machine, you may be wondering why the Designer program tells you how much memory you have free if you only had 256k installed. The answer is simple, this menu will allow you to plan your graphics to run on all Sams and not just 512k

machines. If you use more memory on your graphics than a 256k Sam could handle, this menu display will inform you by displaying a minus sign before the amount of memory free. Of course if a minus sign is displayed within the 256K MACHINE menu, then the figure after the minus sign is **not** memory free, but how much memory you will need to free in order to get your programs working on a 256k Sam. I would not advise you to try and restrict all of your programs to run on a 256k Sam, the SCADs package looks so much better when working with 512k, you can create massive games with 512k.

If you own a 256k machine, you may be wondering why the Designer program tells you how much memory you would have free if you owned a 512k machine. The main reason the Designer displays the 512K MACHINE memory free menu, is to give you the incentive to save up and purchase the 256k memory expansion card. It really is worth expanding your computer to 512k, SCADs will allow you to do so much more with the extra memory. Just look at the difference between the two memory free values, just imagine what you can achieve with all that extra memory.

As you create images and scenery, the amount of memory free for your program to use will obviously come down, graphics do tend to use up loads of memory. There are a number of functions within the Designer program that use no memory at all. The following options will not alter the amount of memory free

COLOURS
JUMP OFFSETS
KEY DEFINITIONS
SOUND EFFECTS
MISSILES

You may use these options as many times as you like and the memory free will remain the same.

When the memory free gets below a fixed value the menu

**WARNING:LOW MEMORY**

will be displayed.

If you try to do something that uses more memory than is currently available, the following menu is displayed.

**WARNING:OUT OF MEMORY**

## Useful Keys

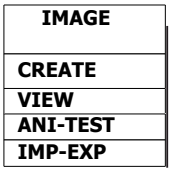
You may change the colour of the pointer at any time (except within the colour menu) by pressing SYMBOL C.

You may rotate the pointer at any time by pressing SYMBOL R.

# IMAGE

Images are used by sprites within the main Supervisor program, you may create up to 255 different sprite images.

To create an image, go to the Main Menu and select the IMAGE option. The screen will clear and you will be presented with the image menu.

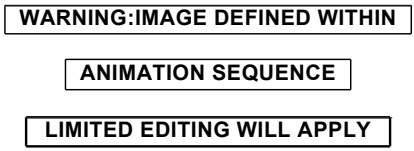



This menu will allow you to select all of the functions related to creating a sprite image. If you have not defined any images yet, the VIEW and ANI.TEST options will not be selectable.

# IMAGE / CREATE

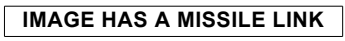
Select the CREATE option. The screen will clear and the image number menu will be displayed. This menu shows the option numbers from 0 to 255, these numbers represent all of the different images you can create. If there are no images defined, all of the numbers will be displayed in blue. If you have previously defined an image, the image number relating to that image will be displayed in red.

Select an image number. If the image you have selected is printed in red, the dimensions menu will be skipped and you will jump to the main design screen, where your drawing will be displayed. If the image you have selected has been used in any animation sequence, the following menu will be displayed




This means that you will not be able to ERASE the image or alter the DIMENSIONS, until the image has been removed from the animation sequences.

If the image number you have selected is used by the missiles option then one or both of the following menus may be displayed.




This means that the image has been defined to fire a missile, if you try to erase this image, the link between this image and the missile will be broken.

This means that the image has been defined as a missile, if you try to erase this image, the links between this image and every other image that tries to fire this image will be broken.

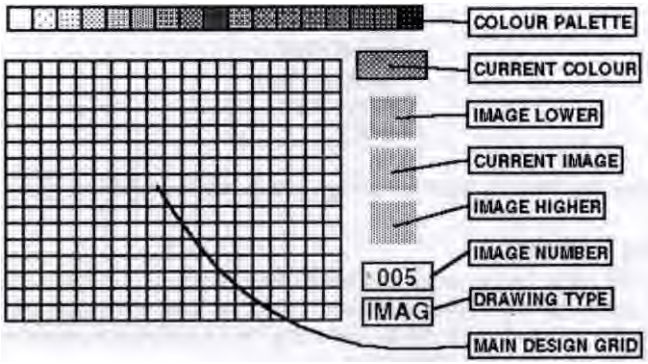
The next menus to be displayed are the X and Y dimension menus. These menus allow you to define the size of the image you are going to design. The numbers within the menus correspond to pixel sizes. Select the X dimension for your drawing and the Y dimension menu will be displayed. If you have selected the wrong X dimension, don't worry, simply press the ESC key and you will return to the X dimension menu. After you have selected the Y dimension, the two selections you have made are displayed within two separate menus. You will now be asked to confirm your selection with the following menu.

| ACCEPT |
| --- |
| REJECT |

Selecting the REJECT option will return you to the Y dimension menu. Pressing the ACCEPT option will take you to the main design screen.

## MAIN DESIGN SCREEN

The main design screen is the screen on which you will define all of your sprite images. The screen is extremely easy to use and needs very little explanation. Across the top of the screen is the colour palette you selected when using the COLOUR option from the Main Menu. On the left hand side of the palette is the paper colour, although you can't see it, it is there. On the right hand side of the palette is the mask colour. The mask colour is not a colour as such. Although it is a checked colour, and it shows as a checked colour in the enlarged design grid, it will not show itself on the scaled image on the right hand side of the screen. Please refer to the section on masking for further information.




On the right hand side of the screen and working downwards from the palette menu, the next item we come across is the current colour box. The current colour box is displaying the colour that is currently selected, this menu will also show you if the mask has been selected.

Down from the current colour is the image lower drawing. If you have defined an image with a lower number to the current image, the image will be displayed here. This image is displayed to try and make designing animated images easier.
Below the image lower drawing, is the current image drawing. This is the actual size of the drawing you are currently working on, an enlarged version of this drawing is displayed in the main design grid.
Below the current image drawing is the higher image drawing. This shows the next image higher than the image that is currently displayed. If there is not an image that is higher than the current image, this area will remain blank.

Below the higher image drawing is the current image number menu. This is the number of the image you are currently working on.

Below the image drawing number is the drawing type, in this case the drawing is an image and so the menu displays IMAG.

To create a drawing, simply move the mouse pointer around the main design grid and press the SELECT key to place the currently selected colour onto the grid. Any changes you make within the main design grid will be mirrored in the current drawing image.

To select a colour, simply move the pointer to the required colour along the top of the screen and press the SELECT key. There is a shortcut for selecting the colours, press the SYMBOL key and then either Z to move down a colour, or X to move up a colour.

There are a number of options available to you within the design screen, to access the design screen menu, press the ESC key. The following menu will be displayed.

| DESIGN |
|---|
| ABORT |
| WIPE |
| MOVE |
| SAVE-QUIT |
| DIMENSIONS |
| EFFECTS |
| COPY |

To return from this menu to the main design grid, press the ESC key again.

### *DESIGN* / ABORT

This option will allow you to erase the image from memory. After selecting this option you will be asked to verify your selection. Selecting 'yes' will erase the drawing and return you to the image number menu. Selecting 'no' will allow you select another option or press ESC to return to the design grid.
If the image you are trying to abort is defined within an animation sequence, you will not be able to erase the drawing.

If the image you are trying to abort is defined as either a missile firer or as a bullet the

following message will be displayed before asking you to verify your choice.

| THIS WILL ERASE RELEVANT |
| --- |
| MISSILE INFORMATION |

If you select to erase the image, all of the links to the missiles for this particular image will be erased.

## *DESIGN* WIPE

This option will allow you to clear the drawing to background colour. After selecting this option, you will be asked to verify your actions. If you select 'yes' the image will be cleared and you will return you to the design grid.

## *DESIGN* MOVE

This option will allow you to shift the image around within the design grid. After selecting the MOVE option, the following menu will be displayed.

| SHIFT |
| --- |
| UP |
| DOWN |
| LEFT |
| RIGHT |

### *SHIFT* / UP

Selecting the UP option, will move the drawing up one pixel. A clear row of pixels will be printed along the bottom of the screen. If any part of the drawing is shifted off the top of the design grid, it will be erased.

### *SHIFT* / DOWN

Selecting the DOWN option will move the drawing down one pixel. A clear row of pixels will be printed along the top of the screen. If any part of the drawing is shifted off the bottom of the design grid, it will be erased.

### *SHIFT* / LEFT

Selecting the LEFT option will move the drawing to the left one pixel. A clear column of pixels will be printed down the right hand side of the screen. If any part of the drawing is shifted off the left hand side of the screen, it will be erased.

### *SHIFT* / RIGHT

Selecting the RIGHT option will move the drawing to the right one pixel. A clear column of pixels will be printed down the left hand side of the screen. If any part of the drawing is shifted off the right hand side of the screen, it will be erased.

## *DESIGN* / SAVE-QUIT

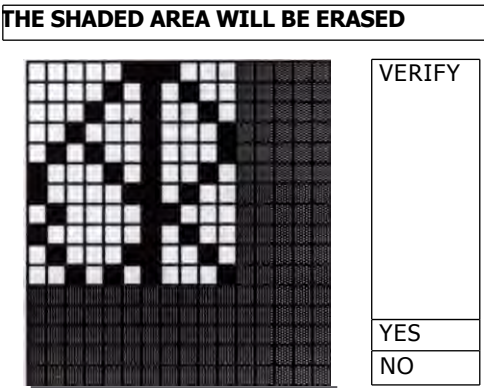
This option will save the drawing that you have created and return you to the image menu.

### *DESIGN I* DIMENSIONS

This option will allow you to alter the dimensions of the drawing that is currently displayed in the main design grid. After selecting this option a menu will appear at the top of the screen showing the current X and Y dimensions. You will then be prompted to select a new X dimension and then a new Y dimension. These menus are identical to the initial dimension menus. After you have made your selection and confirmed the change through the ACCEPT/REJECT menu the option will be processed.

If both of the new dimensions are larger or equal to the old dimensions, the screen will simply be redrawn with the new dimensions and your image placed into the top left hand corner of the new grid.

If one or both of the new dimensions is smaller than the old dimensions then the area of your image that will be erased when the new dimenions are set will be shaded.




You will then be asked to verify the new dimensions you require, if you have made a mistake, or decide not to reduce the dimensions to that degree, select the NO option. The drawing will then be restored to its original size.

If you select YES the image will be redrawn to the new size, any part of the image that was shaded, will now be erased.

### *DESIGN I* EFFECTS

This option will allow you to alter the way the image is displayed within the main grid. After selecting this option, the following menu will be displayed.


#### *EFFECTS* MIRROR

Selecting the MIRROR option will mirror the image within the main design grid, that is, the image will turn around. If the image was facing to the left, it will now face towards

the right. Selecting the MIRROR option again will turn the image back to the left.

### *EFFECTS* / INVERT

Selecting the INVERT option will invert the image within the main design grid, that is, the image will turn upside down. If the image is facing down, it will now face up. Selecting the INVERT option again will turn the image towards facing down again.

### *DESIGN I* COPY

This option will allow you to copy the current image to a new image number. This option is useful when you want to animate an image. Create your first image and then copy it to a new position, then alter the new image slightly, and then copy the new image into another free position and so on.
After you have selected this option, the screen will clear and you will be presented with the IMAGE NUMBER menu. This menu contains all of the image numbers from 0 to 255. Some of the numbers may be coloured differently. All of the image numbers in red represent images that have already been defined, all of the image numbers in blue represent images that have not been defined.

You may only copy your current image into another image number that has not been defined. You will find that by moving the pointer around the screen, you can only highlight image numbers that are printed in blue.

If you do not wish to copy your image press the ESC key and you will be returned to the main design grid.
If you select an image number, the image you have just been using will be transferred to the new image number. You will be returned to the design grid. Please note that the image number on the right hand side of the screen shows the new image number, any modifications you make to the image that is displayed will be made only to the new copy of the image and not to the old copy.

## Key Shortcuts

There are a number of shortcuts you can take while working in the main design screen, to access the key-press shortcuts, press and hold down the SYMBOL key and then select the letter from the table below.

| LETTER | ACTION |
|---|---|
| Z | Move current colour down 1 colour |
| X | Move current colour up 1 colour |
| C | Change the colour of the pointer |
| R | Rotate the pointer clockwise |
| U | Move up to the next image, saving the current image |
| D | Move down to the next image, saving the current image |

## *IMAGE* / VIEW

This option will allow you to view either a range of your defined drawings or all of your defined drawings.

Select the VIEW option. You will not be able to select this menu if there are no drawings defined.

After selecting the VIEW option the screen will clear and you will see the following menu.

| VIEW |
| --- |
| ALL |
| RANGE |

If you wish to see all of the defined images select the ALL option. This will skip the next menu and go straight to the view screen

If you wish to see only a small range of your images, select the RANGE option. After selecting this option, the screen will clear and you will be presented with the IMAGE NUMBER menu. The drawing numbers will be printed up in two different colours. The RED numbers represent images that have been defined, the BLUE numbers represent images that have not been defined.

Move the pointer to the first number you wish to display and press the SELECT key. At the top left hand corner of the screen a small menu will be printed showing the START DRAWING. You may now select the FINISH DRAWING. If you have made a mistake in selecting the START DRAWING, press the ESC key and you can then re-select the START DRAWING.

After you have selected the FINISH IMAGE number, you will then be asked to verify the image range. Select NO to return to the selection process or select YES to view the images within the range you have chosen.

The number range you have chosen will now be displayed on the screen. There is room on the screen to display up to 24 drawings. If the range you have selected contains more than 24 drawings, the first 24 drawings will be displayed. Pressing the SELECT key will then display the next batch of drawings.

You can abort the VIEW option at any time by pressing the ESC key.

You will see that the drawing number that corresponds to the drawing is printed underneath the drawing.

## IMAGE / ANI - TEST

You may have noticed there are two different animation options. The first animation option is accessed from the Main Menu screen, the second animation option is selected from the IMAGE menu. There is a difference between the two options.

The ANI-TEST option that can be selected from the IMAGE menu is used to test the

animated images in one direction. The animation data is forgotten once you have finished using the option.

The ANIMATION option that can be selected from the Main Menu is used to store the animations for use within the Supervisor program.
We will describe here the use of the ANI.TEST option. The ANIMATION option from within the Main Menu will be described further into the manual.

To test a temporary animation, select the ANI-TEST option from within the IMAGE menu. The screen will clear and you will be presented with the IMAGE NUMBER menu. You are now required to select the first image for the animation and the last image for the animation, this selection process is identical to the selection process for defining the range of image numbers within the VIEW/RANGE menu.

Please remember the rules on animation discussed in the *introducing animation* section of the manual. If you make a mistake within the range selection stage, a self explanatory error message will be displayed.
Once you have selected the range of images to be animated, the next menu will be displayed

| MOVEMENT |
|---|
| UP |
| DOWN |
| LEFT |
| RIGHT |
| STILL |

### *MOVEMENT I* UP . DOWN . LEFT . RIGHT . STILL

This menu will allow you select the direction in which the animated image will move. I think the menu is self explanatory and so I wont go into detail what the options mean.

After selecting the image direction, you will then be prompted for the delay between movements.

| DELAY |
|---|
| KEY |
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |

This menu will allow you to select the delay between moving the image on the screen. Selecting 0 results in no delay, and the image moving at full speed. Selecting 8 results in the image moving at it's slowest speed.

The KEY option will allow the image to move a frame at a time, whenever the SELECT key is pressed.

**INSERT DATA DISC**

You should now place into the disc drive the disc containing the graphical data you wish to import. If there is a fault on the disc or the disc isn't formatted, an error message will be displayed. You should rectify the error and then press the SELECT key. The 'insert data disc' menu will then be displayed.

The Designer program will now look through the discs directory looking for any SCREEN$ format files. Once a SCREEN$ file has been found, the Designer will check that the SCREEN$ file has been designed in Sam MODE 4. The Designer will then make a list of all the files on the disc that are both in SCREEN$ format and in mode 4. If there are no files that meet the specification the message

**NO RELEVANT FILES**

will be displayed. Pressing the SELECT key will then return you to the INSERT DATA DISC menu option. You will now be able to try and find a disc with the correct type files. Remember Flash files are not SCREEN$ files. They need to be converted.

Once the Designer program has found some files of the correct format, two menus will be displayed. The menu on the left hand side of the screen shows a list of all of the files on the disc that meet the requirements for importing files. The menu to the right has two options. The first option TRY ANOTHER DISC is self explanatory, selecting this option will take you back to the INSERT DATA DISC menu.

If you select the second option, the menu on the right will disappear and you will then be able to select a filename from the directory menu on the left hand side of the screen. If you do not wish to select a file press the ESC key.

Once you have selected a file, the file will be loaded onto the screen. Once the screen has loaded, the colour selection menu will be printed.

| **COLOUR SELECTION** |
| --- |
| **IMPORTED COLOURS** |
| **CURRENT COLOURS** |

***COLOUR SELECTION / IMPORTED COLOURS***

Selecting this option, IMPORTED COLOURS will be change the current colour palette to the colours used by the imported graphics screen. If you already have graphics defined within the Designer, selecting this option may alter their colours and make them look hideous.

***COLOUR SELECTION*** I **CURRENT COLOURS**

If you select the second option CURRENT COLOURS then the colour palette will not be changed. However if the graphics you are importing use different colours to the colours you are using, the imported graphics may look awful.

The choice is yours. You can press ESC whilst this menu is being displayed. If you have changed your mind about importing this particular screen, select the CURRENT COLOURS option.

Once you have decided what to do about the colours the next menu will be displayed.

| GRAPHICS GRABBER |
|---|
| STANDARD GRABBER |
| GRAB THEN CLEAR |
| DISC OPERATION |
| GO TO MAIN MENU |

There are two types of grab available. Both of the grabs will copy a drawing from the screen and store it in the Designers memory. The difference between the two grabs is what happens after the drawing has been stored in memory. The standard grab will leave the drawing intact after it has been copied. The grab then clear option will erase the drawing from the screen after it has been stored in memory. This grab then erase option is useful when you have to grab a drawing or piece of scenery that is larger than 32 x 32 pixels. After you grab a section of the drawing you require, the section disappears, this helps you to keep track of which part of the drawing or scenery has been grabbed.

***GRAPHICS GRABBER / STANDARD GRABBER***

This option will select the standard grab. The screen will remain intact after the drawing has been grabbed.

***GRAPHICS GRABBER / GRAB THEN CLEAR***

This option will select the clear type grab. Once the drawing has been copied into the Designers memory, the area of screen the drawing was removed from will be cleared.

***GRAPHICS GRABBER / DISC OPERATION***

The DISC OPERATION option can be selected if you do not wish to use the screen that has been loaded. This option will allow you to go back into the disc operations and select another screen file.

***GRAPHICS GRABBER /* MAIN MENU**

This option will allow you to return to the Main Menu. The screen that has been loaded into memory will be cleared.

Once you have selected the type of grab you are going to use, a large menu will appear on the screen. This menu will be the IMAGE NUMBER menu. You may notice that some of the numbers are coloured red and the rest of the colours are coloured blue. The red numbers represent images that have already been defined. The blue numbers represent free image places. You should select the number of the image you wish to grab. You will note that you can only select blue numbers, the Designer will not allow you to overwrite existing images.

The next menu that will be displayed is the X Dimension menu. Make your selection. If you are unsure of the size of the graphics you want to grab, don't worry. Select a size you know will be bigger and you can always trim them down to size at a later date. After selecting the X dimension you should select the Y dimension. After selecting the Y dimension you will be asked to confirm the size of the graphics you wish to grab. Again this is an ACCEPT/REJECT menu.

Once you have made your selection the menus will disappear and you can see the SCREEN$ file you have loaded. On the screen there is a pointer which you can move around in the usual way, you should also notice a box with a magnified view of where the pointer is pointing. This is your grab box. The size of the box depends on the size of the object you are going to grab. As you move around the screen the image in the grab box changes. If you get too near the grab box it will move to a new position so that you can see the graphics that were underneath the box. Once you have the graphics you wish to grab in the grab box, press the SELECT key. The image from the grab box will be transferred to the number you selected in the IMAGE NUMBER option screen. Once the image has been transferred to memory the IMAGE NUMBER option screen will be displayed again. You may now select another image number you wish to grab.

If whilst you are in the grab screen, you feel the dimensions you selected are too small or too large you can press the ESC key, and this will allow you to select a new set of dimensions.

To end your grabbing session, keep pressing the ESC key until you get back to the Main Menu.

## *I-O* / EXPORT

Once you have created a number of different graphics you may need to take some graphics from one drawing data file and transfer the graphics to another drawing data file. To do this you will need to export the graphics. Exporting the graphics will print up to 35 of your images onto the screen. The Designer program will then save the screen as a standard SCREEN$ file. You may then import the graphics into a different Designer file or a separate art package.

Select the EXPORT option.The screen will now clear and a large menu will appear with the numbers from 0 to 255 displayed. On top of this menu is another menu. The top menu will display how many images you have selected to export. This menu will currently display 0. You will notice that the numbers in the Main Menu are two different colours. The dark blue numbers are the numbers you can select to be exported. The numbers printed in the lighter blue have not been defined and so can not be exported.

You may select up to 35 images to be exported in one screen. To select an image, move the pointer over the number representing that image and press the SELECT key. You will notice that the number changes to RED and the NUMBER TO EXPORT menu has been incremented. You may select up to 35 different graphics using this method.

If you wish to DE - SELECT an image number, move the pointer over to the number, which is now displayed in red, and press the SELECT key again. The SELECT key will toggle the number on and off. Once you have selected all the numbers you wish to export press the ESC key. The following menu will be displayed.

| EXPORT |
|---|
| RETURN |
| ABORT |
| SAVE |

### *EXPORT* / RETURN

Selecting the RETURN option will simply remove the menu and allow you to continue selecting or de-selecting image numbers.

### *EXPORT*/ABORT

If you select the ABORT option the whole of the exporting operation will be cancelled and you will return to the Image menu.

### *EXPORT*/SAVE

If you select the SAVE option, the screen will clear and all of the images you selected will be printed onto the screen. You will now see the menu

**INSERT DATA DISC**

displayed. You should now insert the disc you wish to save your exported drawings on, in the disc drive (a) and press the SELECT key. If you do not wish to save your drawing data, press the ESC key. After pressing the SELECT key, the disc drive will start.

If there is a problem with the disc or the disc drive, the screen will clear and a disc error message will be displayed. Take a note of the disc problem and try to rectify the error. If you now press the SELECT or ESC keys, the error message will disappear and the ' insert data disc' menu will appear.

After a short time, two menus will be printed onto the screen. The menu on the left will display a list of all of the current screen files that are already stored on the disc. If there are no screen files stored on the disc, the message

**NO RELEVANT FILES**

will be displayed. The menu to the right of the screen will be displaying four options.

The first option TRY ANOTHER DISC is self explanatory, selecting this option will take you back to the INSERT DATA DISC menu.

The second option SELECT A FILE can only be highlighted if a directory has been printed on the left hand side of the screen. If you select this option, the menu will disappear and you may then select a filename from the list on the left hand side of the screen. The data printed on the screen will then overwrite the data in the filename you have selected. You should use this option with care, you could lose a lot of valuable work by being careless. The filename is verified before it is overwritten and this should eliminate any mistakes. Remember if you make a mistake press the ESC key.

The third option ENTER FILENAME will allow you to create a new name for the data in memory. Selecting this option will bring up the ALPHABET menu. You may now select the name of the data you wish to save.

The fourth option SAVE AS:SCREEN will save the screen data as a file called SCREEN. This is the default setting and will never change.

No matter which method you use to select a name for your data file, the file will not be

saved straight away. A menu will be displayed towards the top of the screen.

| PLEASE CONFIRM DISC SAVE FILENAME |
| --- |
| **SCREEN** |
| **YES**<br>**NO** |

If you select the YES option, the file will be saved to disc.

If you select the NO option, you will be returned to the previous menu.

Please do not be alarmed if the data takes a long time to save. After the data has been saved it is very carefully verified to ensure no errors have occurred. If an error is found a message will be displayed telling you exactly what is wrong.

Verifying a data file takes 10 (ten) times longer to check than is does to save a file.

# SCENERY

Scenery is used for creating rooms, you may create up to 255 different pieces of scenery. To create a piece of scenery, go to the Main Menu and select the SCENERY option. The screen will clear and you will be presented with the scenery menu.

| SCENERY |
|---|
| CREATE |
| VIEW |
| IMP-EXP |

This menu is very similar to the image menu except that the ANATEST option is missing from this menu. If there are no scenery drawings currently created, you will not be able to select the VIEW option.

# SCENERY / CREATE

Select the CREATE option. The screen will clear and the scenery number menu is displayed. This menu is identical to the image number menu except that the numbers now represent scenery drawings instead of image drawings.

Select a scenery drawing number. If the image you have selected is printed in red, the dimensions menu will be skipped and you will jump to the main design screen, were your drawing will be displayed. If the scenery number you have selected is already being used in a room that has already been defined the following menu will be displayed.

**WARNING:SCENERY PLACED IN ROOM I**

**LIMITED EDITING WILL APPLY**

This means that you will not be able to select the ABORT option from the design screen options menu. There may also be some restrictions when you try to alter the scenery drawings dimensions.

If the scenery number you selected had not been previously defined, the next menu to be displayed is the SCENERY TYPE menu.

| SCENERY TYPE |
|---|
| FOREGROUND |
| CENTREGROUND |
| BACKGROUND |

This menu will allow you to select the type of scenery drawing that you are going to define. Please refer back to the sections of the manual that deal with scenery. After you have selected the scenery type, and confirmed your selection, the screen will clear and you will be presented with the X and Y dimensions menu.

You will note that there are a smaller number of options to select on the X dimension menu. We have limited the number of been X dimensions used in order to save a large amount of memory. The Y dimensions are used as normal.

After you have confirmed the size of the scenery you are going to define, the screen will clear and you will be presented with the main design screen. This screen is very similar to the image design screen, and I will only detail the changes that have been made.

You may only select the MASK colour from the colour palette if you have selected FOREGROUND from the SCENERY TYPE menu.

The drawing type menu on the right hand side of the screen will display either FORE,CENT or BACK.

The design screen menu has one extra option at the bottom,

**PLANE TYPE**

This option will allow you to change the scenery into a different type. After selecting this menu the scenery type menu will be re-displayed. You may now select a new scenery type for the scenery drawing that is displayed in the design grid.

If you have opted to change the scenery type, and the scenery is being used within a room, the Designer will check to ensure that the new scenery type will not clash with any other piece of scenery within the room. These checks may take a little time. If the changes are acceptable, the scenery type will change. If the changes result in a piece of scenery clashing within a room the following error message will be displayed

**CHANGE SCENERY TYPE ERROR**

**TYPE COLLIDING WITHIN ROOM**

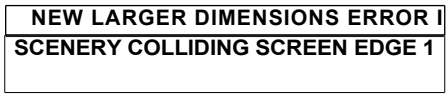
As you may or may not know, you can not place a piece of background or centreground scenery on top of each other, they can be side to side, but not on top. The only type of scenery drawings that can be placed over the top of other scenery drawing is foreground scenery. If you are trying to convert a piece of foreground scenery into background/centreground and there is already a piece of background/centreground scenery underneath then changing the foreground scenery will result in scenery drawings overlapping, which is illegal. Similarly two pieces of foreground scenery can not be placed on top of each other. If you are trying to convert a piece of centreground/background scenery to foreground and there is already a piece of foreground scenery placed on top of them, this would also result is the clashing of scenery, which is illegal.

There are no problems when you convert a piece of centreground to background or visa-versa as this will never result in a clashing of scenery.

If you try to alter the dimensions of a piece of scenery from within the design screen menu, there are also a number of checks that take place. If you are reducing the size of the scenery item, this does not cause problems. If you increase the size of a piece of scenery the Designer checks to see if the new dimensions would cause a clashing of scenery within a room. If the scenery's new dimensions cause an overlap within a room the following error message is displayed



If the scenery's new dimensions cause an overlap with the screen edge, the following error message is displayed




Pressing the SELECT key will return you to the main design screen options menu.

## SCENERY / VIEW

This option is identical to the IMAGE / VIEW option. The VIEW option will now allow you to view all of the scenery drawings you have created.

## *SCENERY /* I - O

This option will allow you to load into the Designer, scenery that has been created on another art package. You may also opt to print to a screen file, any scenery you have created within the Designer.

The operation of the SCENERY 1.0 menus is identical to the IMAGE 1.0 menus except for the following points.

### *I-O* / IMPORT

1) You will be asked to specify the graphic type every time you import a graphic item.
2) You will be importing graphics for use as scenery, and not images.

### *I-O* / EXPORT

1) You will be outputting scenery graphics, instead of images.

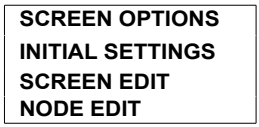
Please remember that any mask information that has been defined within the foreground drawings can not be exported. Similarly mask information can not be imported into the Designer program.

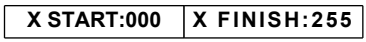
# ROOM DESIGN

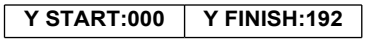
This is probably one of the most complex areas within the Designer to use, we have tried to make it's use as simple as possible.

The room Designer will allow you to create up to 255 different Rooms. To create a Room you should have first designed all of the scenery that will be placed into the room ( see Creating Scenery Option). Once you have designed a room you will be able to place Nodes within the room to control how the various sprites will move around.

To select the room Designer option, go to the Main Menu and then select the ROOM DESIGN option. You are now in the room Designer section of the Designer program. You will now see displayed on the screen a number of menus





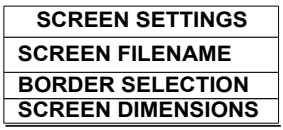
The other two menus below these menus inform the user of how much memory has already been used by the room Designer and how much memory is still free.

The information in the bottom three menus is informing you of the size of the current main screen playing area. The example above shows the playing area to be the full Sam mode 4 screen (256x192). You can alter the size of the screen through the INITIAL SETTINGS option.

## *SCREEN OPTION* / INITIAL SETTINGS

When you first enter the room Designer you may want to alter some of the initial settings, to do this select the INITIAL SETTINGS option.

The screen will now clear and the INITIAL SETTINGS menu will be displayed




If you do not require the sprites to move around the full screen, the SCREEN SETTINGS menu will allow you to reduce the main playing area, set the border colour for the area between the computers border and the game playing area or to specify a filename that will load a screen file into be used as a border. ie a Panel.

If you wish the sprites to move around the full size of the screen then you should not use this menu.

## *SCREEN SETTINGS* SCREEN DIMENSIONS

Firstly, lets alter the size of the main sprite window. Select the SCREEN DIMENSIONS option. After a delay the SCREEN DIMENSIONS menu will be displayed. This menu will allow you to change the shape of the main sprite window, for all sprites.

| SCREEN SETUP | |
|---|---|
| X START : | 000 |
| X FINISH : | 255 |
| X WIDTH : | 256 |
| Y START : | 000 |
| Y FINISH : | 191 |
| Y WIDTH : | 192 |

This is an example of a new screen, the screen is automatically set to full size. You can select one of four options from with this menu, X START, X FINISH, Y START or Y FINISH.

Selecting of one these four options will print the coordinate selection menu. This menu has the numbers from 0 to 255. Some of the number within this menu will be shown in red all of the other numbers will be displayed in blue. You may select any of the numbers in red and these numbers represent the pixel coordinates on a screen.

### *SCREEN SETUP* / X START

Selecting this option will allow you to define where the left hand side of the main sprite window starts. The left hand side of the screen has to start at an even number and so only the even numbers are displayed in red. The screen has to be at least 32 pixels across and so the red numbering ends 32 pixels away from the X finish coordinate. Please note that if you have already placed some scenery into a room, you will not be able to move the border over the scenery.

### *SCREEN SETUP* / X FINISH

Selecting this option will allow you to define where the right hand side of the main sprite window ends. The right hand side of the screen has to end on an odd number and so only the odd numbers are displayed in red. The screen has to be at least 32 pixels across and so the red numbering starts 32 pixels away from the X start coordinate.

### *SCREEN SETUP* / Y START

Selecting this option will allow you to define where the bottom edge of the main sprite window starts. There are no odd/even restrictions with the Y coordinate. The screen has to be at least 32 pixels high and so the red numbering ends 32 pixels below the Y finish coordinate.

### *SCREEN SETUP* / Y FINISH

Selecting this option will allow you to define where the top edge of the main sprite window ends. There are no odd/even restrictions with the Y coordinate. The screen has to be at least 32 pixels high and so the red numbers starts 32 pixels above the Y start

coordinate.

Once you have selected the size of your screen press the ESC key to return to the SCREEN SETTINGS menu.

## *SCREEN SETTINGS I* BORDER SELECTION

If you have reduced the size of the main sprite window, there will now be a gap between the computers border and the main sprite window. The SCREEN SETTINGS menu will allow you to fill that gap, with either a solid colour, or by loading a screen file into memory to act as a panel. The screen file could take the form of some sort of panel which might contain the score, no if lives left etc.

To select a solid colour, select the BORDER SELECTION menu. The screen will clear and your defined palette of colours will be displayed along the top of the screen. To select one of the colours, move the pointer to the desired colour and press the SELECT key. The computers border colour will change to match the colour you have selected. If you are happy with the colour press the ESC key. This border colour will now be displayed around the border of the main sprite window whilst you are designing a room.

## *SCREEN SETTINGS I* SCREEN FILENAME

To select a screen file to be loaded into the area around sprite window, select the SCREEN FILENAME menu option. You will now see the following menu.

| SCREEN FILENAME |
|---|
| CHOOSE FILENAME |
| CANCEL AND CLEAR |

This menu will allow you to either clear an existing screen border filename from memory or to select a new filename.

### *SCREEN FILENAME* / CHOOSE FILENAME

To select a filename and use it to fill the section of screen outside of the main sprite playing area select the CHOOSE FILENAME option, you will then be prompted to insert the data disc and press the SELECT button. After you have selected the filename, the screen will be loaded and you will return to the main screen Designer menu. You will note that the screen that you loaded into memory is displayed behind the menu, this is perfectly normal.

Whenever you select the ROOM DESIGN option from the Main Menu, the screen filename you have selected will be displayed. If you have sufficient memory free (over 24576 bytes) the screen file will be stored in memory and you will not need to load it into the computer again. If however you have less than 24576 bytes of memory free, whenever you select the ROOM DESIGN option you will be prompted to insert the disc containing the screen file that will be used as the border for the main sprite window.

It is suggested if you have less the 24576 bytes of memory free you do NOT use the screen filename option, as it can be a pain to keep loading the screen in every time you enter the room Designer.

***SCREEN FILENAME*** / CANCEL AND CLEAR

To clear an existing border filename from memory select the CANCEL AND CLEAR menu option. Any filename which you were using for a screen border will now be cleared from memory (but not from the disc).

# *SCREEN OPTION* / SCREEN EDIT

This option will allow you to design a room. To create a room, select SCREEN EDIT from the SCREEN OPTIONS menu. You will now see the following menu displayed.

| SCREEN |
|---|
| EDIT |
| VIEW |

This menu will give you the option of either view a screen that has already been defined or creating/editing a screen.

## *SCREEN* / EDIT

Select the EDIT option. The screen will clear and you will see the ROOM NUMBER menu displayed. This menu represents all of the screens that can be created using the screen Designer. Numbers that are printed in Red represent rooms that have already been defined, blue numbers represent rooms that have not been defined. Selecting a red number (defined room) will display the room and print the ROOM EDIT menu. Selecting a blue number (undefined room) will display a blank screen and then print the ROOM EDIT menu.

| SCENERY |
|---|
| PLACE |
| MOVE |
| ERASE |
| HIGHLIGHT |
| VIEW |
| DESTROY |

Below this menu are a number of menus giving you information about the current room. The menus tell you the number of foreground, centreground and background scenery drawings that are currently placed in the room. The total number of drawings in the room and the total amount of memory free. If you have not placed any scenery into the room the only option you may select at this stage, is the PLACE option.

***SCENERY***/PLACE

This option will allow you to place scenery into the room, after selecting this option the following menu is displayed

| PLACE |
|---|
| FORE |
| CENT |
| BACK |

You may now choose what type of scenery you wish to place into the room. If you need

to know the difference between the different type of scenery, please refer back to the scenery introduction at the start of this manual. If you can not select one or more of the scenery types from the menu, then that scenery type has not been defined.

Select a type of scenery and the SCENERY NUMBER menu will be displayed. This menu has all of the scenery drawing numbers between 0 and 255. The numbers may be one of three colours. Red numbers represent the scenery drawings of the type you have selected. If you selected FORE in the last menu, all of the red numbers represent foreground scenery. The dark blue numbers represent scenery drawings that have been defined but are not of the correct type. The light blue numbers represent scenery drawings that have not been defined.

You may either select one of the red scenery numbers, or press the ESC key to return to the PLACE menu. If you have selected a red number, you will return to the room design screen, and the scenery item you selected will be placed in the top left hand corner of the screen. You may move the piece of scenery around the screen using either the mouse or the arrow keys.

You will note that the there is a coordinate display showing the exact position of the piece of scenery you are moving around the screen. This coordinate display is to help you position your scenery in exactly the right position.

To place a piece of scenery, press the SELECT key. You may now move the scenery again and place another copy of it in another part of the screen. There are a couple of rules that denote how scenery can be placed into a room. 1) You can not place a piece of background or centreground scenery over the top of another piece of background/centreground scenery. 2) You can not place a piece of foreground scenery over the top of another piece of foreground scenery.

When you have finished placing that particular piece of scenery, press the ESC key, and the following menu will be displayed.


SELECT NEXT DRAWING UP

SELECT NEXT DRAWING DOWN

USE CURRENT DRAWING

NO OF CENTREGROUND ON SCREEN

TOTAL SCENERY ON SCREEN

CURRENT SCENERY NUMBER

CURRENT SCENERY DRAWING

This menu will allow you to select the next piece of scenery of the same type you originally selected. Selecting the UP/DOWN options will display the next higher/lower scenery drawing. The menu will also tell you how many pieces of that particular type of scenery (fore,cent,back) have been placed into this room, and how may pieces of

scenery in total have been placed into the room.

To select the piece of scenery that is displayed at the bottom of the menu, select the USE option.

If you do not require to place any more pieces of scenery of that particular type, press the ESC key and you will be returned to the main room design menu. You may select PLACE again and then place a different type of scenery.

### SCENERY / MOVE

This option will allow you to move the scenery that has already been placed into the room. Select the MOVE option. You may now select which type of scenery you wish to move. You may move either foreground scenery or centreground/background. Note that you have only two options, the centreground and background scenery counts as one option.


After selecting the type of scenery to move, you will see a display of the current room. The pointer has changed it's shape to an **'M'** character. This is the move pointer. To move a piece of scenery, place the pointer over the piece of scenery you wish to move and press the SELECT key. The pointer will now disappear and you will now be able to move the piece of scenery you selected around the screen. You will notice that the coordinates of the piece of scenery you are moving around the screen are displayed. To place the piece of scenery you are currently moving press the SELECT key.

Once you have replaced a piece of scenery, the move pointer will return and allow you to select another piece of scenery to move. If you do not wish to move any move pieces of scenery press the ESC key, and you will be returned to the room Designer options menu.

If you press the ESC key whilst you are moving a piece of scenery around the screen, the scenery will be removed from the room.

If you find that you can not pick up a piece of scenery, you have probably selected the wrong scenery type.

### *SCENERY/ERASE*

This option will allow you to remove pieces of scenery from the room. Select the ERASE option. You may now select which type of scenery you wish to erase. You may erase either foreground scenery or centreground/background. Note that you have only two options, the centreground and background scenery counts as one option.

| ERASE |
|---|
| FORE |
| CENT |
| BACK |

After selecting the type of scenery to erase, you will see a display of the current room. The pointer has changed it's shape to an **'E'** character. This is the erase pointer. To erase a piece of scenery, place the pointer over the piece of scenery you wish to erase and press the SELECT key. The piece of scenery underneath the pointer will now be removed from the room. You may now move onto other pieces of scenery and remove them. To return to the room design options menu, press the ESC key.

Note. Erasing scenery from a room does not erase the scenery from the Designers memory, you may use the piece of scenery you erased from the room again.

### *SCENERY/* HIGHLIGHT

This option will allow you to hi-light, within a room, all of a specific type a scenery. This option is useful for viewing where you have placed your centreground scenery. Select the HIGHLIGHT option. You will now be prompted to select from the following menu

| HIGHLIGHT |
|---|
| FORE |
| CENT |
| BACK |

This option will allow you to select any of the three types of scenery, the CENT and BACK option are not linked on this menu.

After you have made your selection, the current room will be displayed on the screen, and the type of scenery you selected will be highlighted. Pressing the SELECT key will toggle between the scenery being highlighted and the scenery displaying normally. To return to the options menu, press the ESC key.

### *SCENERY /* VIEW

This option will allow you to see your the room you have designed with all of the menus removed. Select the VIEW option, the palette will change to your own defined colours and all of the menus will be removed. To return to the options menu, press either the SELECT or the ESC keys.

### *SCENERY/* DESTROY

This option will completely clear from memory the currently displayed room. The scenery drawings that were used to create the room will not be erased. Select the DESTROY option. You will now be asked to confirm you wish to destroy the room. Selecting YES will erase the room and return you to the room number menu. Selecting NO will cancel the destroy instruction and allow you to select an option from the options menu.

To save the room you are working on, press the ESC key whilst you are in the room design option menu. Pressing ESC will return you to the room number options menu.

# *ROOM DESIGN: HINTS*

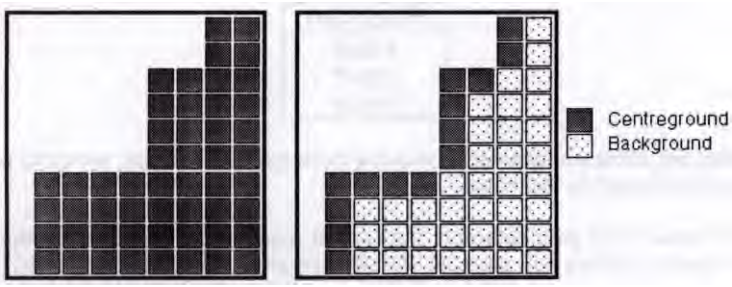
There are a number of ways of making the Supervisor work faster when you are designing your rooms. By following the tips given below, you will find that your games will run faster.

1) DO NOT USE LARGE AMOUNTS OF FOREGROUND DRAWINGS.

The more foreground drawings you use, the slower your sprites will move around the screen. This is because every time the Supervisor moves a sprite it has to reprint all of the foreground drawings back onto the screen. If you have 25 foreground drawings placed in a room, every time the Supervisor moves a sprite it has to reprint 25 foreground drawings.

Foreground drawings have been included within the room Designer for use as occasional special effects. After using the program for a while you will learn how many foreground sprites you can use for a particular room. If you are only going to be moving a small number of sprites around inside the room, then you can get away with using more foreground drawings.

2) LIMIT THE AMOUNT OF CENTREGROUND SCENERY TO WHATEVER IS NECESSARY.




Every time the Supervisor moves a sprite it checks the list of centreground coordinates to see if the sprite is going to hit the centreground (a sprite can not move onto centreground scenery). So therefore the more centreground scenery you have placed within the room, the longer the list will be. The longer the list, the longer the Supervisor takes to check the list.

The two diagrams above show you a section of a room. The section of room on the left is made up completely of centreground scenery, this would create a long list of coordinates for the Supervisor to check. You do not need this amount of centreground scenery within the room. The diagram on the right has a great deal less centreground scenery, and it will still achieve the same effect, the sprite would not be able to move into shaded area in either diagram. However the diagram on the right would enable the spites to move faster, simply because there is less centreground scenery to check.

You can of course design 2 pieces of identical scenery, 1 piece could be centreground and another identical piece could be background. The rooms would therefore look identical.

You should try to limit the amount of centreground scenery to a bare minimum, this will enable your sprites to move that little bit faster.

### *SCREEN* / VIEW

To view the rooms that have been created in the room Designer, select the ROOM DESIGN option from the Main Menu, then select SCREEN EDIT, and finally select VIEW. If you are already using the room Designer then you will not need to go to the Main Menu. If is very difficult when writing a manual to know where you will be in the menus, so I am telling you the route to options from the Main Menu. You will soon learn to navigate around the menus without any descriptions.

After you have selected the VIEW option, you will see displayed the ROOM NUMBER menu. Any rooms that have already been defined will be displayed in red. You may select any of the red numbers. After you have selected a number, the screen will clear and the room will be displayed. Pressing either the SELECT key or the ESC key will take you back to the ROOM NUMBER menu.

Whilst a room is being displayed, you may display the next room in the sequence by pressing the SYMBOL key and then either **'U'** to see the next room up, or **'D'** to see the next room down.

## *SCREEN OPTION* / NODE EDIT

As I have already mentioned in the introductory sections of the manual, you may create nodes within your rooms to instruct the sprites to perform certain tasks. To enter the node editor from the Main Menu, select ROOM DESIGN, and then select NODE EDIT.

You are now in the node editor section of the room Designer. You may only add nodes to rooms that have already been defined. You should now see on the screen the ROOM NUMBER menu. All of the rooms that have been defined are displayed in red. Select a room number. The screen will clear and the room will be displayed.

There should now be three menus displayed on the screen.

| NODES |
|---|
| PLACE |
| MOVE |
| ERASE |
| ALTER |
| VIEW |

This is the main node editor menu. Below this menu will be another two menus, the first menu will display the number of nodes that are currently defined in the room, and the second menu is informing you of how much memory you have free.

If there are no nodes already on the screen, the only option you may select from the main node editing menu is the PLACE option.

### *NODES I* PLACE

After selecting this option, the menu will be removed and the room will be displayed. You will notice that the colours within the room have all been darkened, although you

should still be able to tell where all the scenery is placed. The colours have been darkened to that the node points will stand out.

The pointer can now be moved around the screen. You will notice that the coordinates of the pointer position are displayed as you move the pointer. This will enable you to place the node in exactly the right position.

You should have carefully planned how the room will look, and you should know the exact position of where the nodes will be placed. You can not move the pointer outside of the main sprite window. If you wish to place a node in one of the corners, you may rotate the pointer using the SYMBOL-R key combination.

When placing a node it must be remembered that a sprite will only be classed as hitting the node when the sprites **BOTTOM LEFT** hand corner has the same coordinates as the node.

When you have positioned the pointer at the correct position on the screen, press the SELECT key. A white pixel will mark the point where the node is placed and the node setup menu will be displayed.

| | NODE SETUP |
|---|---|
| | X:210 Y:115 |
| 1 | NO DATA DEFINED |
| 2 | NO DATA DEFINED |
| 3 | NO DATA DEFINED |
| 4 | NO DATA DEFINED |
| 5 | NO DATA DEFINED |
| 6 | NO DATA DEFINED |
| 7 | NO DATA DEFINED |
| 8 | NO DATA DEFINED |
| | ERASE THIS NODE |

The menu displayed on the left is a typical node setup menu. I have had to substitute the arrow keys with numbers as I cannot find any arrow characters on this word processor. The numbers from 1 to 8 refer to the eight different directions. You may now select one of the directions and allocate a command to it.

In order to get an insight into how nodes work, please refer to the *Introducing Nodes* section of the manual.

Going back to the NODE SETUP menu, you should now choose one of the directions.

If you select the ERASE THIS NODE option, the following menu will be printed

| CONFIRM:ERASE THIS NODE |
|---|
| YES |
| NO |

If you select the NO option, the confirm menu will be removed and you will be allowed to choose another option from the NODE SETUP menu.

If you select the YES option, the node will be erased and toy will be returned to the main node options menu.

After you have selected one of the directions from the NODE SETUP menu, the menu will clear and the DEFINE NODE menu will be displayed.

| DEFINE NODE |
| --- |
| CLEAR DATA |
| JUMP |
| FIRE |
| SPEED |
| REMOVE |
| PLACE |

This menu will allow you to assign a command to the direction you selected in the NODE SETUP menu. To return to the NODE SETUP menu press the ESC key.

### *DEFINE* NODE /CLEAR DATA

Selecting this option will allow you to clear the data for the direction you selected within the NODE SETUP menu. After selecting the CLEAR DATA option you will be asked to verify your choice.

### *DEFINE NODE* / JUMP

This option will program the sprite to jump. After you have selected this JUMP option, the jump definition menu will be displayed. This menu will allow you to select the type of jump you require. The JUMP DEFINITION menu is displayed listing all of the possible jump types. Selecting one of the numbers from the JUMP DEFINITION menu will then return you to the NODE SETUP menu. The jump type you selected will now be displayed alongside the direction you selected.

It should be noted that the JUMP DEFINITION menu will list all of the jumps that may be used, some of these jumps may not have been defined. It is up to you to ensure that only defined jumps are used by the nodes. To define a jump, please read the relevant section of the manual.

The sprite can jump in one of three directions, left, straight up, and right. The sprite will jump in the direction you selected in the NODE SETUP menu.

### *DEFINE NODE* / FIRE

Selecting this option will program the sprite to fire a missile. There are no sub-menus for this option. After selecting the FIRE option, you will be returned to the NODE SETUP menu. The direction you selected from this menu will now have the message FIRE MISSILE.

### *DEFINE NODE* / SPEED

This option will program the sprite to leave the node in the direction you selected in the NODE SETUP menu. After selecting the SPEED option you will be presented with an X speed menu, a Y speed menu or both. You may now select the speed with which you wish the sprite to leave the node. After making your selection, you will be returned to the NODE SETUP menu. The speeds you selected will now be displayed within this menu.

### *DEFINE NODE* / REMOVE

This option will program the sprite to remove itself from the screen. After selecting the REMOVE option, you will be returned to the NODE SETUP menu.

***DEFINE NODE*** / **PLACE**

This option will program the sprite to be removed from the screen, and to be replaced at another point on the screen. Select the PLACE option. the DEFINE NODE menu will be removed and the following message will be displayed.

**SELECT POINT TO PLACE SPRITE**

You should now press the SELECT key. This message will disappear and the coordinates of the pointer will be displayed. You may now move the pointer to the position on the screen where you wish the sprite to be transferred to, when you are sure you are on the right position, press the SELECT key. You will now return to the NODE SETUP menu.

When positioning the pointer over the new sprites position, you should note that the sprites bottom left hand corner will be placed in the new position. You should make a note at which position you are replacing the sprite, and then place a node at the point the sprite will appear. This new node, at the sprites replace position, should be programmed for the directions the sprite should leave the new position.

It must be remembered that when you are positioning the pointer for the location the sprite will be transferred to, you must ensure there is enough room for the sprite to fit onto the screen. If the sprite is moved to the new position and the sprite is overlapping the screen edge, an error message will be displayed when using the Supervisor. Remember the sprites bottom left hand corner will be placed at the new position.

***NODES*** / **MOVE**

This option will allow you to move a node that has already been placed onto the screen to a different part of the screen. You will only be able to select this option if there is at least 1 node already placed on the screen.

Select the MOVE option. The menus will be removed and you will have a clear view of the room. You will notice the pointer has changed shape and now contains the character 'M'. This is the move pointer. All of the nodes that have been defined will be displayed as bright white dots. To move a node, place the pointer tip over the node and press the SELECT key. The pointer tip has to be exactly over the node point. Once you have selected a node you may move the node around the screen. Whilst you are *carrying* the node, the border will flash. This is simply a reminder that you are carrying a node.

To place the node in a new position on the screen, press the SELECT button again. The node will be placed back onto the screen. If you wish to cancel the node move operation whilst you are carrying the node, press the ESC key. This will return the node back to it's original position.

***NODES*** / **ERASE**

This option will allow you to erase nodes from the screen. You may only select this option when there is at least one node already placed on the screen.

Select the ERASE option. The menus will be removed and you will have a clear view of the room. You will notice the pointer has changed shape and now contains the character **'E'.** This is the erase pointer. All of the nodes that have been defined will be displayed as bright white dots. To erase a node, place the pointer over the node and press the SELECT key. The node will now be erased. To return to the node options menu, press the ESC key.

### *NODES /* **ALTER**

This option will allow you to alter the settings for a particular node.You may only select this option when there is at least one node already placed on the screen.

Select the ALTER option. The menus will be removed and you will have a clear view of the room. The coordinates menu will be printed on the bottom of the screen. To alter a particular node, move the pointer so that the tip of the pointer is placed over the node you wish to alter and press the SELECT key. If you miss the node the message

**NODE IS NOT DEFINED**

will flash up onto the screen. The message will disappear after a couple of seconds.

Once you have selected a node, the NODE SETUP menu will be displayed and you may edit the node as described in the previous pages. Once you have completed altering the node, press the ESC key and you will be returned to the main node options menu.

### *NODES* /**VIEW**

This option will allow you to view the nodes within the room, you will be able to see exactly how the nodes will react with each other.

Select the VIEW option. The VIEW menu will now be displayed.

| VIEW |
|---|
| JUMP |
| FIRE |
| SPEED |
| REMOVE |
| PLACE |

### *VIEW/* **JUMP**

Selecting the JUMP option will allow you to see all of the nodes that have a jump definition defined, and the paths the sprites will take when they jump from a node.

The paths the sprites will take will be plotted onto the screen in a bright green colour. After all of the paths have been plotted, the pointer will be printed onto the screen, along with the coordinated position. The pointer is placed onto the screen to aid you in determining where you may wish to place other nodes.

Pressing the SELECT key, will return you to the main nodes options menu.

### *VIEW/* **FIRE**

Selecting the FIRE option will allow you to view all of the nodes that have a fire command defined. All of the nodes which contain a fire command will start to flash. The

pointer will be placed onto the screen along with the coordinate position.

### *VIEW* / SPEED

Selecting the SPEED option will allow you to view all of the nodes that have speed commands defined, and the paths the sprites will take when they move from a node.

The paths the sprites will take will be plotted onto the screen in a bright green colour. After all of the paths have been plotted, the pointer and the pointers coordinates will be displayed.

### *VIEW* / REMOVE

Selecting the REMOVE option will allow you to view all of the nodes that have a remove command defined. All of the nodes which contain a remove command will start to flash. The pointer will be placed onto the screen along with the coordinate position.

### *VIEW* / PLACE

Selecting the PLACE option will allow you to view all of the nodes that have a place command defined.

Any node that has a place command defined will display a line to the opposition the sprite will be placed. After all of the paths have been displayed, the pointer and the pointers coordinates will be displayed.

## NOTES ON VIEWING NODES

When you select any of the VIEW options, you will notice that the coordinates menu is displayed. As you move the pointer around the screen, the coordinates menu is updated, informing you of the current position of the pointer.

When you are viewing the sprites speeds, you will notice that the sprite paths are highlighted by a series of dots. These dots represent the position of the sprites bottom left hand corner will occupy.

When positioning nodes that will intercept other nodes, this display is important. You should place the pointer over the dot to find its exact coordinates, you will then be able to place the node at the correct position so that the sprite will collide correctly with the new node.

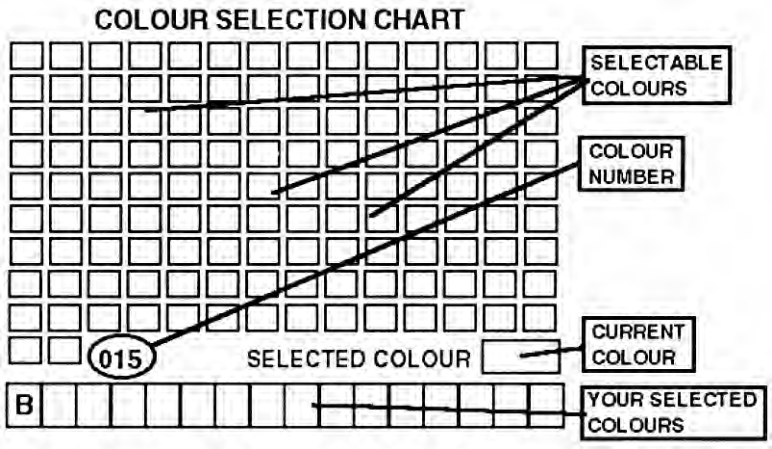
It is important when you are creating node paths that the sprites hit the correct nodes, using this method of creating paths should eliminate any problems of sprites missing the nodes.

# COLOURS

The colour selection screen will allow you to select the 16 colours which will be displayed throughout your game. The colour selection will display all 128 of the Coupe colours on the screen at the same time. From these 128 colours you will pick the 16 most suitable for your game.

To enter the colour selection screen, go to the Main Menu and select the COLOURS option. The Main Menu will clear and the colour selection screen will appear.

You will now see 10 rows of colours, with each row containing 14 different colours. If you move the pointer around these coloured blocks you will notice that the number near the bottom left hand side of the screen is changing. This number is the colour code for the colour the pointer is currently pointing at.




Running along the bottom of the screen are the 16 paintpots which contain the colours you will use to design your drawings. You will not be able to see the first colour as it is the background colour. On the screen there will be a 'B' displayed.

To change a colour is very simple. Move the pointer over one of the 128 different colours displayed at the top of the screen. Press the SELECT button. You will notice that the colour you selected is transferred to the CURRENT COLOUR block. You may now move the pointer to one of the paintpots at the bottom of the screen press the SELECT key again. The colour that was displayed in the CURRENT COLOUR block will now be transferred to the paintpot you have just selected. Once you have selected all of the colours you wish to change press the ESC key, this will then return you to the Main Menu.

One very important point to remember when you are designing your colours is that all of the menu items on the design screen are displayed in colour 1. In order to clearly see these menus you must ensure that any text written in colour 1 is clearly visible against the background colour.

Due to the complex colour arrangements of some of the menus, your defined colours will not be displayed all of the time. Your defined colours are switched in whenever it is necessary to do so.

Even though the Sam Coupe is capable of displaying all 128 colours on the screen at the same time, this process is very complex and therefore has not been incorporated into the Supervisor program. You are limited to the 16 colours you select at the bottom of the screen. There are occasions when these colours may be altered outside of this colour selection screen. If you import graphics, you will be given the option of using the imported colours. You must be careful, and always make a note of the colours you are using in your program, just in case there is an accident !!

# FILE

This option will allow you to access the disc routines, to save and load your main drawing data.

On selecting this menu, the following menu will be displayed.

| DISC |
|---|
| LOAD |
| SAVE |

## *DISC/ LOAD*

Select the LOAD option. When you have selected the LOAD option a warning message will be displayed

**DATA IN MEMORY WILL BE LOST**

This message is warning you that if you load another data file into memory, the data file that may already exist in memory will be erased. If you do have another file in memory, and you have not saved that file, you should press the ESC key and save your existing file.

If there are no data files in memory or if the file in memory is saved then you should proceed by pressing the SELECT key. After pressing the select key the warning message will disappear.

**INSERT DATA DISC**

is now displayed on the screen. If you change your mind about loading a data file into memory, press the ESC key and you will go back one menu. You should now place the disc containing drawing data into drive 1. After inserting the data disc, press the SELECT button. The disc drive will now start.

If there is a problem with the disc or the disc drive, the screen will clear and a disc error message will be displayed. Take a note of the disc problem and try to rectify the error. If you now press the SELECT or ESC keys, the error message will disappear and the ' insert data disc' menu will appear.

If the Designer program can not find any drawing data files on the disc in the drive, the message

**NO RELEVANT FILES**

will be displayed. If you now press the SELECT or the ESC keys the message will disappear and the 'INSERT DATA DISC' message will be displayed. You may now try again.

Once the Designer program has found drawing data files, two menus will be displayed. The left hand menu has a list of all of the drawing data files that were found on the data disc. The right hand menu contains two options.

The first option speaks for itself. TRY ANOTHER DISC, once selected will do just what it says. It will allow you to insert another disc and get another directory. Pressing the SELECT button on this option will return you to the 'insert data disc' message.

The second option SELECT A FILE will allow you to select a file from the directory menu. Press the SELECT key on SELECT A FILE, the right hand menu will then disappear and you may select an option from the directory menu. If you select the SELECT A FILE option and the menu has disappeared leaving only the directory menu and you change your mind about loading any of the files in the directory menu, press the ESC key and the LOAD DISC OPTIONS menu will re-appear on the right hand side of the screen.

Once you have selected one of the files from the menu directory, the menu will disappear and the data file will be loaded into memory. Once the file has been completely transferred to memory the Main Menu will be displayed.

If there is a problem when the file is loading into memory, an error message will be displayed as described above. Pressing the select key after the error message has been displayed takes you to the INSERT DATA DISC menu again. If you own a 256k Coupe and you try to load a file which is larger than the memory will allow, an error message will be displayed.

## *DISC / SAVE*

Select the SAVE option. After selecting the save disc option, the message

| **INSERT DATA DISC** |
|---|

will be displayed. You should now insert the disc you wish to save your drawing data on, into disc drive 1 and press the SELECT key. If you do not wish to save your drawing data, press the ESC key. After pressing the SELECT key, the drive will start.

If there is a problem with the disc or the disc drive, the screen will clear and a disc error message will be displayed. Take a note of the disc problem and try to rectify the error. **If** you now press the SELECT or ESC keys, the error message will disappear and the ' insert data disc' menu will appear.

After a short time, two menus will be printed onto the screen. The menu on the left will display a list of all of the current data files that are already stored on the disc. If there are no drawing data files stored on the disc, the message

| **NO RELEVANT FILES** |
|---|

will be displayed. The menu to the right of the screen will be displaying four options.

| **SAVE DISC OPTIONS** |
|---|
| **TRY ANOTHER DISC** |
| **SELECT A FILE** |
| **ENTER FILENAME** |
| **SAVE AS:FILE0** |

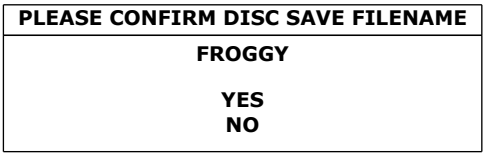
The first option TRY ANOTHER DISC will allow you to insert another data disc by taking you back to the INSERT DATA DISC menu.

The second option SELECT A FILE can only be highlighted if a directory has been printed on the left hand side of the screen. If you select this option, the menu will disappear and you may then select a filename from the list on the left hand side of the screen. The drawing data that is currently in memory will then overwrite the data in the filename you have selected. You should use this option with care, you could lose a lot of valuable work by being careless. The filename is verified before it is overwritten and this should eliminate any mistakes. Press the ESC key to abort any mistake.

The third option ENTER FILENAME will allow you to create a new name for the data in memory. Selecting this option will bring up the alphabet menu. You may now select the name of the data you wish to save.

The fourth option SAVE AS:FILEO will allow you to save your data with the same name as it was when you loaded it. If you had loaded a data file called PACMAN then when you come to use this fourth option the menu will display SAVE AS:PACMAN. If you have designed all of your data from scratch this option will display SAVE AS:DRAW1.

No matter which method you use to select a name for your data file, the file will not be saved straight away. A menu will be displayed towards the top of the screen.




If you select the YES option, the file will be saved to disc. If you select the NO option, you will be returned to the previous menu. Do not be alarmed at the length of time the data takes to save. This is perfectly normal. The data is very carefully verified after it has been saved to ensure there are no errors.

# ANIMATION

The animation section of the Designer will allow you to define up to 64 different animation sequences. Please see the discussion on animation towards the beginning of this manual

Upon selecting the ANIMATION option, the screen will clear and you will be presented with the ANIMATION SEQUENCE menu. This menu has the numbers from 1 to 64. These numbers represent the 64 animation sequences. If an animation sequence has already been defined, the animation number will be displayed in green. All grey numbers represent animation sequences that have not been defined.

Select one of the animation numbers. You will now be presented with the SEQUENCE MENU.

| SEQUENCE NUMBER : 28 | | |
|---|---|---|
| 1 | UP | NOT DEFINED |
| 2 | UP - RIGHT | NOT DEFINED |
| 3 | RIGHT | NOT DEFINED |
| 4 | DOWN - RIGHT | NOT DEFINED |
| 5 | DOWN | NOT DEFINED |
| 6 | DOWN - LEFT | NOT DEFINED |
| 7 | LEFT | NOT DEFINED |
| 8 | UP - LEFT | NOT DEFINED |
| 9 | STATIONARY | NOT DEFINED |
| TEST | SAVE | ABORT |

You may now move the pointer and select one of the animation directions (1-9). If there are no animation ranges defined, the only option you may select along the bottom of the menu, is the ABORT option. If there is at least one animation range defined, you may select any of the options.

## *SEQUENCE NO* / DIRECTION

Selecting one of the directions (1-9) will then display the IMAGE NUMBER menu. This menu will allow you to select an animation range. The red numbers signify images that have been defined, blue numbers represent undefined images.

Select the image number for the start of the animation range. You will see the selected number displayed at the top of the screen. Now select the image number for the end of the animation range, the finish image number will also be displayed at the top of the screen.

You will now be asked to verify the range you have selected. If you have made a mistake, select the NO option, and you will be allowed to select another range.

Once you have confirmed the range, you will be returned to the sequence number menu. The range you selected will now be displayed alongside the direction you selected.

If you wish to change a range that has already been defined, select the direction of the

range you wish to alter, the following menu will then be displayed.

| RANGE |
|---|
| CLEAR |
| ALTER |

### *RANGE* / CLEAR

This option will erase the selected animation range, from the animation sequence.

### *RANGE I* ALTER

This option will allow you to alter the currently defined range, after selecting this option, specify the new range as if you were defining a new range.

## *SEQUENCE NO I* TEST

This option will allow you to test the animation sequence you have defined. After selecting the TEST option. The screen will clear and the SPEED menu will be displayed.

| SPEED |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |

This menu will allow you to specify the number of pixels the image will move once it is placed onto the screen.

Moving 1 pixel at a time will cause the image to move very smoothly

After selecting the speed the image will move onto the screen, the DELAY menu will be displayed.

This menu will allow you to set the time delay between updating the images on the screen. Option 1 will result in the image having a very short delay time, and so the sprite will move at a very quick speed. Option 9 will result in the sprite moving at a very sedate place.

| DELAY |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |

After you have made your delay selection, the screen your image will be displayed on the screen. By moving the mouse or the cursor arrow keys, the image will move on the screen. If there is an animation direction defined for the direction the image is moving, the image will be animated.

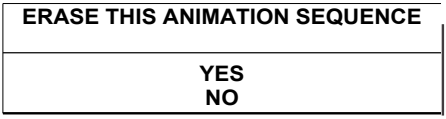
To return to the SEQUENCE NUMBER menu, press the ESC key.

## *SEQUENCE NO* / **SAVE**

This option will save the current animation sequence into memory. After selecting the SAVE option, you will be returned to the ANIMATION SEQUENCE selection menu.

## *SEQUENCE NO* / **ABORT**

This option will clear the currently defined animation sequence from memory. After selecting the ABORT option, the following menu will be displayed.




If you select the YES option, the animation sequence will be erased from memory, and you will be returned to the ANIMATION SEQUENCE selection menu.

If you select the NO option, the animation sequence will not be erased, and the command will be cancelled.

It is important that you understand the rules for animation. The rules are explained in detail in the *introducing animation* section of the manual.

# JUMP OFFSETS

The JUMP OFFSETS section of the Designer, will allow you to define up to 8 different jump sequences. A jump sequence will allow the sprites to jump under control of the Supervisor.

You should always define a jump, going to the right. When the sprite is instructed to jump, the Supervisor will look at the direction the sprite is jumping and alter the jump to suit the direction.

If the sprite was moving to the right, a normal jump to the right would take place.

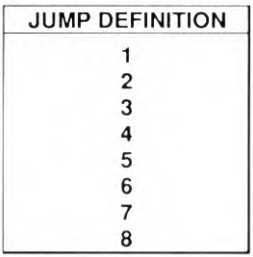
If the sprite was jumping to the left, all of the jump X speeds would be negated, therefore the sprite would jump to the left.

If the sprite was stationary, all of the jump X speeds would be zeroed, therefore the sprite would jump straight up.

After the sprite had completed the jump, the original X speeds would be restored.

Any number of sprites can use the same jump sequence.

Select the JUMP OFFSETS option from the Designers Main Menu. The screen will clear and the JUMP DEFINITION menu will be displayed.




The JUMP DEFINITION menu will allow you to select the jump sequence you wish to define. Any numbers that are highlighted in black, will already have a jump sequence defined. The numbers printed in a light grey colour have no jump definitions defined.

After you have selected a jump sequence number, the screen will clear and the jump JUMP STAGE menu will be displayed.

If the jump sequence number you selected already contains a jump sequence, the JUMP STAGE menu will contain a list of the previously defined relative jump speeds. To the right of the menu, the course the sprite will take whilst jumping will be plotted onto the screen.

At the right hand side of the screen are two menus. These two menus will tell you the maximum height of the jump, and the distance of the jump. Every time you enter a new set of jump speeds, the menus will be updated.

You may enter up to 15 different jump stages within the JUMP STAGE menu. A jump

should first start to move up the screen, and then start to move down the screen. Once you have defined the jump to start moving down the screen, you may not then program the sprite to move up the screen (it would be a strange looking jump if the sprite started to come down and then suddenly moved up again).

| STAGE | X | Y |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 2 | 3 |
| 3 | 2 | 2 |
| 4 | 2 | 2 |
| 5 | 2 | 1 |
| 6 | 2 | 1 |
| 7 | 2 | 0 |
| 8 | 2 | 0 |
| 9 | 2 | -1 |
| 10 | 2 | -1 |
| 11 | 2 | -2 |
| 12 | 2 | -2 |
| 13 | 2 | -3 |
| 14 | 2 | -3 |
| 15 | 0 | 0 |
| ENTER | | SAVE |
| ABORT | | EDIT |

The menu on the left is a typical example of how a JUMP STAGE menu would look.

This menu has 14 of the 15 stages filled in. The HEIGHT menu would be displaying 12, the DISTANCE menu would be displaying 28.

The course the sprite would take would be plotted towards the bottom of the screen on the right hand side.

You may enter additional data into the menu by selecting the options printed at the bottom of the menu

If the jump sequence you selected had not already been defined, you would only be able to select the ENTER and the ABORT options.

## *JUMP STAGE I* ENTER

You may use this option whilst there are between 0 and 14 stages entered. If you have defined all 15 stages within the JUMP STAGE menu, you will not be able to select this option.

The ENTER option will allow you to enter 1 stage within the JUMP STAGE menu. Select the ENTER option. Across the bottom of the screen the following menu will be displayed.

| X | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 |
|---|---|---|---|---|---|---|---|---|---|---|

This is the X SPEED menu. You may now select the x speed you wish the sprite to take. After making your selection, your choice will be entered into the JUMP STAGE menu. The X SPEED menu will now be replaced by the Y SPEED menu.

| Y | DW | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 |
|---|---|---|---|---|---|---|---|---|---|---|

If you have made a mistake in selecting the X speed option, you may press the ESC key to return to the X SPEED menu.

This menu will allow you to enter the Y speed of the jump. This is the POSITIVE Y SPEED menu, you may select the NEGATIVE Y SPEED by clicking the pointer on the DW option.

| Y | UP | 00 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
|---|---|---|---|---|---|---|---|---|---|---|

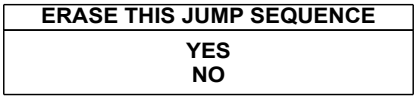
It should be noted that after you have entered a negative Y speed, selecting the UP option will result in the following error message


After you have entered a speed for the Y section of the jump, the JUMP STAGE menu will display your selection, and the next stage of the jump will be displayed. The HEIGHT and DISTANCE menus will be updated.

You may now repeat your selection and enter another stage of the jump by selecting the ENTER option again.

## *JUMP STAGE* / ABORT

This option will erase the complete jump sequence from memory. After selecting the ABORT option, the verify menu will be printed.




Selecting the YES option will erase the jump sequence from memory and return you to the JUMP DEFINITION menu.

Selecting the NO option will cancel the ERASE command and return you to the JUMP STAGE menu.

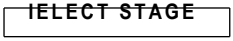
## *JUMP STAGE* / SAVE

This option will save the jump sequence that is currently displayed in the JUMP STAGE menu. You may only select this option when there is at least 1 jump stage in the jump sequence defined. After selecting the SAVE option, you will be returned to the JUMP DEFINITION menu.

## *JUMP STAGE* / EDIT

This option will allow you to alter a previously defined stage within the JUMP STAGE menu. You will only be able to select this option when there is at least 1 jump stage in the jump sequence defined.

After selecting the EDIT option, the message




will be printed. You may now move the pointer over any of the stages that have been defined within the JUMP STAGE menu and select a stage. Once you have selected a stage, a menu displaying your choice will be prompted at the top of the screen and the following options menu will be displayed.

| STAGE EDIT |
| --- |
| DELETE |
| CHANGE |

## *STAGE EDIT* /DELETE

This option will allow you to remove the stage you have selected from the jump sequence. Once the stage has been removed, all the stages below will move up one place.

After selecting the DELETE option, a verify menu will be printed

| CLEAR THIS JUMP STAGE |
| --- |
| YES |
| NO |

Selecting the YES option will remove the stage from the JUMP STAGE menu.

Selecting the NO option will cancel the ERASE command and return you to the SELECT STAGE message.

## *STAGE EDIT* /CHANGE

This option will allow you to alter the X and Y speeds for a particular jump stage. After selecting the CHANGE option, the X SPEED menu will be printed at the bottom of the screen. You may now enter the new stage in the normal way.

If you have selected a stage to edit, which is in the middle of a group of Y + coordinates, you will not be able to change the Y speed to a negative value. Attempting to change the Y speed would result in the following error message.

| ERROR:SPRITE GOING UP |
| --- |

If you have selected a stage to edit, which is in the missile of a group of Y - coordinates, you will not be able to change the Y speed to a positive value. Attempting to change the Y speed would result in the following error message.

| ERROR:SPRITE GOING DOWN |
| --- |

# KEY DEFINITION

This option, which can be selected from the Main Menu, will allow you to alter the key definitions for use in the Supervisor. Key definitions are used by the Supervisor to simulate joy-stick movements.

After selecting the KEY DEFINITION option, the following menu will be displayed.

| KEY DEFINITION |
|---|
| 1 |
| 2 |

The Supervisor will recognise up to two different sets of keys, this menu will allow you to select which definition you wish to change.

Select either 1 or 2.

After making your selection, the KEY DEFINITION menu will be displayed.

| KEY DEFINITION 1 | |
|---|---|
| UP KEY | :Q |
| DOWN KEY | : A |
| LEFT KEY | : 0 |
| RIGHT KEY | : P |
| FIRE KEY | : SPACE |

The menu title will display the key definition you wished to alter, in this case '1'. The rest of the menu will display the keys that are currently selected. To alter any of the key definitions, select the option you wish to alter.

After making your selection, the following menu will be displayed.

| PRESS A KEY FOR DIRECTION |
|---|

You may now press any key on the keyboard (except ESC). The key you press will then be stored in the KEY DEFINITION menu. The KEY DEFINITION menu will then be re-displayed, allowing you to make more selections.

When you have finished making you selections, press the ESC key.

The Supervisor does not take into account the state of the CAPS, SHIFT or CNTRL keys when you make a selection. Indeed you may select one of the directions to be CAPS.

# CHARACTER SET 1

This option will allow you to create up to 96 proportional characters for use within your games. To create a character, go to the Main Menu and select the CHARACTER SET1 option. The screen will clear and you will be presented with the TEXT 1 menu.

| TEXT 1 |
|---|
| CREATE |
| VIEW |
| TYPE-TEST |
| IMP-EXP |

This menu has very similar options to both the IMAGE and SCENERY menus. If there are currently no characters defined, you will not be able to select the VIEW and TYPE.TEST options.

## *TEXT1* / CREATE

Select the CREATE option. The screen will clear and the CHARACTER SET menu will be displayed. This menu differs from the IMAGE and SCENERY menus, in that instead of representing drawings with numbers, the CHARACTER SET menu shows the alphabet. Any characters that have been defined will be displayed in red. White characters have yet to be defined.

After selecting the character you wish to create/edit the familiar design screen will be displayed. There are a couple of small changes to this screen. Instead of the image number being displayed, the character you have chosen to edit is displayed instead. The drawing type will be displaying TXT1 , signifying you are entering a character set 1 drawing.

All the usual editing features are available through the design screen options menu.

You do not have to design a character to represent the character in the menu. For example, just because you selected the 'H' character from the CHARACTER SET menu, you don't have to define the character to represent a 'H'. However it makes good sense to keep a 'H' as a 'H', just so you can keep track if what you are doing.

You may like to define two upper case character sets, instead of designing an upper case and a lower case set. This is perfectly acceptable.

You may use the mask colour when defining your character sets. Please refer to the discussion on masking character sets in the *introducing masking* section of the manual.

You may, if you wish, design more than 1 set of numeric characters within one character set. The print numbers commands within the Supervisor will allow you to select the starting code for the numbers before they are printed. Normally the character code for 0 is 48, however if you stipulate a different starting character code within the Supervisors PNUMA or PNUMB command then different sets of characters will be displayed. As an example, if you used character 116 as a starting reference, then the following codes would be printed for numbers

0-116 1-117, 2-118, 3-119, 4-120, 5-121, 6-122, 7-123, 8-124, 9-125

This may be useful for defining a different numeric character style for high score tables, etc.

## *TEXT 1* / VIEW

This option is identical to the IMAGE/VIEW option. The view option will now allow you to view all of the character set drawings you have created.

## *TEXT 1* / TYPE-TEST

This option will allow you to link the characters together in a sentence on the screen. You can then test to see if the character spacing has been defined correctly.

Select the TYPE.TEST option. The screen will clear and the CHARACTER SET menu will be displayed. This screen is displayed in the colours defined by the COLOUR section of the Designer. This means that I can not tell you which colours represent defined and undefined characters. However you may only select characters that have been defined, no matter what colour they are displayed in.

You may now select characters from the menu, and they will be printed along the bottom of the screen. The DELETE option will delete the last character in the line.

You will notice that the character at the top left of the menu is the SPACE character. It is impossible to print the space character (for obvious reasons) and so this character was substituted instead.

To return to the TEXT 1 menu press the ESC key.

## *TEXT 1* / IMP-EXP

This option will allow you to load into the Designer, characters that have been created on another art package. The operation of these functions is identical to the IMAGE IMPORT/EXPORT functions

# SOUND EFFECTS

The sound effects section of the Designer will allow you to create Tone envelopes, Volume envelopes, and 32 fixed sound commands. The sound effects manager is extremely versatile, allowing you to create a multitude of different sound effects.

Please refer to the *introducing sound effects* section of the manual for more information about creating sounds.

After selecting the SOUND EFFECTS option from the Main Menu, the screen will clear and you will see the SOUND EFFECTS menu displayed.

| SOUND EFFECTS |
|---|
| TONE ENVELOPES |
| VOLUME ENVELOPES |
| COMMANDS |

This menu will allow you to access all of the parts to making a sound effect. The best way of creating sounds is to experiment, you cannot harm the computer in any way by using strange values. The stranger the values you use, the stranger the sounds.

To return to the Main Menu, press the ESC key.

## *SOUND EFFECTS 1* TONE ENVELOPES

The tone envelopes section of the sound effects editor will allow you to create up to 16 different tone envelopes. After selecting the TONE ENVELOPE option, the screen will clear and you will be presented with the TONE ENV options menu. This menu lists all of the possible tone envelopes you may define. The envelope numbers printed in red contain defined tones envelopes, the numbers in blue represent envelopes that have yet to be defined.

Select a tone envelope number. The TONE ENV menu will be replaced by the TONE ENVELOPE DESIGN menu screen.

**TONE ENVELOPE 05**

| STAGE | STEPS | SIZE | PAUSE |
|---|---|---|---|
| 1 | 000 | +00 | 000 |
| 2 | 000 | +00 | 000 |
| 3 | 000 | +00 | 000 |
| 4 | 000 | +00 | 000 |
| 5 | 000 | +00 | 000 |
| 6 | 000 | +00 | 000 |
| 7 | 000 | +00 | 000 |
| 8 | 000 | +00 | 000 |
| ENTER | EDIT | SAVE | ABORT |

**ENVELOPE TIME 000.00 SECONDS**

This screen consists of three menus. The top menu displays the tone number you have selected to edit/create. The bottom menu will display the time it takes to execute 1 pass of the tone envelope. The middle menu is the main tone envelope editing menu.

You will notice that there are four options at the bottom of the editing menu. If there is no tone envelope defined, you will only be able to select two of the options, ENTER and ABORT.

As you can see from the diagram on the previous page, you may enter up to 8 different stages within a tone envelope.

### *TONE ENV I* ENTER

This option will ailow you to enter data into the tone envelope. You may select this option as long as you have at least one stage of envelope left to define.

Select the ENTER option. The ENVELOPE STEPS menu is now displayed. You may select any step in the range 1 - 255. The ENVELOPE STEPS represent the number of times the specific stage within the tone envelope is repeated.

After selecting an ENVELOPE STEPS option, the next menu to be displayed is the ENV STEP SIZE, you may select any number between -99 to +99. If you have made a mistake in selecting the number of ENVELOPE STEPS, you may press the ESC key to step back 1 menu. The ENV STEP SIZE number informs the sound command of the increase/decrease to the tone value.

After selecting a ENV STEP SIZE option, the next menu to be displayed is the ENVELOPE PAUSE menu, you may select any number between 1 and 255. This number represents the time it will take before the sound manager executes the next step. A value of 1 represents 1/100th of a second, a value of 255 represents 2.55 seconds.

After selecting the ENVELOPE PAUSE time, you will be returned to the TONE ENVELOPE EDITING menu. The values you have selected will then be added to any values that were already displayed within the menu.

Example.

For the first stage in our tone envelope we have selected the following values.

| | | |
|---|---|---|
| ENVELOPE STEPS | : | 020 |
| ENV STEP SIZE | : | +09 |
| ENVELOPE PAUSE | : | 030 |

The ENVELOPE TIME would be displaying 6.00 seconds (20 steps * 30/100ths second delay).

If we set an initial tone (more about this later) to be 15, the sound manager, on first executing this tone envelope, would output a tone value of 24 (15+9) and the envelope steps would be reduced to 19. Then 30/100ths of a second later (envelope pause) the sound manager would then output a tone value of 33 (24+9), the envelope steps would be reduced to 18. The process would continue until the ENVELOPE STEPS value reached zero.

5.99 seconds after the tone envelope first started, the tone would be completed. If we had defined a second stage in the tone envelope, the sound manager would then start to work its way through the second stage.

The TONE ENVELOPE editor will allow you to create tone which will last a very long time, in practice most tone envelopes will only last a few seconds, at the very most.

### *TONE ENV /* EDIT

This option will allow you to edit an existing tone stage. Select the EDIT option. A menu will appear at the top of the screen

| SELECT STAGE TO EDIT |
|---|

You may now move the pointer over the tone envelope main menu and select a stage. After select a particular tone stage the following menu will be printed.

| EDIT STAGE |
|---|
| ALTER |
| DELETE |

If you press the ESC key whilst this menu is displayed, you will return to the SELECT STAGE section.

### *EDIT STAGE /* ALTER

This option will allow you to change the existing values for the stage you selected. You enter the envelope values as if you were entering a new stage. Once you have selected all three variables, you will return to the TONE ENVELOPE EDITING menu.

### *EDIT STAGE /* DELETE

This option will allow you to delete the tone stage that you had previously selected.

### *TONE ENV /* SAVE

This option will save the currently defined tone envelope. After selecting the SAVE option you will be returned to the TONE ENV number selection menu.

### *TONE ENV /* ABORT

This option will erase the tone envelope from memory. After selecting the ABORT option, you will be asked to verify your choice with the following menu.

| ERASE THIS ENVELOPE |
|---|
| YES |
| NO |

Selecting the YES option will erase the envelope and return you to the TONE ENV number selection menu.

Selecting the NO option will cancel the command.

# *SOUND EFFECTS* / VOLUME ENVELOPE

This option will allow you to edit/create up to 16 different volume envelopes. The section is identical to the TONE ENVELOPE editing section and so I will not repeat myself by giving you the same instructions twice.

Although the editing section is identical, the volume envelopes will loop around once the volume value goes above 15 (maximum volume). You can create some special effects by making the volume envelope loop around.

# *SOUND EFFECTS* / SOUND COMMANDS

This section of the sound effects editor will allow you to link together the various tone and volume envelopes to produce a sound. This option will allow you to create up to 32 fixed sound effects. These sound effects will be used by the sprites when colliding, bouncing, exploding etc. Although the sound effects editor will only allow 32 fixed sounds, you may create an unlimited number of sounds using the Supervisor. The sounds you will create here are for use with the automatic sound generation routines. Please see the Supervisor section on sound for more details.

After selecting the SOUND COMMANDS option, the screen will clear and you will be presented with the SOUND number options menu. This menu will allow you to select the sound number you wish to create/edit. The numbers printed in red represent sounds that have already been defined, numbers in white represent sounds that have not, as yet, been defined.

Select a sound number. The screen will clear and the sound editing menu will now be displayed. If you selected a sound that has already been defined, the values for that particular sound will be displayed, allowing you to edit the sound command further, or simply play the sound.

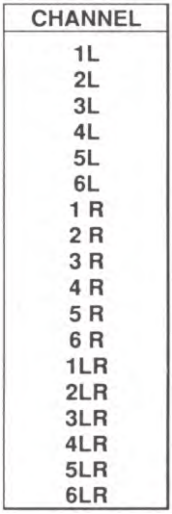
The sound editing menu looks like this

| SOUND COMMAND 12 | | | | | | | |
|---|---|---|---|---|---|---|---|
| **CHAN** | **OCTAVE** | **TONE** | **LENGTH** | **VOLUME** | **NOISE** | **VOL ENV** | **TONE ENV** |
| **1LRc** | **00** | **128** | **001.23** | **02** | **00** | **NO DEF** | **NO DEF** |

You may select any of the above options, by moving the pointer over the relevant section and pressing the SELECT key

## *SOUND EDIT* / CHAN

This option will allow you to specify which channel (1-6) the sound should be played through. You will also need to specify the stereo placing of the sound LEFT,RIGHT,LEFT-RIGHT. Finally you will need to specify the sound priority (queue,clear). You can not save or play a sound until this option has been selected and the required data entered into the menu.

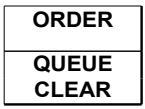



This is the first stage of the CHANNEL menu. The menu will allow you to select both the channel number, and the stereo status. 'L' represents the left channel, 'R' represents the right channel.

Selecting a value 4R represents playing the sound on channel 4, on the right speaker only.

You may create some stunning stereo effects by playing two different sounds on separate channels.

After selecting this option, the channel menu will be removed and it will be replaced by the ORDER menu.The order menu has two options. The options define the priority of the sound to be played.




Selecting the QUEUE option will cause the sound to be placed in the sound queue. The sound will be played once the sound gets to the front of the queue.

Selecting the CLEAR option will cause this sound to take top priority. Any sounds that were in the queue would be cleared and this sound would be played straight away.

After you have made your selection, the information you selected will be placed into the SOUND COMMAND menu.

### *SOUND COMMAND* / OCTAVE

This option will allow you to select a the starting octave for the particular sound. Upon selecting this option, the OCTAVE menu will be displayed.

The OCTAVE menu consists of the numbers from 0 to 7, each number represents an octave. Number 0 being the lowest octave, number 7 being the highest octave.

### *SOUND COMMAND* / TONE

This option will allow you to select a starting tone for the sound you are currently defining. Upon selecting the TONE option, the INITIAL TONE menu will be displayed.

The initial tone menu consists of all the possible tone numbers (0-255). Select a tone number and it will be inserted into your sound command.

## *SOUND COMMAND* / LENGTH

This option will allow you to specify the length of time the sound will take to play. There are two ways you can specify the length. Firstly, fixed length timing will allow the sound to play for a fixed length of time. Volume env timing, will allow the sound to play until the volume envelope has been completed.

If you are not going to be using a volume envelope, then you must use fixed length timing.

Select the LENGTH option. The following menu will be displayed.

| TIME OPTIONS |
| --- |
| FIXED TIME |
| REPEAT RATE |

### *TIME OPTIONS* / FIXED TIME

This option will allow you to specify a fixed length of time for the sound to play. After selecting the FIXED TIME option, the menu will be removed and the TIME SECONDS menu will be displayed.

You may select any time from 0 and 255 seconds. After making your selection, the menu will be replaced by the TIME HUNDREDTHS menu. You may now select how many hundredths of a second the sound will be played for.

After making your selection, you will be returned to the SOUND COMMAND editing menu.

### *TIME OPTIONS* / REPEAT RATE

This option will allow you to select the number of times the volume envelope is repeated. If you are not using a volume envelope, you will not be able to use this option.

You may select the volume envelope play just once (option 1), or to repeat up to 15 times (option 15). After making your selection you will return to the SOUND COMMAND editing menu.

## *SOUND COMMAND* / VOLUME

This option allows you to specify the starting volume for your sound. If you are using a volume envelope, then this value will change whilst the sound is playing.

Select the VOLUME option. The VOLUME menu will be displayed. You may now select the starting volume for the sound. You may specify any number in the range 0 (silent) to 15 (full blast).

## *SOUND COMMAND* / NOISE

This option will allow you to select the type of sound that will be played. You can opt to have just pure tone, pure noise or a mixture of tone and noise. on selecting the NOISE option, the noise menu will be printed.

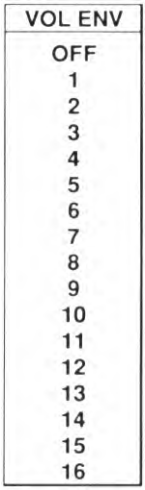
The noise menu consists of a number range from 0 to 8. Please see the table over to find out how each option works.

| Noise Type | Noise Action |
|---|---|
| 0 | Tone only |
| 1 | Tone + 31.24khz noise |
| 2 | Tone + 15.6khz noise |
| 3 | Tone + 7.8khz noise |
| 4 | Tone + variable noise |
| 5 | Noise 31.25khz |
| 6 | Noise 15.6khz |
| 7 | Noise 7.6khz |
| 8 | Variable noise |

It should be remembered that you may only select to use a noise option ( 1 - 8 ) when you are using channels 1 and 4. If you try to use noise on any other channel, an error message will be displayed.

## *SOUND COMMAND* / VOL ENV

This option will allow you to select one of 16 volume envelopes. On selecting this menu option, the volume envelope selection menu will appear.




This menu will allow you to select a volume envelope that you have previously defined using the volume envelope editor.

The numbers that are displayed in red represent volume envelopes that have already been defined. The numbers in blue represent undefined volume envelopes. You may only select a volume envelope that has already been defined.

The top option, OFF will allow you to cancel a currently selected envelope.

After making your selection, the volume envelope selection menu will be removed. If you have selected a volume envelope, the information menu about the volume envelope you selected will be displayed at the bottom left hand side of the screen.

The sound editing menu will now display the volume envelope number within the VOL ENV section. Displayed after the volume envelope number is a description of how the sound will play, the wording of this description will depend on which selection you made when specifying the length of the sound.

If you selected that the sound would play for a fixed length of time then the word **FIX** will be displayed after the volume envelope number.

If you selected the sound to play for a certain number of times, then the word **REP** will be displayed after the volume envelope number.

The currently selected volume envelope will be displayed on the bottom left hand side of the screen. This will allow you to plan exactly how long the sound will play for. The envelope playing time for the volume envelope is displayed underneath the volume envelope information menu. If you have selected for the volume envelope to repeat a number of times, then all you have to do is multiply the number of times the envelope will repeat with the playing time for the single envelope.

### *SOUND COMMAND* / TONE ENV

This option will allow you to select one of 16 tone envelopes. On selecting this menu option, the tone envelope selection menu will appear.

| TONE ENV |
|---|
| OFF |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |

This menu will allow you to select a tone envelope that you have previously defined using the tone envelope editor.

The numbers that are displayed in red represent tone envelopes that have already been defined. The numbers in blue represent undefined tone envelopes. You may only select a tone envelope that has already been defined.

The top option, OFF will allow you to cancel a currently selected envelope.

After making your selection, the tone envelope selection menu will be removed, and the tone play type menu will be displayed.

| TONE |
|---|
| REPEAT |
| FIXED |

This menu will allow you to select how the tone envelope will behave if the tone envelope has been completed and the sound is still playing.

#### *TONE* / REPEAT

Selecting this option will allow the tone envelope to repeat itself, whenever the tone envelope has been completed. The tone envelope will keep repeating itself until the sound that is currently playing comes to an end.

### *TONE*/FIXED

Selecting this option will allow the tone envelope to only be executed once. If the sound has not completed playing when the tone envelope has finished, then the tone will remain constant until the end of the sound.

If you have selected a tone envelope, the information menu about the tone envelope you selected will be displayed at the bottom right hand side of the screen.

The sound editing menu will now display the tone envelope number within the TONE ENV section. Displayed after the tone envelope number is a description of how the tone will play, the wording of this description will depend on which selection you made in the REPEAT / FIXED menu.

It is normally a good idea to set the tone envelope to repeat.

## PLAYING / SAVING A NOTE

To play, save or erase a sound, press the ESC key from within the sound command menu. After pressing the ESC key, the following menu will appear.

| SOUND OPTIONS |
| --- |
| SAVE SOUND |
| ERASE SOUND |
| PLAY SOUND |
| PLAY SCALE |

### *SOUND OPTIONS* / SAVE SOUND

This option will save the sound command that is currently displayed on the screen. Before this option is executed a number of checks take place. This may result in one of the following error messages being displayed.

**ERROR : NOISE ILLEGAL ON CURRENT CHANNEL**

This error message is informing you that you have selected to output a noise on a channel that does not allow noise output. The only channels on which you may use noise are 1 and 4.

**ERROR : CHANNEL INFORMATION HAS NOT BEEN DEFINED**

This error message is informing you that you have not defined a channel on which to output your sound.

**ERROR : SOUND CANNOT REPEAT WITH NO VOLUME ENVELOPE DEFINED**

This error message is informing you that you have tried to set the length of sound to repeat, even though you have not specified a volume envelope. You should either select a volume envelope, or, change the length of the sound so that the sound plays for a fixed length of time.

**ERROR : SOUND HAS NO DURATION**

This error message is informing you that you have not specified the length of time the sound should play for. You should either select a fixed length sound, or, select a sound to repeat a volume envelope.

If there is an error message printed, then pressing the ESC key will remove the error message and allow you to enter a new piece of data.

If there are no errors within your sound definition, the sound command information will be saved to memory and you will be returned to the sound command selection menu.

## *SOUND OPTIONS* / ERASE SOUND

This option will allow you to clear the information on the sound command from memory. The volume and tone envelopes will not be erased. After selecting this option, the

**ARE YOU SURE YOU WISH TO DELETE THIS SOUND COMMAND**

**YES**
**NO**

Selecting the YES option will cause the sound command to be cleared from memory.

Selecting the NO option will cause the erase operation to be cancelled.

## *SOUND OPTIONS* / PLAY SOUND

This option will allow you to test a sound. Selecting this option will play the sound that you have currently defined. The sound information is tested for any errors before the sound is played. If any errors are detected, they will be reported. The error checks are the same as the checks performed by the SAVE option.

If there are no errors within the sound command, the sound will play. Whilst the sound is playing, the following message will be displayed.

**PRESS ESC TO ABORT SOUND**

Pressing the ESC key will cause the sound to stop playing.

It should be noted that the sound is updated 100 times every second. If you are playing the sound in the Supervisor and there are sprites moving on the screen, then the sound

may not sound exactly as it is played within the Designer. This is due to the fact that the Supervisor may be too busy updating the sprites. This is discussed within the *introducing sound effects* section of the manual.

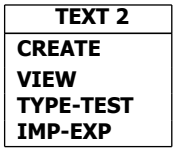
### *SOUND OPTIONS* / PLAY SCALE

This option is very similar to the PLAY SOUND option. Instead of simply playing a single note, the Designer will play a full scale of notes one after another. This tone value you selected within the sound command menu will be ignored. Instead, the value of the tone variable within the sound command will be altered so that a full scale can be played. The scale will be played in the octave selected within the sound command.

This operation will not physically alter the value of the tone you specified within the sound command.

# CHARACTER SET 2

This option will allow you to create up to 96 proportional characters for use within your games. To create a character, go to the Main Menu and select the CHARACTER SET2 option. The screen will clear and you will be presented with the TEXT 2 menu.




This menu has very similar options to both the IMAGE and SCENERY menus. If there are currently no characters defined, you will not be able to select the VIEW and TYPE.TEST options.

## *TEXT2* / CREATE

Select the CREATE option. The screen will clear and the CHARACTER SET menu will be displayed. This menu differs from the IMAGE and SCENERY menus, in that instead of representing drawings with numbers, the CHARACTER SET menu shows the alphabet. Any characters that have been defined will be displayed in red. White characters have yet to be defined.

After selecting the character you wish to create/edit the familiar design screen will be displayed. There are a couple of small changes to this screen. Instead of the image number being displayed, the character you have chosen to edit is displayed instead. The drawing type will be displaying TXT2, signifying you are entering a character set 2 drawing.

All the usual editing features are available through the design screen options menu.

## *TEXT 2* / VIEW

This option is identical to the IMAGE/VIEW option. The view option will now allow you to view all of the character set drawings you have created.

## *TEXT 2* / TYPE-TEST

This option will allow you to link the characters together in a sentence on the screen. You can then test to see if the character spacing has been defined correctly.

## *TEXT 2* / IMP-EXP

This option will allow you to load into the Designer, characters that have been created on another art package. The operation of these functions is identical to the IMAGE IMPORT/EXPORT functions

# MISSILES

This option, selectable from the main menu, will allow you to enter all of the missile data for the sprite images. This data will allow the sprites, under control of the Supervisor, to fire missiles.

After selecting the MISSILES option, you will be presented with the missile firer menu. This menu consists of all of the image numbers in the range 0 to 255. The numbers may be any one of three different colours. The colour of the image number will inform you on the current missile status of the image.

## *MISSILE FIRER MENU*

If the image number is displayed in light blue, then the image has not been defined. You will not be able to define any missile information for these images (for obvious reasons). If the image number is displayed in dark blue, then the image has been defined, but there is currently no missile definition for that image. If the image number is displayed in red, then the image has been defined and there is a missile definition for that particular image.

If you decide you do not wish to create a missile definition, you may press the ESC key to return to the main menu.

If you select a red image number (missile defined), then you will skip the next few menus.

After selecting a dark blue image number (missile undefined), the missile firer menu will be removed, and the missile image menu will be displayed.

## *MISSILE IMAGE MENU*

This menu is asking you to select a number of the image you wish to use as the missile. You may select any of the dark blue numbers (defined image), don't worry if you make a mistake, you can step back to this menu and make another selection. If you do not wish to select an image number, pressing the ESC key will take you back to the MISSILE FIRER menu.

This menu consists of all of the image numbers in the range 0 to 255. The numbers may be one of two colours. The colour of the image number will inform you on the current definition status of the image. If the image number is printed in light blue, this signifies that the image has not been defined. If the image number is printed in dark blue, this signifies that the image has been defined.

You can not select image 0 to be a missile image. This is due to the way the image numbers are stored in memory by the Designer. I hope you don't find this to much of a problem.

After selecting the image number you wish to use as a missile, the screen will clear and the missile offset selection screen will be displayed.

## *MISSILE OFFSET SELECTION*

The selection screen will be displayed in the colours you have selected in the COLOURS menu. The missile firing image and the missile image will be displayed near the top of the screen.

You may move the missile image around the screen using the arrow keys or the mouse. The idea of this screen is to define the start position of the missile in relation to the image that is going to fire the missile.

If you find you have made a mistake and the two images that are displayed are not the images you wish to use, you may press the ESC key to return to the image selection menus.

Please remember when you are selecting the starting position for the missile image, that you should allow enough clearance between the two images so that the image collision boxes do not collide. This is discussed within the *introducing missiles* section of the manual.

Once you are happy with the position of the missile image, press the select key. The missile image will now be frozen and the direc menu will be displayed

## *DIREC MENU*

This menu will allow you to select the direction the missile will travel when it is fired. The DIREC menu displays eight arrows which represent the eight different directions the missile can travel. Selecting one of the directions will take you to the speeds menu. Pressing the ESC key, will take you back to the missile offset selection screen.

### *DIREC* / X SPEED

If you have selected any speed other than up/down, then you will be asked to enter a value for the speed the missile will travel on the X axis. The following menu will be displayed.


You may now select a value for the X speed of the missile.

All of the values within the X SPEED menu are positive, the Designer will automatically assign a negative value if the missile is travelling to the left.

If you have made a mistake in selecting the direction the missile will travel, press the ESC key, this will step you back to the direction selection menu.

### *DIREC* / Y SPEED

If you have selected any speed other than left / right, then you will then be asked to enter a value for the speed the missile will travel on the Y axis. The following menu will be displayed.

| Y SPEED |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |

You may now select a value for the Y speed of the missile.

All of the values within the Y SPEED menu are positive, the Designer will automatically assign a negative value if the missile is travelling towards the bottom of the screen.

If you have made a mistake in selecting the direction the missile will travel or the X speed, press the ESC key, this will step you back to the last menu.

After making your speed selections, the MISSILE INFO menu will be displayed.

| MISSILE INFO | |
|---|---|
| SPRITE IMAGE | : 076 |
| MISSILE IMAGE | : 250 |
| MISSILE X OFFSET | : +39 |
| MISSILE Y OFFSET | : -13 |
| MISSILE X SPEED | : -02 |
| MISSILE Y SPEED | : +01 |

| MISSILE |
|---|
| SAVE |
| ERASE |
| EDIT |

The menu on the left hand side of the screen displays all of the selections you have made. The menu on the right hand side of the screen will allow you alter the settings, save the missile information, or erase the information.

### *MISSILE / SAVE*

This option will save the missile information you have defined to memory. After saving the information, you will be returned to the MISSILE FIRER menu.

### *MISSILE* / ERASE

This option will allow you to clear the missile information for a particular image. After selecting the ERASE option, the following menu will appear.

| CONFIRM : CLEAR THIS MISSILE |
|---|
| YES<br>NO |

If you select the YES option, the missile information will be erased and you will return to the MISSILE FIRER selection menu.

If you select the NO option, the erase instruction will be ignored.

### *MISSILE* / EDIT

This option will allow you to alter the missile settings. After selecting the EDIT option, you will be returned to the DIREC selection menu.

# USING THE SUPERVISOR PROGRAM

The SCADs Supervisor program extends Sam BASIC with a host of powerful games-writing commands. By now you should be fully familiar with the Designer program, you hopefully understand all of the concepts used by this sprites package. It is now time to write your first program.

## *LOADING THE SUPERVISOR*

Before any of your programs can be entered into the computer, the SCADs extended BASIC (Supervisor) must be loaded into memory. If you try and enter any SCADs commands when the Supervisor is not loaded, Sam BASIC will return a NOT UNDERSTOOD error.

To load the Supervisor, turn the computer off, wait 5 seconds ( this will clear any programs that may exist in memory ), and then turn the computer back on again. Insert the SCADs system disc and press the f9 key. Select the Supervisor option and wait while the program loads into memory.

Once the Supervisor has loaded into memory, the computer will reset, this is perfectly normal. The SCADs Supervisor is now activated. The Supervisor will stay in memory until you either turn the computer off, or press the reset key on the back of the computer. Typing NEW will not remove the Supervisor.

Certain Sam BASIC commands will cause the Supervisor to crash. This is because the Sam BASIC commands try to use an area of memory which is used by the Supervisor. In order to prevent the Supervisor from crashing, the commands listed below have been disabled.

### OPEN, CLOSE, GRAB, PUT, SCROLL, ROLL, SCREEN, DISPLAY, SOUND

The OPEN and CLOSE commands are disabled because they will alter the memory configuration, the configuration is fixed by the Supervisor. You have 41158 bytes available for use within your programs.

The GRAB, PUT, SCROLL and ROLL commands are disabled because they will overwrite a section of memory used by the Supervisor.

The SCREEN and DISPLAY commands are disabled because the Supervisor has already opened all of the screens that are required by SCADs.

The SOUND command is disabled as the Supervisor takes over the sound chip, any outside interference using the SOUND command may result in the sound becoming garbled.

# ENTERING SCADS COMMANDS

The Supervisor is now loaded into memory, the screen is blank and we are ready to begin. All SCADs commands are preceded by a special character. This character informs the Supervisor that the command is a SCADs command. All of the SCADs extended BASIC commands must have this special character before the command. There should NOT be a space between the special character and the rest of the command.

The special character is displayed by pressing the **f0** function key. The character looks like a small solid square.

SCADs commands act in a very similar way to BASIC commands. If you enter a command in lowercase letters and the Supervisor recognises the command, the letters will be converted into uppercase letters when you press ENTER. There MUST be a space between the command and the first variable.

## *ENTRY ERRORS*

If you have made a mistake when entering the command, the line error symbol, an inverse question mark, will be placed at the point in the line where the error occurs, and the computer will beep. There are two types of errors that may occur when entering a command.

**UNRECOGNISED COMMAND:** The command you have entered is not recognised by the Supervisor. The error mark will be displayed after the command name and the command name will not be converted into uppercase letters.

**INCORRECT NUMBER OF VARIABLES:** The number of variables you have entered after the command is incorrect. If you have not entered enough variables, the error mark will be displayed after the last variable. If you have entered too many variables, the error mark will be displayed after the last legal variable. All of the variables after the error mark should not have been entered.

Please remember that some of the Sam BASIC commands have been disabled. This may result in the error mark being displayed after a Sam BASIC command. Please see the previous page for a list of illegal Sam BASIC commands.

## *RUN TIME ERRORS*

There are a number of errors that may occur when you are running a SCADs program. If an error occurs, the computer will stop and the following message will be displayed.

**SCADs ERROR No:013**

Unfortunately due to a lack of space in the Supervisor, the program does not display full error messages. Instead of printing an error message, an error number is displayed. You will need to refer to the back of this manual for an explanation of the error number that is being displayed.

move.

The system incorporated into the Supervisor will overcome this problem. The sprites updated positions are not printed to the screen until the VIEW command is issued. The VIEW command informs the Supervisor to print all of the sprites onto the screen.

**MOVES 1:MOVES 6:MOVES 12:VIEW**

This command line executes 3 times faster because the screen is only updated once instead of three times. The sprites would also appear to move smoothly because they all appear to move at the same time.

It is quite possible (but silly) to write a complete game without updating the screen at all. You would not be able to see the sprites moving (as the screen is not being updated), but the sprites would act in exactly the same way as when the screen was being updated. What **I** am trying to say, is that it does not make any difference to the Supervisor if the sprites are displayed or not, the sprites will still bounce off scenery and window edges, etc, even though they are not printed onto the screen. The VIEW command simply takes a snapshot of the current position of the sprites and displays them.

Within your program, you should have a main loop. A main loop is a section of program that is repeated over *and* over again. This is the loop that contains all of you sprite movement commands and collision detection routines. This main loop should also contain a VIEW command. The VIEW command should be included only once, after all of the sprite movement commands. Using the VIEW command only once will speed up the execution of your program. You may, of course, use the command more than once, but this is not advisable.

# WRITING A PROGRAM

Enough of the theory for the time being, lets write your first SCADs program. Lets create a Breakout game. This game is fairly simple to program, and we can add some special effects. Please load the Supervisor into the computer as described in the *getting started* section of the manual.

It is important to remember that ALL SCADs commands must be preceded by the SCADs command symbol. Throughout this chapter of the manual I will use the # symbol. Whenever you see this symbol within the listings in this manual, you should press the **f0** function key on the Sam keyboard. This will display a small solid square. Do not enter a # symbol using the sam keyboard.

Please type the lines of program into the computer, your going to have fun. Ignore the breaks in line numbers the whole program will come together in the end.

Before you can start to use the SCADs Supervisor, you must have a data file loaded into memory. The data file will have been created using the Designer program, the data file for this demonstration is called DRAW 2.

The first lines of the program should be standard for all SCADs BASIC programs. When the program is first ran you should type RUN 20. You only need to load the drawing data into the Supervisor once. Therefore if you require to run the program again, simply type RUN.

```
10 GO TO 30
20 #FILE "DRAW2"
30 #INIT
```

The last section of program will load the drawing draw file DRAW2 into SCADs memory. The INIT command should always be included at the start of your programs.

Next, we need to define some sprites. We shall call the bat sprite, sprite 0, and the ball sprite, sprite 1

```
40 #LINK 0,111:#SWINDOW 0,87,247,0,184
50 #LINK 1,107:#SWINDOW 1,87,247,0,184
```

Lines 40 and 50 define 2 sprites. After the sprites have been defined using the LINK command, we create a sprite window for each of the sprites. When the sprites are first initialised, the sprites window is set to the full size of the screen. This is not acceptable in our game, and so we define a smaller window for the sprites.

```
180 #SPUT 0,140,16,1:#SPUT 1,140,30,1
210 #SPEED 1,-3,4
```

The commands in line 180 place 2 sprites onto the screen. The first command places

# GETTING STARTED

Before I go into the details of how to program using the SCADs extended BASIC, I think we had better do some more theory into how the sprites actually move.

The Supervisor system uses a two screen method for moving the sprites. This is the most popular method used by games programmers as it allows the sprites to move extremely smoothly. Don't worry about all of this theory, the Supervisor does all of the screen handling for you. If you are not in the least bit technically minded then please skip this section of text, although I would advise everybody to read it.

When the Supervisor is loaded into memory, another screen is opened, therefore there are two screens open, the standard BASIC screen and the screen that has just been opened. We shall call these two screens A and B.

At the start of your program screen A is being displayed on the monitor. When the sprites are moved, the screen that is not currently displayed (screen B) is updated. Once this undisplayed screen (B) has been completely updated, the screens are swapped, the screen that has just been updated (B) is now displayed and the screen that was being displayed (A) is switched out.

When the sprites are next moved, the process is reversed. The screen that is not currently displayed (A) is updated. Whilst the updating is taking place screen (B) is still being displayed. Once screen (A) has been completely updated, the screens are again swapped around. This process is repeated over and over.

You can not detect the screen change over, this is because the screens are swapped in a fraction of a second.

All we have said in the above description is that the screens are updated, the actual process of updating the screen is shown below.

In order to prevent the screen from becoming corrupted by sprite movement, the screen area where the sprite is going to be placed is copied into a separate section of memory called the screen buffer. After the section of screen is stored, the sprite image is printed over the top of the background. When the sprite is about to move, the background is retrieved from the screen buffer and reprinted onto the screen over the top of the sprite, this will completely remove the sprite. The sprite position is then updated and the procedure is repeated, the new background is then stored into the screen buffer, etc.

The process described above is fine if there is only one sprite on the screen, but there may be problems when two or more sprites are passing over each other. To overcome this problem, before any of the sprites are printed onto the (un)displayed screen, all of the backgrounds are stored into the screen buffer, and then all of the sprites are printed onto the screen. This will overcome the problem of screen corruption, but there is a drawback. If there are 20 sprites printed onto the screen, and you only wish to move 1 of the sprites, then ALL of the sprites have to be removed, the single sprite that is moving has its coordinates updated. All of the sprites then have to store the background again, and then all of the sprites are reprinted.

This process is fine if all the sprites are moving. However, in the game of breakout most of the sprites are stationary (block sprites), and only two sprites are moving (bat, ball). Lets say that there are 60 block sprites displayed on the screen. The Supervisor has to remove all 60 sprites (blocks) and then replace them all back onto the screen again whenever the bat or ball move. Removing and replacing 60 sprites takes a great deal of time, and so the bat and ball will only move relatively slowly. The way around this problem is to create a special sprite that will not be updated every time a sprite moves on the screen. We have called this special sprite a permanent sprite. A permanent sprite is a sprite that will never move,it is placed onto the screen before any other sprite and it does not get updated when all of the standard sprites are updated. The only action you may perform on a permanent sprite is to remove it. Please see the discussion on permanent sprites a little later in the manual.

Below is a list of actions that are performed by the Supervisor every time the screen is updated.

1) Replace the backgrounds of all non permanent sprites on the screen.

2) Update all of the sprites positions.

3) Print any new text onto the screen.

4) Store all of the backgrounds on the screen into a screen buffer.

5) Print all of the non permanent sprites back onto the screen.

6) Print any foreground scenery onto the screen.

7) Swap screens. ie display new updated screen.

You will notice that any foreground scenery that should be displayed on the screen is printed to the screen last. This is so that all sprites appear to be moving behind the foreground scenery.

When you move a sprite, the position of the sprites coordinates are updated, but the sprite is not actually moved on the screen. The screen is only updated whenever you inform the Supervisor to update the screen. This may cause a bit of confusion at first, but there is a simple reason why we have made the Supervisor act in this way.

Imagine there are 12 sprites printed onto the screen. If you wanted to move only three of the sprites, say sprites 1,6 and 12, you may use the following commands

**MOVES 1:MOVES 6: MOVES 12**

If the screen was updated after every move instruction, then all 12 sprites would have to be updated after the first move command, then all 12 sprites would have to be updated after the second move command, and finally all 12 sprites would have to be updated after the last move command. This means in order to move 3 sprites, the Supervisor would need to update all of the sprites on the screen 3 times, this would cause your programs to slow down, and the sprites would not move smoothly. If this command line was repeated over and over again, then the sprites would appear to be taking turns to

sprite 0 (bat sprite) at position 140,16 and the sprite is placed in plane 1. The second command places sprite 1 (ball sprite) at position 140,30 and the sprite is also placed in plane 1.

The command in line 210, allocates a speed of -3 (x),4 (y) to sprite 1. This is the ball sprite.

Lets see the sprites moving.

```
230 #MOVEALL:#VIEW
250 GOTO 230
```

The MOVEALL instruction in line 230 instructs the Supervisor to move all of the sprites. The VIEW command in line 230 instructs the Supervisor to update the screen. Line 250 simply makes this process repeat indefinitely.

Now type **RUN 20** and press return. After a small delay while the drawing file loads into memory, you will see a strangely coloured ball bouncing around the screen. A stationary bat is placed at the bottom of the screen. Once this is working press the ESC key. This is the first and ONLY time you will type RUN 20, from now on whenever you are asked to run the program you should just type RUN and press return. If an error message is displayed, please check your typing and correct any mistakes.

Lets sort out those strange colours. Type (without a line number) **#COLOUR** and press return. Now try running the program again, just type RUN remember and NOT RUN 20. The colour scheme looks a little better this time. The COLOUR command changes the colours to the colours that have been defined within the Designer program.

Lets get the bat moving.

```
100 #SETINP 0,2,5,0,68,"00000000"
```

This defines sprite 0 (bat sprite) to be controlled by the keyboard input (2). The sprite will move at a speed of 5 on the x axis, and 0 on the y axis. The value 68 informs the Supervisor to only allow the sprite to move left and right. The information within the quotes does not need to be defined for this demonstration.

Now try running the program again. You can now control the bat using the cursor keys.Both of the sprites on the screen are moving inside their own sprite windows. You will notice that the ball is passing straight through the bat, we shall deal with this now.

```
110 #BOUNCE 1
```

This command informs sprite 1 to bounce off all of the other sprites. Try running the program again.

The screen looks a little bare at the moment, lets add some background graphics. I have created a *room* within the Designer program for use with this demonstration.

```
120 #ROOM 14
```

Try running the program now. The screen looks a little bit better. All of the graphics within the room are background. They are displayed simply to enhance the look of the game.

Its a very strange game of Breakout at the moment, there are no blocks, lets define and display the blocks.

```
60 LET J=200:FOR 1=2 TO 55:#LINK I,J
70 LET J=J+6:IF J=224 THEN LET J=200
80 NEXT I:LET SC=0
```

The block images start at image number 200, and are spaced 6 images apart. The last block image is image 218. The images are different colours. These three lines create 54 sprites. The LET SC=0 instruction is clearing the score variable, this will be discussed a little later.

We have now created all of the block sprites, we shall now display them.

```
140 LET S=2:FOR Y=1 TO 6:FOR X=1 TO 9
150 #PSPUT S,X*18+70,Y*14+96,1:LET S=S+1
160 NEXT X:NEXT Y
```

The three lines above will display the block sprites. The sprite (s) is being placed at a position calculated by the x and y loops. The sprites are placed in sprite plane 1. Type RUN now and see the results. Isn't it colourful !

Lets add collision detection. You will need to type over a line that has already been entered.

```
230 #HIT 1,LENGTH(0,X):#MOVEALL:#VIEW:IF X<>0
      THEN GO TO 260
260 #NEXTHIT LENGTH(0,X):IF X=0 THEN GO TO 230
270 #TOGGLE X
310 GO TO 230
```

The HIT command in line 230 will instruct the Supervisor to store in variable X the number of sprites that are in collision with sprite 1 (ball). After the teSt, the sprites are moved (MOVEALL), and the screen updated (VIEW). We then test variable X to see if any of the sprites are in collision with the ball. If there is a collision, the program will jump to line 260.

The NEXTHIT command in line 260 will report the sprite number of the sprite that is in collision with the ball. The number of the sprite that is in collision with the ball is stored

in the X variable. The program will then test to see if the ball is in collision with the bat ( sprite 0). If the ball is in collision with the bat, no further action is taken, and the program will jump back to the start of the main loop (line 230). If the sprite number reported in variable X is not sprite 0, then the ball must be in collision with a block. Line 270 will remove the block sprite (X) from the screen. We then jump back into the start of the main loop. Try RUNning the program now.

It is still not possible for you to 'lose a life', the ball will simply bounce back off the bottom of the screen. We shall deal with this shortly. Firstly lets introduce some sound.

## 90 #AUTOSOUND 1,6,1

This instruction informs the Supervisor to automatically play a sound whenever a certain action takes place. In this case, the instruction is informing the Supervisor, whenever sprite 1 (ball) hits a window edge (6) make sound (1). The complete list of actions is detailed within the *command guide* section of the manual. Sound 1 was created within the Designer program. Try RUNning the program. Whenever the ball hits a window edge, the Supervisor will play sound 1. You will notice that whenever the ball hits a block, no sound is played.

## 290 #SCOM 2

This command will play sound 2. It will not wait for any actions, the sound will be played straight away. Line 290 is only ran when a block is removed from the screen. Therefore whenever a block is removed, sound 2 will be played. Try RUNning again.

Lets add some special effects.

## 270 #RIMAGE X,LENGTH(0,J):#TOGGLE X:#TOGGLE X
## 280 LET J=(J-200)/6+7:#ANIMOFF X,J,9,3

The lines 270 and 280 are fairly complex. The RIMAGE command will return the image number, in variable j, that sprite X is currently using. This is used by the commands within the next line. The TOGGLE commands, firstly remove the sprite, and then replace the sprite back onto the screen. This may seem totally illogical, but there is a very good reason for doing it. All of the block sprites that are displayed on the screen are permanent sprites, (refer to the section on permanent sprites for more details). Permanent sprites can not perform any actions, they can only be removed from the screen. The first TOGGLE command removes the permanent sprite. The next TOGGLE command will then replace the sprite back onto the screen again. As soon as a permanent sprite is removed from the screen, it is not classed as being permanent anymore, therefore when the block sprite is replaced back 'onto the screen it is not a permanent sprite anymore. This allows us to perform the ANIMOFF command. You can not use the ANIMOFF command on permanent sprites.

We have now changed the sprite from a permanent sprite to a normal sprite. Line 280 calculates, from the sprite image number, an animation sequence which will be used to animate the sprite off the screen. The animation sequence's to remove a sprite start at

sequence number 7, therefore image 200=7, image 206=8, image 212=9, image 218=10. Remember, there are 4 coloured blocks on the screen, the blocks are made up from image numbers 200,206,212, and 218.

The ANIMOFF command will remove sprite X from the screen, using animation sequence number j, using sequence range 9 with a delay of 3 moves between each stage.

Try running the program now. It is starting to look a bit more *professional !!*

I want you to try a little experiment, I hope this will convince you to use permanent sprites wherever possible. Move the cursor to line 150 and press the EDIT key. I want you to change the first command from PSPUT to SPUT, simply delete the P. Now press return. Now RUN the program, isn't it slow. This demonstrates the advantage of using permanent sprites. When the Breakout game is using permanent sprites, only 2 sprites are normally updated on the screen, sprite 0 (bat) and sprite 1 (ball). There may also be a number of sprites being animated off the screen and these sprites are also being updated. However, by changing the PSPUT command to a SPUT command, the Supervisor has to update all of the sprites that are currently being displayed, all 56 of them when the game first starts. If you let the game run using normal sprites, then you will notice that eventually the ball will start to speed up. As more and more sprites are being removed from the screen, the Supervisor has less sprites to update, and so the ball will start to increase in speed.

Go back to line 150 and change the SPUT command back to a PSPUT command.
Lets add some text to the game.

```
120 #ROOM 14:#TEXTA 10,160,"BALLS"
190 #TEXTA 10,50,"READY":#VIEW:#COLOUR
200 PAUSE:#TEXTA 10,50,"     " 5 SPACE CHARACTERS
```

Line 120 prints the message BALLS at the top left hand side of the screen (10,160). Line 190 will print the message READY towards the bottom left of the screen. The message is then VIEwed. Line 200 causes the computer to wait for a keypress and then remove the READY message from the screen. Try RUNning the program.

The character set used by the TEXTA command was designed within the Designer program. It's time to print some numbers onto the screen.

```
130 LET BA=3
170 #PNUMA 36,140,1,48,BA
```

The variable BA in line 130 stores the number of balls left in the game. The PNUMA command in line 170 instructs the Supervisor to print a number onto the screen at position 36,140. The number will be 1 digit long and the character set will start at ascii code 0 (48). The number to be printed in stored in BA. Try RUNning the program.

Lets add a score.

```
290 #SCOM 2:LET SC=SC+10:#PNUMA 20,107,5,48,SC
```

This command line increases your score by 10 every time a block is removed from the screen. The score is then updated using the PNUMA command. The score is displayed at position 20,107. The score is 5 characters long, starting at ascii code 48. The number to be printed is held in SC. Try RUNning the program.

Lets make life a little more difficult, we will now detect for the ball moving past the bat.

```
240 #SYPOS 1,LENGTH(0,Y):IF Y<16 THEN GO TO 320
320 PAUSE 20:#TOGGLE 0:#TOGGLE 1:LET BA=BA-1
330 IF BA<>-1 THEN GO TO 170
```

The SYPOS command in line 240 will return into variable Y, the Y position of sprite 1 ( ball). If this is less that 16 then the ball has moved beyond the bat. Once the ball has moved beyond the bat, line 320 pause's the program and then removes sprites 0 and 1. The number of balls left is decremented by one. If there are any balls left (line 330) then the program will jump to line 170.

Lets finish off this simple demo by adding the final lines of program.

```
30 #INIT:#INKBLACK
300 IF SC=540 THEN GO TO 370
340 #TEXTA 110,40,"GAME OVER"
350 #VIEW:FOR 1=1 TO 3000:NEXT I
360 #INKBLACK:#CTEXT:GO TO 110
370 #TEXTA 110,40,"WELL DONE"
380 FOR 1=1 TO 50:#MOVER 2,63:#VIEW:NEXT I
390 GO TO 350
```

Line 30 will initialise all of the settings, and then the INKBLACK command will turn all of the palette colours to black. This will allow the computer to set up the screen without the user seeing anything. The COLOUR command in line 190 will redisplay all of the colours. The reason the INKBLACK command is added last is because, if you had any errors within your program, you would not be able to see the error message as all of the colours would be black. You can press the **f6** function key to return to the standard Sam palette settings.

Line 300 tests for the highest possible score indicating that the game is over. All of the other commands should be familiar to you by now. The reasoning behind the loop in line 380 is simple. The loop will allow the last block sprite to be completely removed from the screen. This loop is not needed, but its a nice touch. The MOVER command will move all of the sprites between sprite number 2 and sprite number 63.

Please feel free to experiment with this simple demonstration. To make life a little more difficult edit the SETINP command in line 100 to read SETINP 0,2,5,0,68,"00000001". Continuous movement, have fun !!

# PERMANENT SPRITES

As you may already know, the sprites are only updated on the screen whenever a VIEW command is issued. When the VIEW command is issued, all of the sprites are removed from the screen and then all of the sprites are then replaced back onto the screen in their new positions. Even if the sprites have not altered their position, the sprites still have to be removed and then replaced. This can cause the Supervisor to slow down if there are a large number of sprites on the screen.

In most cases, there will be a number of sprites that are displayed on the screen that will never actually move, they may include blocks, items to pick up, etc. Because these sprites never move, they do not need to be updated every time the VIEW command is issued. I therefore decided to introduce a special instruction to inform the Supervisor that this particular type of sprite (permanent sprites) do not need to be updated with all of the other (normal) sprites.

It should be noted that a permanent sprite is extremely restricted in what it can do. In fact, the only operation that may be performed on permanent sprites is removal. This is fine for games like Breakout, were the majority of the sprites are stationary. There are two Breakout demonstrations on the system disc, one using normal sprites, and one using permanent sprites. These demo's will allow you to gauge the effectiveness of using permanent sprites.

As a general rule, all of the sprites that are not going to move on the screen should be permanent sprites. Door sprites, which are discussed a little later in the manual, are ideal candidates for permanent sprites.

Permanent sprites will still have all of the collision attributes of normal sprites, they must be placed on a sprite plane just like all other sprites.

Permanent sprites are defined when you place them onto the screen. Normal sprites are placed onto the screen using the SPUT command. Permanent sprites are placed onto the screen using the PSPUT command.

Once a permanent sprite has been placed onto the screen, the only way of removing the sprite is using the TOGGLE command. Once a permanent sprite has been toggled off the screen, it will become a normal sprite again. If you toggled a permanent sprite off the screen, and then toggled the sprite back onto the screen, the sprite would not be permanent anymore. The only way of defining a permanent sprite is to use the PSPUT command, and the sprite will only remain permanent while it is on the screen.

Permanent sprites MUST be placed onto the screen BEFORE any other sprite is placed onto the screen. If there are already normal sprites on the screen, and you issue a PSPUT command, an error message will be displayed.

When normal sprites are placed onto the screen, you will not see them until you issue a VIEW command. Permanent sprites will appear on the screen immediately after issuing the PSPUT command and before the VIEW command. You will still need to use the VIEW command to see any normal sprites that may have been placed onto the screen.

# PLATFORM SPRITES

A Platform sprite is a sprite that will support another sprite. ie other sprites will walk on top of platform sprites. You will note that in DEMO1 there are a number of platform sprites used within the game. The main character within DEMO1 (froggy) can stand on the sprites that have been defined as platforms. If the platform is moving, froggy will move with the platform.

To define a sprite as a platform sprite, use the PLATFORM *sprite number* command.

Once a sprite has been defined as a platform sprite, it will act in a slightly different way to normal sprites.

1) Normal sprites will not detect collisions with platform sprites.
2) However, platform sprites will report collisions with normal sprites.
3) A sprite can only be defined as a platform sprite when the sprite is off the screen.
4) Door sprites can not be defined as platform sprites.
5) Once a sprite is defined as a platform sprite, the sprite will remain defined as a platform sprite until either an INIT or CLINK command is issued.

All of the sprites that are controlled by either the joy-stick or keyboard will *link* to a platform sprite, whenever the joy-stick/keyboard controlled sprite is standing on top of the platform sprite. This means that if the joy-stick/keyboard controlled sprite is standing on top of the platform sprite and the joy-stick/keyboard controlled sprite is stationary, then the sprite will move with the platform. You may, of course, move the joy-stick/keyboard controlled sprite while it is standing on top of a platform sprite.

Sprites that are NOT controlled by the joy-stick/keyboard, program controlled sprites, will NOT *link* to a platform sprite, whenever they are standing on top of a platform sprite. If the program controlled sprite is stationary, the program controlled sprite will remain stationary while the platform moves underneath the sprite. If the drop flag is set on the program controlled sprite, then as soon as the platform has passed under the program controlled sprites *feet,* the sprite will drop.

Platform sprites can be programmed to follow nodes. If a joy-stick or keyboard controlled sprite is standing on top of a platform sprite, then it will also follow the node route that the platform sprite is following.

As well as defining the platform sprites to move, you may also allocate a platform speed to the platform sprite, this is NOT the speed that the platform sprite will move, it is the speed that the sprite that is currently standing on the platform sprite will move. This feature will allow you top create conveyor belts. I think we had better have an example to make this a little clearer.

Sprite 10 is a platform sprite, it is stationary and has a platform speed of 2 defined. Sprite 2 is a joy-stick/keyboard controlled sprite that has been defined to move at a speed of 2 if it moves to the left or to the right.

If sprite 2 is standing on top of the platform sprite and sprite 2 is stationary, then sprite 2

will move to the right at a speed of 2 pixels/move. If you then move the joy-stick to the right, then sprite 2 will move to the right at a speed of 4 pixels/move ( 2 + 2 ). If the joy-stick is continually held to the right, then sprite 2 will continue to move at a speed of 4 pixels/move until sprite 2 leaves the platform sprite. As soon as sprite 2 leaves the platform sprite, then sprite 2 will start to move to the right at a speed of 2 pixels/move.

If while sprite 2 is still standing on the platform you moved the joy-stick to the left, then sprite 2 will stop moving altogether. This is because sprite 2 is trying to move at a speed of -2 and the platform speed is 2, therefore -2 + 2 = 0 and so the sprite does not move.

If the platform speed had been defined as speed 1 and sprite 2 had been defined as speed 2, then sprite 2 would have to following speeds.

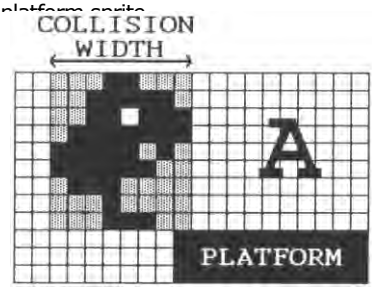
| | |
|---|---|
| Sprite 2 stationary | Sprite 2 would move right at a speed 1 pixel/move<br>Sprite speed (0) + Platform speed (1) = +1 |
| Sprite 2 joy-stick left | Sprite 2 would move left at a speed 1 pixel/move<br>Sprite speed (-2) + Platform speed (1) = -1 |
| Sprite 2 joy-stick right | Sprite 2 would move right at a speed 3 pixels/move<br>Sprite speed (+2) + Platform speed (1) = +3 |

These examples are for a platform sprite that is stationary, the platform sprite can also be moving, in the case of a moving platform the same speeds would be incurred, but the speeds would be relative to the speed at which the platform is moving.

The platform speed is defined using the PLATSPEED command. The speed may be a positive (moving to the right) or a negative (moving to the left) value. This value may be changed while the platform sprite is on the screen.

You may have noticed that within the *Introducing Nodes* section of the manual, we introduced the FEET command. This command allows you to reduce the collision edges of a sprite. This command can be extremely useful when working with platform sprites.

A sprite is classed as standing on a platform when at least 1 pixel of the sprites collision box is over 1 pixel of the platform sprite.



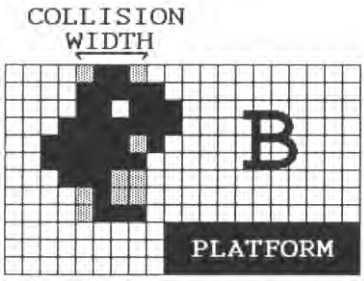

The sprite in the diagram (A) above is classed as being stood on the platform. This is because the right hand edge of the sprite is 1 pixel over the platform. In all of the following examples the we shall assume that the sprite has been programmed to drop.

This sprite will not drop. It does look a little strange, the sprites feet are not standing on the platform and so the sprite looks like it is walking in mid air.

To overcome this problem, the FEET command will effectively reduce the area of the collision box. If the above example if we used the command

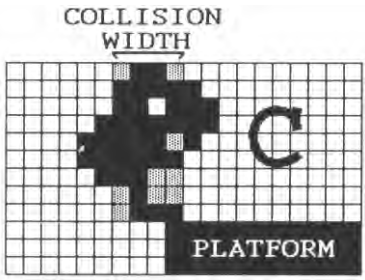
FEET 0,2,2

Assuming this was sprite 0. The command would have the following effect.




The collision box of the sprite has be reduced by a value of 2 pixels on each side of the sprite. In this case, the sprite would now drop if it was in this position because non of the collision box is over the platform.

The sprite would need to be in the following position before the sprite would not drop.




This looks far more realistic, the sprites feet are actually over the edge of the platform, and so, the sprite does not look like it is walking on air.

You may select different feet variables to every sprite. When you reduce the size of the collision box with the FEET command, the sprites collision detection will be reduced. This means that another sprite has to be inside of the sprites new collision box before a collision is detected.

The FEET command has three variables, the first variable is the sprite number you wish to apply the FEET command too. The next variable is the value to reduce the left hand side of the sprite by, the last variable is the value to reduce the right hand side of the sprite.

# DOOR SPRITES

Under normal circumstances, all of the sprites on the screen will pass over each other. This makes it very difficult to create doors, ie sprites that block other sprites from passing over them. It was therefore decided to create a special sprite called a door sprite.

A door sprite will block the travel of any sprite that can detect collisions in the plane that the door sprite occupies. This means that if a door sprite has been placed onto the screen on sprite plane 4, then any sprite that can detect collisions with sprite plane 4 will not be able to pass over the door sprite.

You may of course define some sprites to pass straight through the doors by disabling the detection of certain collision planes (see CRANE command).

Once a sprite has been defined as a door sprite, then the sprite will remain defined as a door sprite until either an INIT or CLINK command.

You may only define a sprite to act as a door sprite when the sprite is off the screen.

Door sprites have a number of limitations. Once a sprite is defined as a door sprite, it can not move and it can not fire missiles. In fact the only operation that may be performed on door sprites, is to remove them. This makes them ideal candidates as permanent sprites. Please refer to the discussion on permanent sprites.

Door sprites still have full collision detection, just like normal sprites. They will detect collisions with any sprite that comes into contact with them.

There is one problem associated with door sprites. If the door sprites are very narrow, and a sprite approaching them is travelling at a fast speed, then the sprite may be able to pass straight through the door sprite without stopping, this may be especially true of missile sprites. The solution to this problem, is to make door sprites wider. You do not have to fill the image in completely, you may leave a large blank column down one side of the sprite, this will not be seen, but the door will have a larger collision box, and so a fast moving sprite is more likely to collide with the door.

Another problem associated with door sprites, is missiles. If you have a missile firing sprite collide with a door, and the sprite then fires a missile, the missile may pass through the door, this is because the missile may be placed onto the screen at an offset that takes it past the door. The solution to this problem is identical to the solution to the last problem, make the door sprite wider.

Both of the solutions to the two above problems involve making the door sprites wider, this is no problem if you have placed the door sprite onto the screen as a permanent sprite, as the sprite is not updated. The bigger the sprites, the more time it takes to update them on every screen.

This is another reason why you should make all of the door sprites PERMANENT SPRITES. Permanent sprites will allow the Supervisor to execute faster, and every little bit of speed, no matter how small, helps !

# MISSILE SPRITES

You may define any number of sprites to fire missiles. Missile firing is a fairly complex subject, but if you take your time the results are worth it.

The first thing to do when defining sprites to fire missiles, is to decide which sprites are going to fire missiles. After making a list of all of the missile firing sprites, you should enter the Designer program and select the MISSILE option. You must create a missile information table for every IMAGE that will fire a missile. This is described within the *missiles* section of the manual. Please note you are defining the missile information for every IMAGE that will fire a sprite and NOT for every SPRITE.

After you have created all of the missile information within the Designer you may leave the Designer program. All of the rest of the missile information is programmed within the Supervisor.

The first thing to do within the Supervisor is to define a block of sprites to be used as missiles. This *block* of sprites must be a continuous range of undefined sprites. You may only create a single block of sprites, and so, all of the sprites that are going to be firing missiles will use sprites within this range.

The command to define a block of sprites is the MISSRANGE command. The format of the command is as follows

**MISSRANGE** *start block, finish block*

This command will define a number of sprites to act as missiles. For the purpose of this example we will issue a MISSRANGE 46,63 command. This means that all of the sprites within the range 46 to 63 will act as missile sprites. When you issue this command, non of the sprites within the range 46 to 63 must have been defined. If any of the sprites are defined, an error message will be displayed. Once a missile range has been defined you may not use any of the sprites within the range for any other purpose.

After defining the block of sprites to act as missiles, we must use another command for every sprite that is to fire missiles. This is the MISSILE command. The format of the MISSILE command is as follows.

**MISSILE** *Sprite number, distance, delay, low sprite, high sprite, plane, plane detect, explode animation.*

This command must be issued for every sprite that is to fire missiles. The command is described in detail in the *command guide* section of the manual.

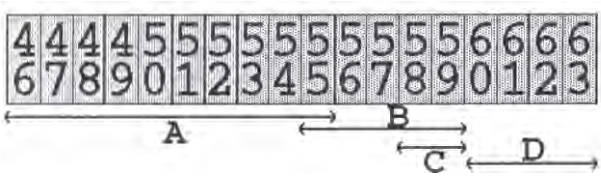
The *sprite number* is the number of the sprite that is going to fire the missiles.

The *distance* variable is the number of moves the missile will make before it is removed from the screen. This can be used to make the missile only have a short range.

The *delay* variable determines how quickly the sprite can actually fire missiles. The higher the number, the slower the sprite will fire missiles.

The *low sprite* and *high sprite* variables determine how many missiles the sprite can have on screen at once. The *low sprite* variable must contain a sprite number that is within the block of sprites that can fire missiles. In this example the *low sprite* variable must be a sprite number higher than 45. The *high sprite* variable must contain a sprite number that is within the block of sprites that can fire missiles. In this example the *high sprite* variable must be a sprite number less that 64.

It should be noted that a number of missile firing sprites can share the same range of missiles. Lets look at an example to explain this a little better.




The diagram above represents the block of sprites that have been defined as missiles. We have defined 4 sprites to fire missiles, sprites A, B, C and D.

Sprite A is defined to use missile sprites 46 to 55. Sprite B is defined to use missile sprites 55 to 59. Sprite C is defined to use missile sprites 58 to 59. Sprite D is defined to use missile sprites 60 to 63.

When a sprite is instructed to fire a missile, the Supervisor will look through the range of missile sprites for the sprite that will fire the missile, until it finds a missile sprite that is not currently displayed on the screen. If the Supervisor can not find a free sprite, then the fire instruction is cancelled. Once the Supervisor finds a free sprite, the image number of the missile, defined within the Designer, is linked to the missile sprite, and the missile sprite is placed onto the screen. The missile is then classed as being fired. The missile is now on the screen, and so the Supervisor will not try to use the missile sprite again until it is removed from the screen.

As we have said previously, a number of missile firing sprites can use the same missile sprites. In the above example sprite missile number 55 can be used as a missile by sprites A and B. Sprite missile numbers 58 and 59 can be used by both sprites B and C. Remember, even though the missile sprites can be shared, if the missile sprite is currently on the screen, it can not be used. If sprite B had fired all of its missile allocation (sprites 55 to 59) and all of the missile were still on the screen, then sprite C would not be able to fire a missile until missile sprites 58 or 59 were removed from the screen.

If there are a number of sprites firing missiles on the screen, then it may NOT be a good idea to set the *low sprite* and *high sprite* variables to the maximum of 46 and 63. This is because 1 missile firing sprite may take up all of the missiles and so there may be no missile sprites available when you wish to fire a missile. In the example, sprite A,B and D have a number of missile sprites dedicated for their own use. ie Sprite A has 46 to 54, Sprite B has 56 to 57, and sprite D has 61 to 63.

With careful planning, your sprites should always have missiles available to fire. There is one point left to make with regards to missile sprites. In the example, sprite D has 3

missile sprites allocated to it. If sprite D can fire missiles at a fast rate, then as soon as sprite D had fired all of its three missiles, it would not be able to fire any more missiles until at least one of the missile sprites is removed from the screen.

Getting back to the MISSILE command.

The *plane* variable determines in which plane the missile sprites will travel. This may be any of the sprites planes. ie 1,2,4,8,16,32,64,128.

The *plane detect* variable determines in which sprite planes the missiles detect collisions. If you do not select plane 128 then the missiles will travel through any scenery.

The *explode animation* variable determines how the missiles will react when they collide with another sprite, or a piece of scenery. You may opt for the missile to be simply removed from the screen, or the missile can be animated off the screen. If you use a value of 0 for this variable, the missile will be removed. If you wish the missile to be animated off the screen, you will need to specify an animation sequence between 1 and 64. The missile image will then be changed to the first image within the animation sequence, the Supervisor will then 'run-through' the animation sequence, and only when the animation sequence has been completed will the missile be removed from the screen. This is demonstrated in DEMO1, whenever a missile collides with another sprite, the missile will be animated through an explosion sequence.

The missile will only be animated off the screen, or removed from the screen, once the sprite number the missile has collided with is reported using the MISSHIT command. This command is described a little later in this chapter.

If the missile collides with a piece of scenery, the missile will not report the collision. It will automatically be either removed, or animated off the screen.

Once you have defined the missile command for every sprite that will fire missiles, we need to give the sprites a number of missiles to fire. The AMMO command will allocate a number of rounds of missiles to a particular sprite. ie AMMO 12,49. This command would allocate 49 missiles to sprite 12. This has nothing to do with sprite numbers, it is simply the number of missiles that may be fired. Once the sprite had fired 49 missiles, the sprite would not be able to fire any more missiles until another AMMO command is issued. There are two other commands dealing with ammunition, AMMADD and RAMMO.

The AMMADD command will add a number of rounds of ammunition onto the current number of rounds for the sprite.ie AMMADD 12,30 would add another 30 rounds of ammunition to sprite 12.

The RAMMO command will report the number of rounds of ammunition that the sprite is currently carrying.

A sprite can carry a maximum of 254 missiles. A sprite can be defined to have an unlimited amount of ammunition if you use the command AMMO 12,255. This would give sprite 12 an unlimited amount of ammunition.

Lets now look at what actually happens after a missile has been fired.

Once the missile has been fired by a sprite, the missile will travel at the speed, and in the direction, defined by the missile definition table (defined within the Designer).

If the missile does not hit a sprite, and hits the screen edge, the missile will be removed, ready to be fired again. The missile will not be animated off the screen.

If the missile does not hit a sprite, and does not hit the screen edge, it will be removed once it has travelled its maximum distance (defined within the MISSILE command). The missile will not be animated off the screen.

If the missile collides with a piece of scenery, and the missile can detect collisions with scenery (sprite plane 128), then the missile will either be removed, or animated off the screen, depending on how the *explode animation* variable was defined within the MISSILE command.

If the missile collides with another sprite, the missile will stop moving, the sprite the missile collided with will also stop moving. Nothing will now happen until the missile collision is reported to you using the MISSHIT command. Once the number of the sprite that the missile collided with is reported, the sprite that the missile hit will be free to move again. The missile will then either be removed, or animated off the screen, depending on how the *explode animation* variable was defined within the MISSILE command. It is up to your program to decide what action to take with the reported sprite number.

Missile collisions are reported using the MISSHIT command. This command should be included within the main loop of your program. **A** missile that is in collision with a sprite will not be removed from the screen until the collision has been reported. The format of the MISSHIT command is as follows

```
MISSHIT LENGTH(0,VAR)
```

This command will return a value into the variable specified within the LENGTH function. If the variable contains a value of 255, then there are no missiles in collision at the time the command was issued.

If the variable contains a value other that 255, then the variable contains a sprite number which has been hit by a missile. The missile that had collided with the reported sprite number, will then be automatically removed, or, animated off the screen. It is up to your program to decide what to do with the sprite that had been hit by the missile. If the sprite had been moving prior to being hit by the missile, then after the collision has been reported, the sprite will start to move again. Your program may decide to remove the sprite that was in collision with the missile, or the sprite may be animated off the screen, etc, etc.

You may define a missile explosion sound using the MISSOUND command. This sound will be used by all of the missiles whenever a collision is reported. Please refer to the *command guide* chapter for more details on this and other missile commands.

# JOY-STICK / KEYBOARD SPRITES

Within 99% of your programs you will have at least one sprite that is under control of the joy-stick or keyboard. The SCADs Supervisor will allow you to define any number of sprites to be controlled by the joy-stick or keyboard. In most cases, however, in practical terms there will be only one, or, possibly two sprites under joy-stick or keyboard control.

To define a sprite to be under joy-stick/keyboard control, we use the SETINP command. Full programming details for using this command can be found in the *command guide* section of the manual. The format of this command is as follows

SETINP *sprite number, input type, x speed, y speed. directions. "string information"*

The *sprite number* is the number of the sprite you wish to be controlled by either the joy-stick or keyboard. This sprite can not be a permanent sprite, a missile sprite, or a door sprite. The sprite must have already been defined using the LINK command.

The *input type* variable defines which method of input will control the sprite, this may be joy-stick1, keyboard 1, joy-stick 2, or keyboard 2. The keyboard inputs are defined within the Designer program.

The *x speed* and y *speed* variables define at which speeds the sprite will move.

The *directions* variable defines in which directions the sprite can move.

The *string information* string defines a number of attributes for the sprite. The *string information* consists of 8 characters enclosed within quotes. The layout of the *string information* is as follows

1) Define jump key

This option will allow you to specify an input that will make the sprite jump. The options available are, 0 - No Jump,1 - Up key, 2 - Fire key. If you specify a jump key, then you must also specify a jump sequence in character 2.

2) Define jump sequence

This option will allow you to specify a jump sequence for the joy-stick or keyboard controlled sprite. If you have used option 1 or 2 in the previous character, then a jump sequence MUST be entered within this character. The range of jump sequences are 1 to 8. If you enter a value of 0, then the sprite will not be able to jump. If you enter a value of 0 in this option, then you must also enter a value of 0 in the previous option (character 1).

3) Change direction jump

This option, if set, will allow the sprite to change direction whilst it is jumping. To allow the sprite to change direction, use a value of 1. A value of 0 will disable the sprite from changing direction whilst it is jumping.

4) define fire key

This option will allow you to specify an input that will make the sprite fire a missile. The options available are, 0 - No fire,1 - Up key, 2 - Fire key. If you specify either option 1 or 2, then the sprite should have been defined to fire a missile with the MISSILE command.

5) Set drop type

This option will allow you to specify 1 of 5 different types of sprite dropping. This should cover all of the types of games you wish to produce. A full description of the different types of drop are detailed a little further into this chapter. The options available are 0 - No Sprite Drop, 1 - Sprite Drop - No Left/Right Movement, 2 - Sprite Drop with Left/Right Movement, 3 - Sprint Drop When Joy-stick Left/Right, 4 - Sprite Drop When Joy-stick released.

6) Initial drop speed

This variable will allow you to specify the starting drop speed for a sprite. The starting drop speed is the speed that the sprite will initially start to move when the sprite is not standing on a platform. If you specify an acceleration speed ( character 7) then the sprite will increase in speed every time the sprite moves.

7) Acceleration speed

This variable will allow you to specify the acceleration drop speed of a sprite. Once the sprite has started to drop, the sprites speed will increase by the amount within this variable every time the sprite moves. If you set an acceleration speed of 0, the sprite will appear to glide slowly down the screen, provided the initial drop speed is set to 1 or higher.

8) Continuous move

This option will allow you to set the sprite to move continuously. To set this option use a value of 1. A value of 0 will disable this option. When a sprite is set to move continuously, the sprite will not stop moving until it hits a piece of scenery or the screen edge. This option is useful if you are creating a pacman type game.

Once a sprite has been set to be controlled by either the joy-stick or keyboard, the sprite will remain under control until either an INIT or CLINK command. Issuing a SPEED command will also clear the joy-stick or keyboard control.

You should not use the DROP command on a joy-stick or keyboard controlled sprite, as this command is incorporated into the SETINP command.

The Supervisor automatically alters the edge attributes for a sprite when you issue the SETINP command. When a sprite is first created using the LINK command, the sprite is defined to bounce off all window edges. This is not acceptable for a sprite that is under control of the joy-stick or keyboard. When the SETINP command is issued, the Supervisor alters the edge attributes so that the sprite will stop whenever the sprite hits the screen edge. You may, of course, alter this setting after the SETINP command by

issuing new XEDGE and YEDGE commands.

Lets now look at the different types of drop settings for the sprite that may be defined using the SETINP command.

The *drop type* for a sprite is set using character 5 within the *string information* of the SETINP command. There are 5 types of drop, and we shall look at each of the different types individually.

0) No drop.

The sprite will not drop at all. This setting would probably be used if you are creating a non platform type game. ie Pacman or Breakout. You should not use this option if you have defined the sprite to jump.

1 Drop - no left/right movement.

The sprite will drop whenever the sprite is not standing on a platform. The sprites drop speed will be calculated from the *initial drop speed* and *acceleration speed* settings within the *string information* section of the SETINP command. While the sprite is dropping, the sprite will not be able to move left or right. The sprite will simply drop straight down until it lands on another platform or the screen edge.

2) Drop - with left/right movement.

The sprite will drop whenever the sprite is not standing on a platform. The sprites drop speed will be calculated from the *initial drop speed* and *acceleration speed* settings within the *string information* section of the SETINP command. While the sprite is dropping, the sprite WILL be able to move left and right. The sprite will drop until it lands on another platform or the screen edge. While the sprite is dropping moving the joy-stick or keyboard left or right will allow the sprite to move left or right while the sprite is dropping.

3) Drop When joy-stick or keyboard moving left or right.

This option will allow the sprite to float above any platforms whenever the joy-stick or keyboard are selecting up. If the joy-stick or keyboard is moved to the left or the right, the sprite will start to drop. This option can be useful for creating Jetpack type games. This option can be very effective if you set a *drop acceleration speedto* 0, and the *initial drop speedto* either 1 or 2. Experiment !!

4) Drop - When joy-stick or keyboard released.

This option is similar to the previous option, except the sprite will only drop whenever the joy-stick or keyboard is released. Please see the comment on the previous option for details of uses.

The best way of understanding using the SETINP command is to experiment with the various settings. There is one type of game that is not covered by this command, and that is the Asteroids type game. In this game the space ship will rotate whenever the joy-stick or keyboard is selecting a left or right direction. There is a demonstration on the SCADs system disc showing you how to overcome this problem.

# COLLISION DETECTION

All of your programs will require collision detection. There are two collision detection systems incorporated into the SCADs Supervisor, missile collision detection and normal sprite collision detection. Missile collision detection is dealt with in the *missile* section of the manual.

## NORMAL COLLISION DETECTION

As you will know by now, all of the sprites that are placed on the screen travel in one of eight sprite planes. You may program a sprite to detect collisions with a number of sprite planes. A sprite is placed into a plane using the SPUT or PSPUT commands. The last variable used within the SPUT and PSPUT commands determines in which plane the sprite will travel. ie SPUT 10,100,130,32 would cause sprite 10 to be placed onto the screen at coordinates 100,130 and the sprite would travel in plane 32.

When the sprite is first initialised using the LINK command, the collision detection is automatically set to detect collisions with all sprite planes. You may use the CPLANE command after the sprite has been initialised to customise the sprites collision detection so that the sprite will only detect collisions within certain planes.

Lets look at an example to explain this a little better.

| Sprite | Travel .lane | Plane detection | Detect collisions |
|---|---|---|---|
| A | 1 | 6 | B + C |
| B | 2 | 8 | D |
| C | 4 | 15 | A + B + C + D |
| D | 8 | 32 | None |

This table shows how the collision detection may be set up for 4 sprites. The second column of the table informs you of the plane in which the sprite is travelling. The third column informs you of which planes the sprite can detect collisions in. The last column informs you of the sprites that the sprite will detect collisions with.

The plane detection variable in the third column would have been defined by using four CPLANE commands. CPLANE A,6: CPLANE B,8: CPLANE C,15: CPLANE D,32

It can be seen from this example, that sprite A can detect collisions with sprite B, but sprite B can not detect collisions with sprite A. Complex collision detection can be set up using this system.

There are two commands within the Supervisor dealing with collision detection.

**HIT** *sprite number,* **LENGTH** *(0,varl)*
**NEXTHIT LENGTH** *(0,var2)*

The first command, HIT, will report the number of sprites that are in collision with the specified sprite number. ie HIT 10,LENGTH(0,nh) will store in variable nh the number of sprites that Spriten collision with sprite 10 at the time the command was issued. If there

were no sprites in collision with sprite 10 at the time the command was issued, nh would contain 0.

Please remember that if there was a sprite in collision with sprite 10, but sprite 10 could not detect collisions in the plane that the other sprite was travelling in, then the collision would not be reported.

If there was a sprite or sprites in collision with sprite 10, then the Supervisor will create a list of all of the sprites that were in collision at the time the HIT command was issued. Remember, the HIT command will only report the number of sprites that were in collision, it will not report the sprite numbers of the sprites that are in collision.

To find out the sprites that are in collision, we use the NEXTHIT command. The NEXTHIT command will look at the list of sprites in collision, the list was created by the HIT command, and report the first sprite number in the list. If the NEXTHIT command is then issued again, the next sprite number in the list is reported. The NEXTHIT command will keep reporting sprite numbers until the list is empty. Once the list is empty a value of 255 will be returned.

It is important to remember that the list of sprites numbers that the NEXTHIT command reports, is only a list of sprites numbers that were in collision with the sprite specified within the HIT command. The list is only accurate until another MOVE command is issued.

If you issue another HIT command, then the previous list will be cleared and a new list calculated.

As an example, there are two sprites in collision with sprite 10, sprite 4 and sprite 17. If we issue the command HIT 10,LENGTH(0,AT). AT would now contain a value of 2. ie 2 sprites in collision with sprite 10. If we now issued a NEXTHIT LENGTH(0,SP) command, SP would contain a value of 4. If we issued the NEXTHIT LENGTH(0,SP) command again, SP would now contain a value of 17. If we issued the NEXTHIT LENGTH(0,SP) command for the third time, then SP would contain a value of 255 (no more collisions to report).

The list of sprites was created at the time when the HIT command was first issued, if, after the command was issued and before the NEXTHIT command was issued, sprites 4 and 17 had moved away from sprite 10 so that they were not colliding, the NEXTHIT command would still report sprite numbers 4 and 17 (even though they are not in collision anymore).

If in the above example, we issued a HIT 23,LENGTH(0,AT) command after the first NEXTHIT command, then the list of sprites in collision with sprite 10 would be cleared, and a new list of sprites in collision with sprite 23 would be created. Therefore sprite 17 would never have been reported as being in collision.

There is only room in memory for 1 sprite collision list, and this list can only contain information about all of the collisions for 1 sprite. In practice, your program will probably only be testing for collisions with your main character sprite, the sprite controlled by either the joy-stick or the keyboard, so this should not cause too many problems. When you are creating your game, the main loop within your game should contain as few instructions as possible. Try to only use one HIT command within the main loop, and your programs will execute faster.

# PRINTING TEXT AND NUMBERS

The SCADs Supervisor has facilities to allow you to print proportional text onto the screen. As you should know by now, the Supervisor uses a two screen system for displaying sprites. The problem with printing text, is that Sam BASIC will only print text onto one screen. If, after printing text using Sam BASIC commands, you issue a VIEW command, the text that was printed onto the screen disappears, the text will only reappear after another VIEW command. This will cause the text to flicker when your programs are running.

The Supervisor has a number of commands dedicated to printing text onto the screen, the text that is printed will be printed proportionally. Please refer to the introductory section on text at the start of the manual for more details on proportional text.

There are two commands for printing text, and two commands for printing numbers.

## *PRINTING TEXT*

Text is printed using the TEXTA and TEXTB commands. The difference between the two commands, is that, TEXTA will print text from character set 1, and TEXTB will print text from character set 2.

The format of the TEXTA and TEXTB commands is as follows

**TEXTA** *x coord, y coord, "message"*

The coordinates refer to where the BOTTOM LEFT hand corner of the first character within the message will be placed.

The text will not be printed when this command is issued, instead the text is stored in a buffer. The text will be displayed after the first VIEW command. The text/number buffer for each screen is 200 bytes long. This means that you will only be able to print up to 200 characters before a VIEW command must be issued. The text is copied into both text/number screen buffers at the same time, and so the text will not flicker when you swap screens using the VIEW command.

If any of the text is printed over the right hand edge of the screen, the text will not be printed. The text does not automatically print on the next line.

You may clear the text/number buffers using the CTEXT command. This command is described in more detail in the section on printing numbers.

## *PRINTING NUMBERS*

Numbers are printed using the PNUMA and PNUMB commands. The difference between the two commands, is that, PNUMA will print text from character set 1, and PNUMB will print text from character set 2.

The format of the PNUMA and PNUMB commands is as follows

The coordinates refer to where the BOTTOM LEFT hand corner of the first character within the number will be placed.

The *number of digits* variable determines how may digits of the number will be printed. The variable may contain a number between 1 and 5.

The *code character 0* variable should normally be 48. This is the ascii character code for the character '0'. If you use any number other than 48, then the Supervisor will print the number treating the specified character code as digit '0'. This will allow you to create a number of different style numbers within 1 character set. ie You have created, within the Designer, two different styles of numbers, the first style starting at character 48, and the second style starting at character 97. By altering the *character code 0* variable from 48 to 97, you would be able to print onto the screen a different style of numbers.

The *number* variable will contain the number you wish to print. This must be a positive integer number in the range 0 to 65535.

The number will not be printed when this command is issued, instead the number is stored in a buffer. The number will be displayed after the first VIEW command. The text/number buffer for each screen is 200 bytes long. This means that you will only be able to print up to 200 characters before a VIEW command must be issued. The number is copied into both text/number screen buffers at the same time, and so the numbers will not flicker when you swap screens using the VIEW command.

If any of the number is printed over the right hand edge of the screen, the number will not be printed. The number does not automatically print on the next line.

You may clear the text buffers using the CTEXT command. This command can be extremely useful for speeding up your programs. Lets look at how this can be achieved.

Consider a score that is updated on every screen. There are two text/number buffers, one for screen A and one for screen B. When you issue a PNUMA command the score is stored in both of the screen buffers. Lets the first number to be printed is 1. Therefore the character 1 is stored in both screen buffers. Now lets issue a VIEW command, screen A is displayed, and so the buffer for screen A is printed and the buffer is cleared. Now lets issue another PNUMA command. This time we will print the number 2, therefore the character 2 is added to both screen buffers. Screen buffer A has been cleared and so the character 2 will be the only character in the buffer. However, screen B has not been displayed yet, and so the buffer still contains character 1. Character 2 is then added onto the screen buffer for screen B, therefore the buffer now contains two characters 1 and 2. These characters are representing a score and so they will both be printed in the same position. If we now issue another VIEW command. The Supervisor will print the first character in the buffer, ie 1, and then it will print the second character in the buffer, ie 2. The character 2 is written over the top of character 1. This is clearly a waste of time.

If you issue a CTEXT command before you issue a PNUM or TEXT command, the Supervisor will be able to work that little bit quicker. To achieve the fastest possible speed, try to only update any scores, once within your main loop.

# USING SOUND EFFECTS

There are three methods for playing sound effects using the Supervisor. The three methods use three separate commands, AUTOSOUND, SCOM and NOTE. We shall look at the three different methods separately.

## *AUTOSOUND*

This command will allow you to link a predefined sound effect to a specific sprite action. The sound effect must have already been created using the Designer program. Please refer to the AUTOSOUND command within the *command guide* section of the manual.

## *SCOM*

This command will play one of the predefined sound effects created using the Designer program. This command will be useful for special effects such as animating the sprite off the screen.

This command will allow you to add sound effects for events that are not covered by the AUTOSOUND command. Further details can be found in the *command guide* section of the manual.

## *NOTE*

This command will allow you to play any sounds that have not been defined within the Designer. This command is useful for two purposes.

1) If you did not have enough room within the Designer program to allow you to create all of the sounds you wished too.

2) If you wish to create sounds which have variable tone and octave, which the Designer will not allow. This command will allow you to create music by reading tone and octave data and playing them through the NOTE command.

If you are going to play music using the NOTE command, you will need to take a great deal of care in getting the timing of your notes correct. This will require a fairly complex BASIC routine, which I haven't the space to describe here.

# AMMADD

## AMMADD *sprite number, rounds*

AMMADD 34,80

COMMAND: Allocates more ammunition rounds for a missile firing sprite. The sprite specified within the command must be defined to fire missiles. The *rounds* variable will be added to the number of bullets left for that particular sprite. A sprite can carry up to 254 rounds of bullets. Please see the AMMO command for more details.

LIMITS: Sprite number ( 0 - 63 )
Rounds ( 1 - 254 )

ILLEGAL USE: Door sprites
Missile sprites
Undefined sprites
Non missile firing sprites

Associated keywords: AMMO, **RAMMO, MISSILE, MISSRANGE, MISSHIT, MISSOUND**

# AMMO

## AMMO *sprite number, rounds*

AMMO 23,255

COMMAND: Define a sprite to fire a fixed number of bullets. Once you have defined a sprite to fire missiles with the MISSILE command, you must then specify the number of rounds of bullets the sprite can fire. Once the sprite has fired all of the missiles you have specified, then the sprite will not be able to fire any more missiles until you restock the sprite with missiles, using either the AMMO or AMMADD commands.

The sprite can carry up to 254 missiles. The sprite can be programmed to have unlimited bullets by using a value of 255 for the *rounds* variable. Using a value of 0 for the *rounds* variable will stop the sprite firing missiles.

LIMITS: Sprite number ( 0 - 63 )
Rounds ( 0 off, 1 - 254, 255 unlimited missiles )

ILLEGAL USE: Door sprites
Missile sprites
Undefined sprites
Non missile firing sprites

Associated **keywords: AMMADD**, **RAMMO**, **MISSILE**, **MISSRANGE**, **MISSHIT**, **MISSOUND**

# ANIMATE

## ANIMATE *sprite number, sequence*

```
ANIMATE 10,4
```

COMMAND:Sets up an animation sequence for a specific sprite. The sequence number must be in the range from 0 to 64. Setting the animation sequence to 0, will cancel the animation for the selected sprite.

The image number that is currently linked to the selected sprite, must be contained within the animation sequence. ie If the sprite is currently linked to image number 50, then the animation sequence must contain image number 50 somewhere within the sequence definition.

If you are going to use the MERGE command within your programs, then you should take care to ensure that the new image is within the current animation sequence. If the new merged image is outside of the animation range, then should either alter the animation sequence, or cancel the animation. This must be completed before the next move command.

LIMITS: Sprite number ( 0 - 63 ),
Sequence number (0 - off, 1 - 64)

ILLEGAL USE: Permanent sprites
Sprites outside windows
Missile sprites
Undefined sprites

Associated keywords: None

# ANIMOFF

## ANIMOFF *sprite number, sequence, direction, delay*

```
ANIMOFF 3,23,4,2
```

COMMAND:Force the specified sprite to run through a single animation sequence and then remove itself from the screen. This command is very useful when you wish to remove a sprite from the screen, instead of simply making the sprite disappear, the sprite may be displayed through one animation sequence and then disappear.

You may, for example, have defined an animation sequence of an explosion effect. After using this command, the sprite would be changed into the first image within the explosion, on successive MOVE commands the sprite would display each of the explosion images, until the last image had been displayed. The sprite would then be removed from the screen.

Unlike normal animation commands, the animation sequence you select for the sprite does not have to use images that are the same size as the image the sprite is currently using. If the image within the animation sequence is a different size to the current image, then the new image will be centred around the old image.

Once you have issued this command, the animation, and the sprite being removed from the screen is handled automatically by the Supervisor. Although the sprite image, animation sequence and possibly the size has been altered for the sprite, if you placed the sprite back onto the screen, the original image, animation sequence and size will have been restored.

If the sprite is currently moving on the screen, the sprite will stop moving. As soon as this command is issued, the sprite will stop detecting collisions with other sprites. If the sprite was in collision with a missile, the missile will be removed from the screen.

The *direction* variable specified within the command will allow you to select a particular animation range within an animation sequence. It has nothing to do with the direction the sprite will travel.

The *delay* variable will allow you to specify the number of times the MOVE command is called before the next frame of the sprite is updated. It is advised that this value is kept between 1 and 3.

Once a sprite has been removed from the screen, you must use either SPUT or PSPUT to place the sprite back onto the screen again. You may not toggle the sprite back onto the screen.

LIMITS: Sprite number ( 0 - 63 )
Animation sequence ( 1 - 64 )
Direction ( 1 - 9 )
Delay ( 1 - 255 )

ILLEGAL USE: Sprite off screen
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites
Undefined sequence/direction

Associated keywords:TOGGLE

# AUTOSOUND

## AUTOSOUND *sprite number, sprite action, sound number*

```
AUTOSOUND 23,4,13
```

COMMAND: Defines a sprite to automatically play a predefined sound effect, whenever the sprite performs a defined action. You may assign up to eight different sound effects to the first 32 sprites. You can not define any sound effects to sprites 32 to 63.

To clear an AUTOSOUND repeat the command and use a defined sound of 0.

The eight sprite actions are

| | |
|---|---|
| 0 | Fire a missile |
| 1 | Jump |
| 2 | Land after dropping /jumping |
| 3 | Walk |
| 4 | Bounce off Scenery |
| 5 | Bounce off another sprite |
| 6 | Bounce off window edge |
| 7 | Falling |

LIMITS: Sprite number ( 0 - 31 )
Sprite action ( 0 - 7 )
Defined sound ( 0 - off, 1 - 32 )

ILLEGAL USE: Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated keywords: NOTE, **SCOM**, **SKILL**

# BOUNCE

## BOUNCE *sprite number*

```
BOUNCE 45
```

COMMAND: Defines a sprite to automatically bounce off other sprites. All sprites will bounce of scenery without using this command. There are occasions when a sprite will not bounce when they collide with another sprite, if all of the possible bounce directions would cause the sprite to still bedroppcollision, then the sprite will keep travelling in the original direction of travel.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Door sprites
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated keywords: None

# CLINK

## CLINK *sprite number*

CLINK 23

COMMAND:Clears all of the information on the defined sprite. Platform and Door links will also be cleared. This command will allow you to reprogram a sprite. We suggest you use this command as little as possible, if at all, an efficient program should define all of the sprites at the start of the program, and then leave the definitions intact.

This command is the only command (with the exception of INIT) that will clear a platform or door sprite.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Sprite on screen
Missile sprites

Associated keywords:INIT, LINK

# COLOUR

## COLOUR *no variables*

COLOUR

COMMAND:Sets the computers palette to the palette defined within the SCADs Designer colour option.

LIMITS: None

ILLEGAL USE: None

Associated keywords:INKBLACK

# COPYSCREEN

COPYSCREEN *no variables*

COPYSCREEN

COMMAND:Copies all of the graphics from the main screen to the secondary screen. This command may be useful, if used in conjunction with the MAINSCR command, if you wish to draw lines and circles using Sam BASIC. You can then copy the screen to the second screen. This will eliminate flickering when using multiple VIEW commands.

LIMITS: None

ILLEGAL USE: None

Associated keywords:MAINSCR, **VIEW**

# CPLANE

CPLANE *sprite number, plane*

CPLANE 177

COMMAND:Determines in which collision planes the sprite will detect collisions. On initialising the sprite with the LINK command, the collision plane variable is set to 255. ie detect collisions on every plane. You may select to detect collisions on more than one plane by using the following method.

| | |
|---|---|
| +1 | First plane |
| +2 | Second Plane |
| +4 | Third Plane |
| +8 | Fourth Plane |
| +16 | Fifth Plane |
| +32 | Sixth Plane |
| +64 | Seventh Plane (Nodes plane) |
| +128 | Eighth Plane (Centreground scenery Plane) |

By adding up the plane numbers on the left hand side of the table, we can determine a Plane variable. In the example above, Plane variable 177 will detect collisions in planes 128+32+16+1.

If you do not include plane 128 within your calculations, then the sprite will pass straight through any centreground scenery that is placed onto the screen. Similarly, If you omit plane 64, then the sprites will not recognise any nodes that may have been defined within the screen.

LIMITS: Sprite number ( 0 - 63 )
Plane ( Centreground

ILLEGAL USE: Sprites outside window
Missile Sprites
Undefined sprites

Associated keywords:SPUT, **PSPUT, MISSILE**

# CTEXT

## CTEXT *no variables*

CTEXT

COMMAND:Clear the 'to be printed' text buffer. Please see the text section within the Supervisor section of the manual.

LIMITS: None

ILLEGAL USE: None

Associated commands:PNUMA, **PNUMB, TEXTA, TEXTB, VIEW**

# DJCLEAR

## **DJCLEAR** *sprite number, flag*

DJCLEAR 45

COMMAND:Stops the Supervisor from clearing the jump and drop speeds when the sprite is removed from the screen.

Normally, when a sprite is removed from the screen, a number of settings for the sprite are cleared. The Supervisor takes the following actions.

1) The sprites drop speed is restored to the initial drop speed.

2) If the sprite was jumping, the jump is halted.

Under normal circumstances, the above actions taken by the Supervisor will allow the sprites to act consistently. ie If you removed a sprite from the screen that was jumping, if you replaced the sprite back onto the screen at a later time, then you would not want the sprite to continue jumping as this would look extremely strange.

However, there are circumstances when you will require the sprite to continue with the jump or drop. In platform games, the main character sprite may be allowed to jump off the screen edge and into the next room. When the new ROOM command is issued, the sprite is toggled off the screen. Therefore under normal circumstances the sprites jump

would be cleared. This will look strange when the sprite is placed into the new room.

Using this command will inform the Supervisor not to clear the sprites jump or drop speeds whenever the sprite is removed from the screen.

The flag variable specifies wether on not the jump and drop settings will be cleared. A value of 0 turns the command off, and restores the resetting of the jump and drop settings. A value of 1 turns the command on, a informs the Supervisor not to reset the drop and jump settings.

The may be times when you require the jump and drop settings to be restored even though the DJCLEAR command has been issued. ie at the end of a game. In these circumstances, issuing a RESTDJ command will clear the settings. This will not affect the DJCLEAR command, and any further jumps or drops will not be cleared.

LIMITS: Sprite number ( 0 - 63 )
Flag ( 0 - off , 1 - on )

ILLEGAL USE: Undefined sprites
Permanent sprites
Door sprites
Missile sprites

Associated keywords:RESTDJ

# DOOR

## DOOR *sprite number*

DOOR 45

COMMAND:Sets the sprite to act as a door. Door sprites can not move, they are used purely to block any sprite from passing through them. See a further description of door sprites elsewhere in this manual. Once a door sprite has been defined as a door sprite, it will remain defined throughout the whole of the game, or until a CLINK or INIT command is encountered.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Sprite is on screen
Already defined as a door
Already defined as a platform
Sprite is outside of window
Missile sprites
Undefined sprites
Sprite has speed

Associated **keywords:PLATFORM, INIT, CLINK**

# DROP

## DROP *sprite number, initial drop speed, acceleration speed*

```
DROP 43,1,1
```

COMMAND:Defines the sprite to drop whenever it is not standing on a piece of centreground scenery or a platform. The sprite will start to drop as soon as it 'clears' the piece of scenery it was walking on. The sprite will start to drop at the initial drop speed, and then increase its speed by the acceleration value, every move.

The sprite will drop until it lands on the window edge, another piece of scenery or a platform sprite. The sprite will always land perfectly. ie it will not stop dropping half way through a piece of scenery.

If the sprite is going to be controlled by either the joy-stick or keyboard then you should not use this command, but use the SETINP command instead. The keyboard and joy-stick commands will allow you to set the drop speed.

If the sprite has been set to drop, and the sprite is under node control, then the drop will work in a slightly different fashion. If the sprite is currently on a node, then the sprite will not drop. This has been included so that you may program the node to act as a ladder. A sprite under node control will only drop if it is travelling horizontally, if the sprite is travelling diagonally, the sprite will not drop. This will allow you to program the sprite to travel up stairs.

If you wish to cancel the DROP command, use a value of 0 for both the initial drop speed and acceleration speed variables.

It is a good idea to use the DROP command for any sprites that are going to jump.

LIMITS: Sprite number ( 0 - 63 )
Initial speed ( 0 - 15 )
Acceleration speed ( 0 - 15 )

ILLEGAL USE: Door sprites
Permanent sprites
Sprites outside of windows
Missile sprites
Undefined sprites

Associated keywords:SETINP, **JUMP, AUTOSOUND**

# FEET

## FEET *sprite number, left foot, right foot*

```
FEET 4,4,4
```

COMMAND:This command will allow you to reduce the collision detection box around the sprite. The *left foot* and *right foot* variables will reduce the left and right hand side of the sprite by the specified number of pixels.

This command is extremely useful for sprites that walk along platforms. By reducing the collision box area, the sprite will drop from the platform earlier. This will look more effective. If you do not use the FOOT command, the sprites, when they get very near to the edge of the platform, or scenery, may look like there walking on thin air. Missile collision detection will also take into account the value you have set for the sprites feet. Collisions will not register until the missile is inside the feet area of the sprite. Experimenting with this command is the best way of understanding how it works.

LIMITS: Sprite number ( 0 - 63 )
Left foot ( 0 - 15 ) (Always positive)
Right foot ( 0 - 15 ) (Always positive)

ILLEGAL USE: Door sprites
Permanent sprites
Sprites
outsdemo1indow
Missile sprites

Associated keywords:PLATFORM

# FILE

## FILE *"filename"*

```
FILE "demo1"
```

COMMAND:Loads a drawing data file created using the Designer in to the SCADs Supervisor memory. Once a file has been loaded into memory, it will stay in memory, until another file is loaded.

DO NOT include the .DRW filename extension, this will be added automatically by the Supervisor. This command should be one of the first commands used within your programs. You do not need to load the file every time the file is RUN. To get around this problem, your program should look like this.

```
10 GOTO 30
```
*When the program is run it skips the FILE command, and goes straight to the INIT.*

| | | |
|---|---|---|
| 20 | FILE "DEMO1" | *You should run the program for the first time by either RUN 20, or getting the DOS to auto-run line 20 when the program loads.* |
| 30 | INIT | *This command is important, as it will initialise the Supervisor, ready to run your program.* |
| 40 | | *The rest of your program.* |

LIMITS: Filename length ( 1 - 6 characters )

ILLEGAL USE: None

Associated keywords:INIT

# FIRE

## **FIRE** *sprite number*

```
FIRE 10
```

COMMAND:Instructs a sprite to fire a missile. The sprite must have already been defined to fire missiles using the MISSILE command.

A missile will only be fired if all of the following conditions are met,

1) The sprite is on the screen.

2) The image the sprite is currently displaying has been defined to fire missiles within the Designer program.

3) A missile is free within the sprites missile range.

4) The delay between firing missiles, for the sprite, is zero.

Once the missile has been fired, it will act in exactly the same way as a missile fired using the joy-stick/keyboard.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Door sprites
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated **keywords:ADDAMM, AMMO, RAMMO, MISSILE, MISSRANGE, MISSHIT, MISSOUND, SETINP**

# FULLCLEAR

## FULLCLEAR *no variables*

FULLCLEAR

COMMAND:Removes all of the sprites that are displayed on the screen and clears the full size screen, including any panel that may have been loaded. The sprite coordinates are not lost, and so they can be toggled back on to the screen if you desire. If you TOGGLE permanent sprites back onto the screen they will not be classed as permanent.

LIMITS: None

ILLEGAL USE: None

Associated keywords:ROOM, **PARTCLEAR, FULLWIPE, PARTWIPE, WCLS**

# FULLWIPE

## FULLWIPE *no variables*

FULLWIPE

COMMAND:Removes all of the sprites that are displayed on the screen, but does not clear the screen. Any panel, and any room, that is currently being displayed will be kept intact. The sprite coordinates are not lost, and so they can be toggled back on to the screen if you desire. TOGGLEing permanent sprites back onto the screen will cause them to become normal sprites.

LIMITS: None

ILLEGAL USE: None

Associated keywords:ROOM, **PARTCLEAR, FULLCLEAR, PARTWIPE, WCLS**

# GETINP

## GETINP *input type, length( 0, var )*

GETINP js,LENGTH(0,D1)

FUNCTION:Returns the current position of the specified input type into the variable defined within the LENGTH command. The value used by the *input type* variable determines which input is read.

| *Input type* | Input read |
|---|---|
| 1 | 1st Joy-stick |
| 2 | 1st Keyboard definition (Designer) |
| 3 | 2nd Joy-stick |
| 4 | 2nd Keyboard definition (Designer) |

The number returned by the function will be one of the following.

| | |
|---|---|
| 1 | Up |
| 2 | Up-Right |
| 3 | Right |
| 4 | Down-Right |
| 5 | Down |
| 6 | Down-Left |
| 7 | Left |
| 8 | Up-Left |
| 9 | Centred |
| +32 | Fire pressed |
| 255 | Illegal direction 2 and 4 types only |

A value if 255 will be returned into the variable specified within the LENGTH command only if an illegal keypress has been entered, ie left and right pressed together.

LIMITS: Input type ( 1 - 4 )
var ( any predefined legal sam variable)

ILLEGAL USE: None

Associated keywords:SETINP

# HIT

## HIT *sprite number, length( 0, var )*

```
HIT 56,LENGTH(0,hs)
```

FUNCTION:Reports the number of sprites that are currently in collision with the sprite specified in the command. The number of collisions will be returned in the variable specified within the LENGTH function.

This command will only report the actual number of sprites that are in collision, it will not report the sprite numbers of the sprites that are currently colliding.

Every time this command is used, a list of all of the sprites that are in collision is made.

To find the actual sprite number of the sprites that are in collision, you will need to use the NEXTHIT command.

A sprite is only classed as being in collision when the plane detect of the sprite you are testing, matches the plane in which a sprite is travelling. Please refer to the discussion on collision planes elsewhere in this manual.

This command will not detect collisions with either platform sprites, scenery or missiles. There are a number of special commands when using missiles.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Sprite off screen
Sprite outside window
Missile sprites
Undefined sprites

Associated **keywords:NEXTHIT**, **MISSHIT**

# INIT

## INIT *no variables*

INIT

COMMAND:Initialises all of the Supervisor settings. This command MUST be used at the start of you programs. All settings that have been previously made will be cleared. The sound manager will be reset and any sounds that were currently playing will be stopped.

The command will also change both of the Designer screens to mode 4. This is useful if you wish to edit your program in mode 3. You will not need to alter the mode before you run your program.

The drawing data that has been loaded using the FILE command will be kept intact.

This command should always be issued after a NEW command, failing to execute this command may result in an *screen already open* error message being displayed.

If ever either of the error messages *screen already open* or *screen not open* are displayed, you should issue the INIT **command.**

LIMITS: None

ILLEGAL USE: None

Associated commands:CLINK

# INKBLACK

## INKBLACK *no variables*

```
INKBLACK
```

COMMAND:Reset all of the computers palette to black. This command can be useful for setting up your sprites or text, without them being displayed on the screen.

Do not use this command when first creating your program, as you will not be able to read any error messages that may be displayed.

This command does not erase the colours settings that have been loaded using the FILE command.

LIMITS: None

ILLEGAL USE: None

Associated keywords:COLOUR

# JUMP

## JUMP *sprite number, jump definition*

```
JUMP 44,3
```

COMMAND:Informs a sprite to immediately jump following the jump definition that has been defined within the design program.

It is advised that the DROP command is used so that after the sprite has finished jumping, it will not be stood in mid air. The DROP command need only be used once at the start of the program.

LIMITS: Sprite number ( 0 - 63 )
Jump definition ( 1 - 8 )

ILLEGAL USE: Sprites off screen
Door sprites
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated commands:DROP

# LINK

## LINK *sprite number, image number*

LINK 10,54

COMMAND:Used to attach a drawing image to a sprite. This command must be used before any other command can be used on the sprite, failure to use this command will result in an Undefined sprite error. The images should all have been defined within the SCADs Designer package.

If you are going to define the sprite to be animated, then the image you select for the sprite must be one of the images from the animation sequence.

You can not link any sprites defined as missiles to a sprite image. This is handled automatically by the Supervisor.

Once a sprite image has been linked to a sprite, the link can not be cleared, except by using either the CLINK or INIT commands.

LIMITS: Sprite number ( 0 - 63 )
Image number ( 0 - 255 )

ILLEGAL USE: Sprite already defined
Image not defined

Associated keywords:CLINK, **INIT**, **MERGE**

# MAINSCR

## MAINSCR *no variables*

MAINSCR

COMMAND:Display the main screen. This command can be useful if you wish to draw lines and circles on the screen using the Sam BASIC commands. This command must be used before you attempt to use any of the Sam BASIC graphics commands. After using the Sam BASIC commands, you should use the COPYSCREEN command to copy the graphics to the SCADs Supervisor secondary screen. This will eliminate flicker when issuing multiple view commands.

LIMITS: None

ILLEGAL USE: None

Associated keywords:VIEW, **WCLS, COPYSCREEN**

# MISSHIT

## MISSHIT *length( 0, var )*

MISSHIT LENGTH(0,m1)

FUNCTION:Used to report any sprite numbers that have been hit by missiles. Once a sprite number has been reported, the missile will perform one of two operations. If you have not defined an explode animation sequence within the MISSILE command, then the missile sprite will simply be removed from the screen.

If you have defined an explode animation sequence, the missile will change to the first image used within the animation sequence, the animation sequence will step through image by image until the last image has been displayed. The sprite will then be removed from the screen.

Once a missile collision has been reported, and either the missile sprite has been removed, or the missile sprite is stepping through the explode animation sequence, the sprite that was hit by the missile is then free to move again.

If the MISSOUND command has been used, the missile explosion sound effect will be heard.

It is down to your program to decide what to do about the sprite that has been hit by the missile. As mentioned above the sprite number is returned in the variable you have defined within the LENGTH command.

If there are currently no missiles in collision when the MISSHIT command is execute, a value of 255 will be stored in your defined variable.

Within one loop of your main game program, there may be a number of sprites that collide with missiles. In order to detect all of the collisions, you should execute the MISSHIT command until a value of 255 is returned.

You can not detect missile collisions with the HIT command.

LIMITS: None

ILLEGAL USE: None

Associated keywords:ADDAMM, **AMMO**, **RAMMO**, **MISSILE**, **MISSRANGE**, **MISSOUND**, **FIRE**, **SETINP**

# MISSILE

## MISSILE *sprite number, distance, delay, low sprite, high sprite, plane, plane detect, explode animation*

MISSILE 0,30,5,40,50,2,127,0

COMMAND:Defines a sprite to fire missiles. This command will allow you to program a sprite to automatically fire missiles whenever the FIRE command is used, or whenever a joy-stick/keyboard controlled sprite has the necessary switch set and the correct conditions are satisfied. ie Joy-stick is programmed to fire a missile whenever the joy-stick is pointing up, and the joy-stick is pointing up.

Before you define a sprite to fire a missile, you must have allocated a number of sprites to be missile sprites, refer to the MISSRANGE command.

If you are defining a sprite to fire missiles, and the sprite is not animated, then the image that the sprite is linked to must have its missile options defined within the Designer program.

If the sprite to fire missiles is animated, then at least one image within each animation range must have its missile options defined within the Designer program.

If an image does not have any missile options defined, whenever that image is displayed the sprite will not be able to fire a missile. It is up to you to decide how many images you set to fire missiles. Please see the discussion elsewhere in this manual for a fuller explanation.

The *distance* variable determines how many moves the missile will make before it is removed from the screen.

The *delay* variable determines how often the sprite will be able to fire a missile, a setting of 2 means that the main sprite can only fire every 2 moves.

The *low sprite* variable informs the Supervisor of the lowest sprite number you wish to use as a missile. This variable must be equal too or higher than the low sprite number defined using the MISSRANGE command.

The *high sprite* variable informs the Supervisor of the highest sprite number you wish to use as a missile. This variable must be equal too or less than the high sprite number defined using the MISSRANGE command.

A number of missile firing sprites may share the use of missile sprites.

The *plane* variable determines the plane in which the sprite will travel.

The *plane detect* variable determines which sprites/scenery the missile will detect collisions with. ie 128 will cause the missile sprites to detect collisions with the centreground scenery any sprite travelling on the eighth plane.

The *explode animation* variable will allow you to define an explosion effect. An explosion effect is a animation sequence that will be triggered whenever a missile is in collision and the sprite number the missile collided with is reported using the MISSHIT command. Setting the *explode animation* variable to zero will simply remove the missile sprite once it had been reported by the MISSHIT command. If the missile hits a piece of scenery and the *plane detect* variable has been set to detect collisions with centreground scenery, then the *explode animation* sequence will automatically be executed, and the missile sprite removed.

Animation sequences usually have to have all of the sprites within a sequence the same size. Missile animations are the exception to the rule. Even though the size of the missile is very small, you can still animate it with an explosion that is 32x32 pixels.

A sprite will not be able to fire any missiles until you have given the sprite a number of missiles to fire. Please refer to the ADDAMM and AMMO commands.

LIMITS: Sprite number ( 0 - 63 )
Distance ( 1 - 255 )
Delay ( 0 - 255 )
Low sprite ( 0 - 63 )
High sprite ( 0 - 63 )
Plane ( 1 ,2,4,8,16,32,64,128 )
Plane detect ( 1 - 255 )
Explode animation ( 0 - off, 1 - 64 )

ILLEGAL USE: Missrange not defined
Door sprites
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated keywords:ADDAMM, **AMMO, RAMMO, MISSRANGE, MISSHIT, MISSOUND**

# MISSOUND

## MISSOUND *sound number*

```
MISSOUND 12
```

COMMAND:Programs the missiles to make a predefined sound whenever a missile is removed from the screen. The *sound number* variable may be any of the 32 sound commands that are defined within the Designer program.

It is recommended that the sound command you use clears the sound queue. This will have the effect of making the missile explosion sound play as soon as the explosion occurs.

To cancel all of the missile sounds, set the *sound number* variable to zero.

REMEMBER: If you are wanting to use noise within your explosions (recommended) then you may only use channels 1 and 4.

LIMITS: Sound number ( 0 - off, 1 - 32 )

ILLEGAL USE: Sound not defined.

Associated keywords:MISSILE, MISSRANGE, MISSHIT

# MISSRANGE

## MISSRANGE *start sprite number, finish sprite number*

MISSRANGE 40,50

COMMAND:Defines the number and range of sprites that will be used as missile sprites. Once a missile sprite range has been defined, you will not be able to use the sprites within the missile range for any other purpose. The defined missile range has to include all of the sprites you wish to use as missile sprites

If any of the sprites within the range you have specified, have already been defined, an error will occur. You should either clear the sprites with the CLINK command, or choose a new range.

LIMITS: Start sprite number ( 0 - 63 )
Finish sprite number ( 0 - 63 )

ILLEGAL USE: Sprites defined within range

Associated keywords:MISSILE, MISSHIT, MISSOUND

# MOVEALL

## MOVEALL *no variables*

MOVEALL

COMMAND:Updates all of the sprites that are currently displayed on the screen. All of the animation sequences are updated, all of the sprites are moved, the keyboard and joy-stick ports are read and any sprites controlled by them are moved.

The sprites will not be physically moved on the screen. The sprites positions within the sprite tables will be updated. To see the new updated positions you should use the VIEW command.

LIMITS: None

ILLEGAL USE: None

Associated keywords:MOVER, **MOVES, VIEW**

# MOVER

## MOVER *start sprite number, finish sprite number*

MOVER 16

COMMAND:Move a range of sprites on the screen, starting from the *start sprite number* and finishing with the *finish sprite number.*

LIMITS: Start sprite number ( 0 - 63 )
Finish sprite number ( 0 - 63 )

ILLEGAL USE: None

Associated **keywords:MOVEALL, MOVES, VIEW**

# MOVES

## MOVES *sprite number*

MOVES 54

COMMAND:moves a single sprite on the screen. The command has the same **effect** as the MOVEALL command.

LIMITS: Sprite number ( 0 63 )

ILLEGAL USE: Sprites outside window
Undefined sprites

Associated **keywords:MOVEALL, MOVER, VIEW**

# NEXTHIT

## NEXTHIT *length( 0, var )*

NEXTHIT LENGTH (0,nh)

FUNCTION:returns the number of any sprite that is currently in collision with the selected sprite. The selected sprite is defined by using the HIT command. The HIT command will return the number of sprites that are currently in collision with the selected sprite. The NEXTHIT command will return the actual sprite number that are in collision. If there are no more sprites in collision a value of 255 will be returned into the variable you have defined within the LENGTH function.

Example.

Sprite 23 is currently is collision with sprites 3 and 21.

HIT 23,LENGTH (0,RE)

This will return into variable RE the number of sprites that are currently in collision with sprite 23. For this example the value is 2. Therefore RE=2

Now using the NEXTHIT command

NEXTHIT LENGTH(0,HS)

This command will return a value, into variable HS, of 21. If we use the command again the value in variable HS is now changed to 3. If we use the command again the value in variable HS is now changed to 255, signifying that there were no more collisions to report.

The list of sprites in collision is created at the time the HIT command is used. It is not updated if the sprites move. Therefore if you wait to long before using the NEXTHIT command, the collision information will be out of date. As soon as you use the HIT command, any sprite number that have not been reported are cleared and a new list is made.

This command will not detect collisions between sprites and platforms, or sprites and missiles.

LIMITS: None

ILLEGAL USE: Length command must be used

Associated keywords:HIT, **MISSHIT**

# NODEOFF

## NODEOFF *sprite number*

NODEOFF 39

COMMAND:Turns off the node detection for the selected sprite.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Door sprites
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated keywords:NODEON

# NODEON

## NODEON *sprite number*

NODEON 39

COMMAND:programs a sprite to follow node instructions embedded into a room. In order for a sprite to detect nodes, theforrite must also be programmed to detect collisions in plane 64. Please refer to the CPLANE command ior further details on setting the collision plane.

If a joy-stick or keyboard sprite has been programmed to detect nodes, and the sprite has also been selected to drop, then the sprite will act in a slightly different way than normal. The sprite will only drop whenever the sprite is moving horizontally, if the sprite is moving diagonally, then the sprite will not drop.

Joy-stick and keyboard controlled sprites will only detect nodes that allow the sprite to change direction and allow the sprites to be removed/replaced. None of the other instructions within a node will be performed.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Door sprites
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated **keywords:NODEOFF**

# NOTE

## NOTE *channel, initial octave, initial tone, duration, initial volume, noise, volume envelope, tone envelope*

NOTE 1,3,100,-5,0,0,1,2

COMMAND:Plays a single note. This command can be used to play various notes that are outside the scope of the fixed sound effects. The command, for example, will allow you to play a note with different *initial tone* and *initial octave* variables. This is extremely useful if you wish to play a piece of music or to achieve a special effect that can not be achieved using the fixed sound effects. The format of the command is identical to the format of the fixed sound effects within the Designer program

*Channel:* This variable will allow you to select the channel number, queue status and stereo placing of the note. Please use the following table as a reference.

| Variable Queue | Variable Clear | Channel | Stereo Field | Noise Enable |
|---|---|---|---|---|
| 1 | 129 | 1 | L | NOISE |
| 2 | 130 | 2 | L | TONE |
| 3 | 131 | 3 | L | TONE |
| 4 | 132 | 4 | L | NOISE |
| 5 | 133 | 5 | L | TONE |
| 6 | 134 | 6 | L | TONE |
| 7 | 135 | 1 | R | NOISE |
| 8 | 136 | 2 | R | TONE |
| 9 | 137 | 3 | R | TONE |
| 10 | 138 | 4 | R | NOISE |
| 11 | 139 | 5 | R | TONE |
| 12 | 140 | 6 | R | TONE |
| 13 | 141 | 1 | LR | NOISE |
| 14 | 142 | 2 | LR | TONE |
| 15 | 143 | 3 | LR | TONE |
| 16 | 144 | 4 | LR | NOISE |
| 17 | 145 | 5 | LR | TONE |
| 18 | 146 | 6 | LR | TONE |

You may only use noise on channels 1 and 4, these are highlighted in the table. If you require the note to be queued you should use the value in the first column of the table. If you wish the note to be clear the queue and then play, you should use the value in the second column of the table.

*Initial octave* variable should be set for the starting octave of the sound. The octave may be from 0 (bass) to 7 (treble).

*Initial tone* variable should contain the tone number for the note to be played. Altering the tone value will allow you to play musical scales. Below is a list of the musical scale

and the tone values you should use. The first row in the table describes the note name, the second row informs you of the octave offset and the third row in the table informs you of the tone value.

The double line within the table after the A# informs you of an increase in the octave by a value of 1.

| C | C# | D | D# | E | F | F# | G | G# | A | A# | B | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | +1 | +1 |
| 33 | 60 | 85 | 109 | 132 | 153 | 173 | 192 | 210 | 227 | 243 | 5 | 33 |

*Duration* variable defines how long the sound will play for. The value informs the Supervisor of how many seconds in 1/100ths the sound will play for. If the value of duration was 400, then the note would be played for 4 seconds.

You may, instead of making the sound last for a fixed number of seconds, make the sound last for the duration of the volume envelope. If you are using a volume envelope then this method is preferred. If you wish the sound to be repeated 5 times, then you should use the value of -5 for duration. A negative number signifies repeat.

It should be notes that if you are playing notes whilst there are sprites moving on the screen, the sounds will slow down. If there are no sprites on the screen the sounds will play at full speed.

*Initial volume* variable defines at which volume level the note will start. The variable may be between 0 (silent) to 15 (loud).

*Noise* variable defines how the note will sound. To hear a pure sound you should use a value of zero. If you wish to add noise to a note, you may only use channels 1 and 4. Please see the sound table in the Designer section of the manual for a full description of how each of the noise values executes.

*Volume envelope* variable allows you to select one of sixteen volume envelopes that have been defined within the Designer program.

*Tone envelope* variable will allow you to select one of sixteen tone envelopes that have been defined within the Designer program. You may set the tone envelope to repeat the tone until the sound finishes by using a negative value for the envelope.

LIMITS: Channel ( 1 - 18 ) (+128 clear queue)
Initial octave ( 0 - 7 ) Initial tone ( 0 - 255 ) Duration ( 0 - 32767 ) ( -1 to - 255 repeat )
Initial volume ( 0 - 15 )
Noise ( 0 ), ( 1 - 8 on channels 1 and 4 only )
Volume envelope ( 1 - 16 )
Tone envelope ( 1 - 16 ) ( negative for repeat )

ILLEGAL USE: None

Associated keywords:SCOM, **SKILL, AUTOSOUND**

# PANEL

## PANEL *"screen filename"*

PANEL "GAME1"

COMMAND:Loads a screen file into both of the SCAD screens. You should already have defined an area of screen to act as a panel within the Designer program.

This command should only be used after you have loaded the drawing data into the Supervisor with the FILE command.

This command will only work if there is no room being currently displayed on the screen. It is a good idea if you use the FULLCLEAR command before using this command.

ON NO ACCOUNT SHOULD YOU LOAD A SCREEN FILE USING THE SAM BASIC COMMAND:-

LOAD *"filename"* SCREEN$

THIS WILL CAUSE THE SUPERVISOR TO BECOME CORRUPTED AND THE COMPUTER MAY CRASH. YOU HAVE BEEN WARNED.

LIMITS: Filename ( any legal sam filename )

ILLEGAL USE: Room already being displayed.

Associated keywords:FULLCLEAR

# PARTCLEAR

## PARTCLEAR *no variables*

PARTCLEAR

COMMAND:Toggles off the screen all sprites that are displayed within the main game playing area. Any sprites that are placed outside of the main game area will be left displayed on the screen. The main playing area is then cleared. Any room that is being displayed will be removed, but any panel will be left intact.

LIMITS: None

ILLEGAL USE: None

Associated **keywords:FULLCLEAR, WCLS**

# PARTWIPE

## PARTWIPE *no variables*

PARTWIPE

COMMAND:Remove all of the sprites that are currently being displayed within the main game area. Any sprites that are placed outside of the main game area will be left on the screen. The room will not clear.

LIMITS: None

ILLEGAL USE: None

Associated keywords:PARTCLEAR, FULLCLEAR, FULLWIPE, PCLS

# PCLS

## PCLS *no variables*

PCLS

COMMAND:Clear the main game playing area, any sprites displayed within the main game playing window will be cleared from the screen, however the Supervisor will still display them after the next VIEW command.

This command should not be used for clearing sprites from the screen (as it does not work). This command is useful for clearing any text that may be displayed within the main game window. If you print instructions to the screen before your game starts, then this instruction is useful in clearing the game window before printing another screen of text.

If you are going to use this command to clear text from the screen, then it would be a good idea to also use the CTEXT command.

If you are going to be printing sprites onto the screen with your instructions, then please use the PARTCLEAR command, as this will remove the sprites correctly.

LIMITS: None

ILLEGAL USE: Do not clear sprites using this command

Associated keywords:PARTCLEAR, FULLCLEAR, PARTWIPE, FULLWIPE, CTEXT

opposite direction to the platform, then the sprite will still move in the platform direction but at a reduced speed.

This single command can add a great deal of enjoyment to your games, try randomly changing the platform speed and watch your sprite trying to stay on course. You may, as well as setting a platform speed, set the platform to move using the SPEED command. Absolute mayhem !!

Using a positive value for platform speed make any sprites standing on the platform move to the right. Using a negative platform speed will make the sprites standing on the platform move to the left.

LIMITS: Sprite number ( 0 - 63 )
Platform speed ( -15 to +15 )

ILLEGAL USE: Sprite not defined as a platform
Door sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated keywords:PLATFORM

# PNUMA

## PNUMA *x coord, y coord, digits, code 0, number*

PNUMA 100,170,3,48,lives

COMMAND:Print a number onto the screen using the first character set. The number is printed at position *x coord,y coord,* unlike other sprites, this coordinate is referenced from the BOTTOM left hand side of the character.

You may print any positive integer number in the range 0 to 65535. The number will be printed using the number of digits specified in the *digits* variable. Please see the following table as an example of how the digits variable works.

| Number to print | Digit | Printed number |
|---|---|---|
| 23 | 4 | 0023 |
| 23 | 5 | 00023 |
| 123 | 3 | 123 |
| 32543 | 2 | 43 |
| 4 | 5 | 00004 |

The number variable will allow you to specify the ascii code for digit 0, this is normally 48. However, you may define a number of different number sets within a character set, by altering the code 0 variable, you may access a number of different number character designs.

The *number* variable must be a positive integer number in the range 0 - 65535.

The number is not printed to the screen straight away, instead the number is stored in a buffer. The number is printed onto the screen only after the VIEW command has been used.

LIMITS: X coord ( 0 - 255 )
Y coord ( 0 - 191 )
Digits ( 1 - 5 )
Code 0 ( 32 - 118 )
Number ( 0 - 65535)

ILLEGAL USE: None

Associated keywords:PNUMB, **TEXTA, TEXTB, CTEXT**

# PNUMB

## PNUMB *x coord, y coord, digits, code 0, number*

PNUMB 140,10,5,48,87

COMMAND:Print a number onto the screen using the second character set. Please see the PNUMA command for further details.

LIMITS: X coord ( 0 - 255 )
Y coord ( 0 - 191 )
Digits ( 1 - 5 )
Code 0 ( 32 - 118 )
Number ( 0  65535)

ILLEGAL USE: None

Associated keywords:PNUMA, **TEXTA, TEXTB, CTEXT**

# PSPUT

## PSPUT *sprite number, x coordinate, y coordinate, plane*

PSPUT 61,20,70,1

COMMAND:Place a sprite onto the screen and define it to be a permanent sprite. Permanent sprites can not move. The only thing a permanent sprite can do once if has been placed onto the screen, is to be removed.

Permanent sprites are special. Please see the section on permanent sprites elsewhere

in this manual for a full description of there uses.

A permanent sprite MUST be placed onto the screen before all other sprites. If you try to place a permanent sprite onto the screen, and there is already a normal sprite on the screen, an error message will be displayed.

Whenever a sprite is placed onto the screen it has to occupy one of eight sprite planes. Sprite planes are used by the collision detection routines. A sprite can only occupy one sprite plane. You may then set the collision detection routines for each sprite to detect collisions on certain planes. This will allow certain sprites to pass over other sprites without a collision being detected.

Once a sprite has been 'PSPUTed' onto the screen, it will have the status of a permanent sprite throughout the whole of the time the sprite is on the screen.

After removing a permanent sprite from the screen using the TOGGLE command you may replace the sprite and make it act as a normal sprite by using the SPUT command.

LIMITS: Sprite number ( 0 - 63 )
X coordinate ( 0 - 255 )
Y coordinate ( 0 - 191 )
Sprite plane ( 1,2,4,8,16,32,64,128)

ILLEGAL USE: Sprite already on screen
Missile sprites
Undefined sprites
Sprite over screen edge

Associated **keywords:SPUT, TOGGLE**

# RAMMO

## RAMMO *sprite number, length ( 0,var )*

```
RAMMO 2,LENGTH(0,aI)
```

FUNCTION:Return the amount of ammunition a sprite has left to fire. The amount of ammunition left for the sprite selected within the command will be stored in the variable specified within the LENGTH command.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Door sprites
Missile sprites
Undefined sprites
Non missile firing sprites

Associated **commands:ADDAMM, AMMO, MISSILE**

# RIMAGE

## RIMAGE *sprite number, length( 0, var )*

```
RIMAGE 23,LENGTH(0,x3)
```

FUNCTION:Report the current image being used by the sprite. The image number of the sprite selected within the command will be stored in the variable specified within the LENGTH command.

This command may be useful if you wish to animate a sprite off the screen.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Undefined sprites

Associated keywords:None

# RESTDJ

## RESTDJ *sprite number*

```
RESTDJ 43
```

COMMAND: This command will clear the jump and drop settings for the sprite defined within the command. If the sprites jump flag was set, the this will be cleared. The sprite will still be allowed to jump, the command simply cancels the sprites existing jump. The drop speeds will be restored to their original settings.

This command should only be used on sprites which have had the DJCLEAR command issued. Under normal circumstances, this command will not need to be used.

The sprite must be off the screen when this command is issued. If the sprite is displayed on the screen, an error message will be displayed.

Please refer to the DJCLEAR command for further uses of this command.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Undefined sprites
Permanent sprites
Door sprites
Missile sprites
Sprites displayed on the screen

Associated keywords:DJCLEAR

# ROOM

## ROOM *room number*

```
ROOM 4
```

COMMAND:Display a room which has been created using the room Designer within the SCADs Designer program.

Most games have scenery and this command will instantly display a complete room, complete with all background, centreground and foreground scenery. If there are any sprites displayed on the screen at the time the ROOM command is used, they will be removed. Any sprites that are outside of the main sprite playing area will be removed and then re-displayed when the room is displayed.

After using this command, you will need to SPUT any sprites you wish to be displayed back onto the screen.

LIMITS: Room number ( 0 - 255 )

ILLEGAL USE: Room not defined.

Associated keywords:None

# SATTR

## SATTR *sprite number, length( 0, var )*

```
SATTR 34,LENGTH(0,a2)
```

FUNCTION:Returns information about the sprite into the variable defined within the LENGTH command.

| | |
|---|---|
| +1 | Sprite on screen |
| +2 | Sprite defined |
| +4 | Sprite outside window |
| +8 | Sprite animated |
| +16 | Sprite has coordinates |
| +32 | Sprite will drop |
| +64 | Sprite dropping |
| +128 | Sprite jumping |

If the command returns a value of 19, (16 + 2 + 1 ) then the sprite is on the screen, the

sprite has been defined, and the sprite has coordinates.

LIMITS: Sprite number ( 0 . 63 )

ILLEGAL USE: None

Associated keywords: None

# SCOM

## *SCOM sound number*

### *SCOM* 12

COMMAND:Play one of the predefined sounds created using the Designer program.

LIMITS: Sound number ( 1 - 32 )

ILLEGAL USE: Sound not defined.

Associated keywords:AUTOSOUND, NOTE, SKILL

# SETINP

## SETINP *sprite number, input type, x speed, y speed, directions, " string info"*

SETINP 1,1,2,2,255,"11000010"

COMMAND:Allows you to program a sprite to be controlled by one of four input types.

The *input type* variable will allow you to select which input type controls the sprite. Please refer to the following table to find the desired value.

| *Input type* | *Input* read |
|---|---|
| 1 | 1st Joy-stick |
| 2 | 1st Keyboard definition (Designer) |
| 3 | 2nd Joy-stick |
| 4 | 2nd Keyboard definition (Designer) |

The *x speed* and y *speed* variables allow you to set the speed the sprite will move for any particular direction. Both of the variables should be positive, the Supervisor will automatically assign negative values if they are needed.

The *directions* variable will allow you to define the directions in which the sprite can

move. To calculate the required directions please use the following table.

| | |
|---|---|
| +1 | Up |
| +2 | Up-Right |
| +4 | Right |
| +8 | Down-Right |
| +16 | Down |
| +32 | Down-Left |
| +64 | Left |
| +128 | Up-Left |

Select the directions you wish the sprite to travel, and then add up the numbers down the left hand side of the table. The total of the numbers is the value you should use in the *directions* section of the command.

ie Only move left and right +64+4 = 68, therefore direction=68

Note, the directions the sprite can follow, and the sprite speed may change if a sprite hits a node point on the screen and the sprite has been programmed to detect nodes.

The next information required for the SETINP command is the string information. This string will allow you to program how the sprite will move, wether the sprite will jump or fire missiles. The string is made up of 8 numeric characters, when defining the string, you may not leave any characters out.

| Char | Description | Switches |
|---|---|---|
| 1 | Define jump key | 0 - No Jump<br>1 Up key<br>2 - Fire key |
| 2 | Define jump sequence | Range 1 - 8 |
| 3 | Change direction jump | 0 - No Direction Change<br>1 - Allow Direction Change |
| 4 | Define fire key | 0 - No Fire<br>1 - Up key<br>2 - Fire key |
| 5 | Set drop type | 0 - No Sprite Drop<br>1 - Sprite Drop, No Left/Right Movement<br>2 - Sprite Drop, Left/Right Movement<br>3 - Sprint Drop When Joy-stick Left/Right<br>4 - Sprite Drop When Joy-stick released |
| 6 | Initial drop speed | Range 0 - 9 |
| 7 | Acceleration speed | Range 0 - 9 |
| 8 | Continuous move | Ranae 0 - off. 1 - on |

The simplest way of understanding how all of the various settings work is to experiment.

If you are going to allow the sprite to jump, then it is a good idea to set one of the drop switches.

By using an acceleration of zero, and an initial drop speed off 1, the sprite will appear to slowly glide to the bottom of the screen.

Example, To allow sprite 1 to move around the screen, with the 'up key' to jump with sequence 4, the 'fire key' to fire missiles, under joy-stick 1 control and the sprite to accelerate when it is dropping you would use the following command

```
SETINP 1,1,2,2,255,"14021110"
```

You may assign more than 1 sprite to be controlled by the same input type

LIMITS: Sprite number ( 0 - 63 )
Input type ( 1 - 4 )
X speed ( 0 - 15 )
Y speed ( 0 - 15 )
Direction ( 0 - 255 )
String info ( 8 characters )

ILLEGAL USE: Door sprites
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated keywords:GETINP

# SMERGE

## SMERGE *sprite number, image number*

```
SMERGE 3,187
```

COMMAND:Merges a new sprite image into a sprite that is currently displayed on the screen. The new sprite image does not have to be the same size as the old sprite image.

If the new image that is going to be merged is bigger than the old image, and the new image will not fit inside the current sprite window, then the sprite will be physically moved so that is does fit inside the window.

Care should be used when using this command so that the new image will not overlap any centreground scenery, this may cause the sprite to become stuck to a piece of scenery.

If the sprite you are merging is animated and the new image is not within the current

animation range, you will need to either cancel the animation, or, alter the animation sequence to suit to new image.

LIMITS: Sprite number ( 0 - 63 )
Image number ( 0 - 255 )

ILLEGAL USE: Sprite off screen
Image not defined
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated keywords:LINK, CLINK

# SPEED

## SPEED *sprite number, x speed, y speed*

SPEED 2,3,-1

COMMAND:Sets the speeds that a sprite will move around the screen.

Positive x speeds will move the sprite to the right, negative x speeds will move the sprite to the left.

Positive y speeds will move the sprite upwards, negative y speeds will move the sprite downwards.

The maximum speed a sprite can move in any direction is 15 pixels per move.

If you issue this command that has been defined to under joy-stick/keyboard control, then after the command has been issued, the sprite will no longer be under control of the joy-stick or keyboard.

LIMITS: Sprite number ( 0 - 63 )
X speed ( - 15 to + 15 )
Y speed ( - 15 to + 15 )

ILLEGAL USE: Door sprites
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated keywords:SXSPD, SYSPD

# SPUT

SPUT *sprite number, x coordinate, y coordinate, plane*

SPUT 34,45,94,8

COMMAND:Place a sprite onto the screen at coordinates x,y. The sprite will travel in one of eight planes.

Whenever a sprite is placed onto the screen it has to occupy one of eight sprite planes. Sprite planes are used by the collision detection routines. **A** sprite can only occupy one sprite plane. You may then set the collision detection routines for each sprite to detect collisions on certain planes. This will allow certain sprites to pass over other sprites without a collision being detected.

You may not place missile sprites onto the screen. The missile sprites are handled automatically by the Supervisor program. You can not place permanent sprites onto the screen using this command. You will need to use the PSPUT command for this purpose.

It should be noted that this command will not display a sprite straight away. In order to see the sprite displayed, after is has been 'SPuTed , you should use the VIEW command.

The eight sprite planes are as follows

| Plane code | Sprite plane |
|---|---|
| 1 | 1st |
| 2 | 2nd |
| 4 | 3rd |
| 8 | 4th |
| 16 | 5th |
| 32 | 6th |
| 64 | 7th |
| 128 | 8th |

You may only place a sprite on 1 plane. This plane does not affect how the sprite will detect collisions, this is handled by the CPLANE command. The plane variable is to enable other sprites to detect collisions with this sprite. Please see the discussion on sprite planes elsewhere in this manual.

LIMITS: Sprite number ( 0 - 63 )
X coordinate ( 0 - 255 )
Y coordinate ( 0 - 191 )
Plane ( 1,2,4,8,16,32,64,128 )

ILLEGAL USE: Sprite already on screen
Sprite outside of window
Missile sprites
Undefined sprites

Associated **keywords:PSPUT, TOGGLE**

# SWINDOW

SWINDOW sprite number, x start, x finish, y start, y finish

```
SWINDOW 45,20,254,0,159
```

COMMAND: Define a separate sprite window for the selected sprite. The defined window must reside within the main sprite window. Please see the section of the manual on windows for further details.

The sprite can not be on the screen when the SWINDOW command is issued. If the sprite had been on the screen and removed before the SWINDOW command is issued, the sprites coordinates will be cleared. This means that the sprite can not be TOGGLEd onto the screen after a SWINDOW command, the sprite may only be SPUTed or PSPUTed back onto the screen.

If a sprite is placed outside of its own window, the sprite will not be able to move.

It is important to remember that the maximum size that a sprite window may use is the size of the main playing area defined within the Designer program. The window size you define for a sprite must be larger than the sprite, if the window is smaller, an error message will be displayed.

Windows can be used instead on nodes in some circumstances. Using windows is faster than using nodes. If you require a sprite to move between two points, then the Supervisor will run faster if you set a sprite window to the size of area you wish the sprite to move between. This will save time and space. Two nodes require 40 bytes of memory, setting a window does not use any memory,

Remember, you may select the sprite to bounce, wrap, disappear or stop at a window edge, please see the XEDGE and YEDGE commands.

LIMITS: Sprite Number ( 0 - 63 )
X start ( 0 - 254 )
X finish ( 1- 255 )
Y start ( 0 - 190 )
Y finish ( 1 -191 )

ILLEGAL USE: Sprites on screen
Door sprites
Permanent sprites
Window outside main window
Missile sprites
Undefined sprites

Associated **keywords:XEDGE, YEDGE**

# SXPOS

## SXPOS *sprite number, length( 0, var )*

```
SXPOS 23,LENGTH(0,x3)
```

FUNCTION:Report the current x position of a sprite. The x pixel coordinate of the sprite selected within the command will be stored in the variable specified within the LENGTH command.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Sprites off screen
Undefined sprites

Associated keywords:SYPOS, **SPUT, PSPUT**

# SXSPD

## SXSPD *sprite number, length( 0, var )*

```
SXSPD 11,LENGTH (0,xs1)
```

FUNCTION:Report the current x speed of a sprite. The speed of the sprite on the x axis selected within the command will be stored in the variable specified within the LENGTH command.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Sprites off screen
Door sprites
Permanent sprites
Sprites outside window
Undefined sprites

Associated keywords:SYSPD, **SPEED**

# SYPOS

## SYSPD *sprite number, length( 0, var )*

```
SYSPD 4,LENGTH (0,y1)
```

FUNCTION:Report the current y position of a sprite. The y pixel coordinate of the sprite

selected within the command will be stored in the variable specified within the LENGTH command.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Sprites off screen
Undefined sprites

Associated keywords:SXPOS, SPUT, PSPUT

# SYSPD

## SYSPD *sprite number, length( 0, var )*

SYSPD 61,LENGTH (0,ys1)

FUNCTION:Report the current y speed of a sprite. The speed of the sprite on the y axis selected within the command will be stored in the variable specified within the LENGTH command.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Sprites off screen
Door sprites
Permanent sprites
Sprites outside window
Undefined sprites

Associated keywords:SXSPD, SPEED

# TEXTA

## TEXTA *x coord, y coord, "text"*

TEXTA 100,45,"GAME OVER"

COMMAND: Print a message onto the screen using characters from the first character set. The characters BOTTOM LEFT hand corner is placed at the specified *x coord* and y *coord.*

If any part of the message steps over the edge of the screen, the rest of the message will not be printed.

The message is not printed straight away, instead, the message is stored in a buffer. The message will be displayed after the next VIEW command is issued. The text buffer is 200 bytes long, and so you will be able to store up to 200 characters in the text buffer

before you have to issue a VIEW command. If you wish to print long messages or instructions, then you will need to print the message in a number of parts.

LIMITS: X coord ( 0 - 255 )
Y coord ( 0 - 191 )

ILLEGAL USE:None

Associated keywords:TEXTB, PNUMA, PNUMB, CTEXT

# TEXTB

## TEXTB *x coord, y coord, "text"*

TEXTB 100,45,"GAME OVER"

COMMAND: Print a message onto the screen using characters from the second character set. Please refer to the TEXTA instruction for further details.

LIMITS: X coord ( 0 - 255 )
Y coord ( 0 - 191 )

ILLEGAL USE:None

Associated keywords:TEXTA, PNUMA, PNUMB, CTEXT

# TOGGLE

## TOGGLE *sprite number*

TOGGLE 63

COMMAND:Either remove a sprite from the screen or replace a sprite onto the screen. If the sprite is currently displayed on the screen, then 'toggling' the sprite will remove it. If a sprite is currently off the screen (but it has been on the screen previously) then ' toggling' the sprite will replace it back onto the screen, into its last position.

You may toggle permanent sprites off the screen, but they may not be toggled back onto the screen again. A permanent sprite must be 'PSPUTed' back onto the screen.

If you toggle a sprite off the screen and that sprite is currently in collision with a missile, the missile sprite will also be removed from the screen. Once the missile sprite has been removed. its entry into the MISSHIT list will be cleared.

You may toggle sprite missiles off the screen. If they were in collision with a sprite, the sprite the missile had collided with will not be removed.

LIMITS: Sprite number ( 0 - 63 )

ILLEGAL USE: Missile toggled ON to the screen
Permanent sprites toggles ON to the screen
Sprite has never been on screen
Undefined sprites

Associated keywords:SPUT, PSPUT, WIPE

# VIEW

## VIEW no variables

VIEW

COMMAND:Update all of the sprites on the screen.

All of the commands described within this section of the manual will affect the operation of the sprites, be it moving, removing, placing, etc. However very few of the commands will DIRECTLY affect the sprites that are currently displayed on the screen. When you issue a MOVEALL command, or any other command that will affect the sprites, all of the sprites are moved logically, that is , the position of the sprites within the Supervisor tables are updated. The sprites will not be physically moved on the screen.

This is were the VIEW command comes into operation. The view command will update the positions of the sprites on the screen. This command should usually be called once at the end of the MOVE,MOVER and MOVEALL routines.

ie MOVEALL:VIEW:RETURN

A small number of commands will update the screen directly, without waiting for the VIEW command to be used.

PSPUT, FULLCLEAR, PARTCLEAR, FULLWIPE, PARTWIPE, ROOM

There is a section within this manual that fully explains how the Supervisor operates.

LIMITS: None

ILLEGAL USE: None

Associated keywords:NONE

# XEDGE

## XEDGE *sprite number, left edge, right edge*

XEDGE 34,2,2

COMMAND:Sets how the sprites will behave when they hit the horizontal edges of the window. The sprites can be set to bounce, wraparound, wipe or disappear. You may set a different attribute for the left and right hand edges.

| Value | Effect |
|---|---|
| 0 | Bounce |
| 1 | Wrap |
| 2 | Stop |
| 3 | Disappear |

If you are using a joy-stick controlled sprite0, then **I** would advise you to set the edge attributes to STOP.

The default setting is BOUNCE.

LIMITS: Sprite number ( 0 - 63 )
Left edge ( 0 - 3 )
Right edge ( 0 - 3 )

ILLEGAL USE: Door sprites
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated keywords:YEDGE

# YEDGE

## YEDGE *sprite number, top edge, bottom edge*

YEDGE 4,2,1

COMMAND:Sets how the sprites will behave when they hit the vertical edges of the window. The sprites can be set to bounce, wraparound, wipe or disappear. You may set a different attribute for the left and right hand edges.

Please refer to the table printed under the XEDGE command

If you are using a joy-stick controlled sprite, then I would advise you to set the edge attributes to STOP.

The default setting is BOUNCE.

LIMITS: Sprite number ( 0 - 63 )
Top edge ( 0 - 3 )
Bottom edge ( 0 - 3 )

ILLEGAL USE: Door sprites
Permanent sprites
Sprites outside window
Missile sprites
Undefined sprites

Associated keywords:XEDGE

**All of the commands within this section of the manual must be preceded by the SCADs special instruction symbol. The special symbol, a small solid block, is displayed using the f0 function key.**

**In certain circumstances, this key may display a 0. In such cases, use the shift-0 key combination. This will achieve the same results.**

**The command set may seem a bit complex at first, but with a little use and experimentation, you will soon be able to create complex games using SCADs BASIC without having to refer to this manual at all !!!**

# COMMAND SUMMARY

AMMADD *sprite number, rounds*
AMMO *sprite number, rounds*
ANIMATE *sprite number, animation sequence*
ANIMOFF *sprite number, animation sequence, direction, delay*
AUTOSOUND *sprite number, sound type, sound effect*
BOUNCE *sprite number*
CLINK *sprite number*
COLOUR
COPYSCREEN
CPLANE *sprite number, plane detection total*
CTEXT
DOOR *sprite number*
DJCLEAR *sprite number, flag*
DROP *sprite number, initial speed, acceleration speed*
FEET *sprite number, left foot, right foot*
FILE " *drawing data filename"*
FIRE *sprite number*
FULLCLEAR
FULLWIPE
GETINP *input type, length ( 0 , var )*
HIT *length ( 0, var )*
INIT
INKBLACK
JUMP *sprite number, jump sequence*
LINK *sprite number, image number*
MAINSCR
MISSHIT *length ( 0 , var)*
MISSILE *sprite number, distance, delay, low sprite, high sprite, plane, detect, animation*
MISSOUND *sound number*
MISSRANGE *sprite number, max missile range*
MOVEALL
MOVER *start sprite number, finish sprite number*
MOVES *sprite number*
NEXTHIT *length ( 0 , var)*
NODEOFF *sprite number*
NODEON *sprite number*
NOTE *channel, octave, tone, duration, volume, noise, volume envelope, tone envelope*

PANEL *"screen filename "*

PARTCLEAR

PARTWIPE

PCLS

PLATFORM *sprite number*

PLATSPEED *sprite number, x speed of platform*

PNUMA *x coordinate, y coordinate, number length, start digit, number*

PNUMB *x coordinate, y coordinate, number length, start digit, number*

PSPUT *sprite number, x coordinate, y coordinate, sprite plane*

RAMMO *sprite number, length ( 0 , var)*

RESTDJ *sprite number*

RIMAGE *sprite number, length ( 0 , var )*

ROOM *room number*

SATTR *sprite number, length ( 0 , var )*

*SCOM sound number*

SETINP *sprite number, input type, x speed, y speed, directions, "string info"*

SMERGE *sprite number, image number*

SPEED *sprite number, x speed, y speed*

SPUT *sprite number, x coordinate, y coordinate, sprite plane*

SWINDOW *sprite number, x start, x finish, y start, y finish*

SXPOS *sprite number, length ( 0 , var )* SXSPD *sprite number, length ( 0 , var)* SYPOS *sprite number, length ( 0 , var )* SYSPD *sprite number, length ( 0 , var )* TEXTA *x coordinate, y coordinate, " text message "*

TEXTB *x coordinate, y coordinate, " text message "*

TOGGLE *sprite number*

VIEW

XEDGE *sprite number, left edge attribute, right edge attribute*

YEDGE *sprite number, bottom edge attribute, top edge attribute*

# TECHNICAL INFORMATION

The following information may be of some use to programmers.

## SUPERVISOR MEMORY MAP

| | |
|---|---|
| 0,1,2 | Free pages for BASIC |
| 3 | Main Supervisor machine code |
| 4 | Buffer for storing sprite backgrounds for screen 0 |
| 5 | Buffer for storing sprite backgrounds for screen 1 |
| 6,7 | Screen 1 |
| 8,9 | Data tables (described below) |
| 10 onwards | Graphics data |
| | |
| 13 | Dos on 256k machine |
| 14,15 | Screen 0 (main screen) on 256k machine |
| | |
| 29 | Dos on 512k machine |
| 30,31 | Screen 0 (main screen) on 512k machine |

## TABLE INFORMATION

All of the following SCADs tables occupy two pages in memory (8 and 9). The peek address for the start of the two pages is #24000 (hexadecimal) or 147456 (decimal). These tables have been included to allow you to disassemble the Supervisor to find out how it works. If may also be useful to find out information that is not covered by the SCADs commands.

### #0000:DRAWING IMAGE TABLE

This table consists of 256 entries, each containing 16 bytes of information. Every entry describes information about the sprite image.

| | |
|---|---|
| +0 | Page number image resides in ( 0 - image not defined ) |
| +1 | Width of image (Bytes) |
| +2 | Height if image |
| +3 | Width of image in pixels |
| +4,+5 | Address of image 1 |
| +6,+7 | Address of mask 1 |
| +8,+9 | Address of image 2 |
| +10,+11 | Address of mask 2 |
| +12,+13 | Total length of both drawings and mask |
| +14 | Not used |
| +15 | Internal use |

## #1000:DRAWING SCENERY TABLE

This table consists of 256 entries, each containing 16 bytes of information. Every entry describes information about the scenery drawings.

| | |
|---|---|
| +0 | Page number scenery drawing resides in ( 0 - scenery not defined ) |
| +1 | Width of scenery (Bytes) |
| +2 | Height if scenery |
| +3 | Width of scenery in pixels |
| +4,+5 | Address of scenery 1 |
| +6,+7 | Address of scenery mask 1 ( If foreground scenery ) |
| +8,+9 | Not used |
| +10,+11 | Not used |
| +12,+13 | Total length of both drawings and mask |
| +14 | Scenery type ( 1 - foreground, 7 - centreground, 3 - background ) |
| +15 | Internal use |

## #2000:ROOM TABLE

This table consists of 256 entries, each containing 16 bytes of information. Every entry describes information about rooms.

| | |
|---|---|
| +0 | Page number room data ( 0 room not defined ) |
| +1,+2 | Address of room data |
| +3,+4 | Number of items of scenery within a room |
| +5,+6 | Length of room data |
| +7,+8 | Address of node data |
| +9,+10 | Number of nodes within the room |
| +11,+12 | Length of node data |
| +13,+14 | Total length of all room data |
| +15 | Internal use |

## #3000:TEXT TABLE

This table consists of 192 entries, each containing 12 bytes of information. Every entry describes information about text.

| | |
|---|---|
| +0 | Page number text drawing resides in ( 0 - scenery not defined ) |
| +1 | Width of text (Bytes) |
| +2 | Height of text |
| +3 | Width of text in pixels |
| +4,+5 | Address of text 1 |
| +6,+7 | Address of text mask 1 |
| +8,+9 | Total length of text drawings |
| +10 | Not used |
| +11 | Internal use |

There are actually two tables described here, the text tables for both types of text start at character 32 and end at character 127, text table 2 start directly after text table 1.

## #3900:SPRITE TABLE

This table consists of 64 entries, each containing 64 bytes of information. Every entry describes information about the sprites. It should be notes that the sprites internal representation of y position and y speed is different to the reported position. Internally the screen has coordinate 0 at the very top and not the bottom, similarly a y speed of +1 is moving down the screen and not up.

+0 General flags

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | Sprite on/off screen | 16 | 4 | Sprite has coordinates |
| 2 | 1 | Image linked | 32 | 5 | Sprite is set to drop |
| 4 | 2 | Sprite outside own window | 64 | 6 | Sprite is dropping |
| 8 | 3 | Sprite animated | 128 | 7 | Sprite is jumping |

| | |
|---|---|
| +1 | Page number current sprite image |
| +2 | Current x coordinate |
| +3 | Current y coordinate |
| +4 | X width (Bytes) current sprite image |
| +5 | Y height current sprite image |
| +6,+7 | Drawing image pointer for current sprites image |
| +8,+9 | Mask image pointer for current sprite image |
| +10 | Current image number |
| +11 | X speed of sprite |
| +12 | Y speed of sprite |
| +13 | Sprite window x start (variable) |
| +14 | Sprite window x finish (variable) |
| +15 | Sprite window y start (screen top) (variable) |
| +16 | Sprite window y finish (bottom) (variable) |
| +17 | Sprite window x start (fixed) |
| +18 | Sprite window x finish (fixed) |
| +19 | Sprite window y start (screen top) (fixed) |
| +20 | Sprite window y finish (bottom) (fixed) |
| +21 | X width (pixels) current sprite image |
| +22 | X maximum, full screen edge |
| +23 | Y maximum, full screen edge |
| +24 | Current animation sequence |
| +25 | Current animation direction |

+26 Edge attribute flag

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | X left edge | 16 | 4 | Y top edge |
| 2 | 1 | | 32 | 5 | |
| 4 | 2 | X right edge | 64 | 6 | Y bottom edge |
| 8 | 3 | | 128 | 7 | |

| | |
|---|---|
| +27 | Old x speed ( before drop ) |
| +28 | Drop speed (fixed) |
| +29 | New wrap position (x) |
| +30 | New wrap position (y) |

+31 General flags

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | Sprite to wrap ( x) | 16 | 4 | Bounce flag |
| 2 | 1 | Sprite to wrap ( y ) | 32 | 5 | Joy-stick/keyboard control |
| 4 | 2 | Sprite stopped | 64 | 6 | Sprite on platform |
| 8 | 3 | Sprite to disappear | 128 | 7 | Repeat move (platform) |

+32 Allocated platform number ( 0 - 63 )
+33 Allocated door number ( 63 - 127 )
+34 Allocated missile number / Drop acceleration speed
+35 Distance missile travelled / Updated drop speed
+36 Node speed (x)
+37 Node speed (y)
+38 Jump counter (+128 backwards jump)
(+64 stationary jump)
+39 Jump pointer address / Permanent sprite buffer address
+40 Jump pointer address / Permanent sprite buffer address
+41 Allowable joy-stick directions
+42 X speed (joy-stick / keyboard control)
+43 Y speed (joy-stick / keyboard control)

+44 Joy-stick / keyboard flags

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | Joy-stick/keyboard input | 16 | 4 | Move while jumping |
| 2 | 1 | Joy-stick/keyboard input | 32 | 5 | Move type flag |
| 4 | 2 | Flag clearing drop/jump | 64 | 6 | Move type flag |
| 8 | 3 | Continuous movement | 128 | 7 | Animate off flag |

+45 General flags

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | Jump: joy-stick up | 16 | 4 | Was jumping flag |
| 2 | 1 | Jump: fire button | 32 | 5 | Start jumping flag |
| 4 | 2 | Fire: joy-stick up | 64 | 6 | Was on platform flag |
| 8 | 3 | Fire: fire button | 128 | 7 | Hit by missile flag |

| | | | | | |
|---|---|---|---|---|---|
| +46 | | Jump sequence / Missile flag | 128 | 7 | Missile sprite |
| | | | 64 | 6 | Hit a sprite |
| | | | 1-63 | 0-5 | Sprite number missile hit |

+47 Platform sprite is standing on / platform speed
+48 Feet left / right
+49 Sprite plane 1,2,4,8,16,32,64,128
+50 Plane detection
+51 Missile number hit by (sprite number of missile)

+52 General flags

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | Outside main window | 16 | 4 | Sprite hit node |
| 2 | 1 | Sprite can fire missiles | 32 | 5 | Node reverse possible |
| 4 | 2 | Animate missile off screen | 64 | 6 | About to jump |
| 8 | 3 | Started animation off | 128 | 7 | Sprite to detect nodes |

+53 Distance missile can travel
+54 Missile firing delay period (variable)
+55 Low missile sprite number code (if sprite allowed to fire missiles)
+56 High missile sprite number code (if sprite allowed to fire missiles)
+57 Plane missile will travel in
+58 Plane detection for missiles
+59 Missile firing delay period (fixed)
+60 Missile one of animation sequence number
+61 Directions allowed (when sprite is on a node)
+62 Sprite reverse code ( 0 - 7 ) / old direction allow (+128)
+63 Amount of ammunition sprite is currently carrying

## #4900:NODE REFERENCE TABLE

This table consists of 256 entries, each containing 2 bytes of information. The entries refer to the x positions across the complete screen. The entry consists of a pointer to the first y coordinate that has a node point at the relevant x position on the screen. The y coordinate table consists of a linked list, therefore for a given x coordinate ( in this table ) all of the y coordinates can be accessed at speed.

0,1 Pointer into the next table, referenced from the x coordinate

## #4BOO:NODE LOOKUP TABLE

This table consists of a linked list, linking all of the y coordinates together (for a given x coordinate), the table also contains all of the node programmed information.

+0 Y coordinate
+1,+2 Address of next relevant y coordinate (0 if no more within list)
+3,+4 Data for up
+5,+6 Data for up / right
+7,+8 Data for right
+9,+10 Data for down / right
+11,+12 Data for down
+13,+14 Data for down / left
+15,+16 Data for left
+17,+18 Data for up / left

The data will consist of one of the following.

| First byte of data | Second byte of data | Second byte of data |
|---|---|---|
| 0 | 0 | Not defined within node |
| 1 - 192A | X coordinate | Remove and replace |
| 193 | ? | Remove node |
| 194 | ? | Fire missile |
| 195 | Jump sequence | Jump left |
| 196 | Jump sequence | Jump up |
| 197 | Jump sequence | Jump right |
| 225 - 255B | X Speed | New speed |

* The y coordinate is stored first for this particular option. The Supervisor will subtract 1 from this value to calculate the exact position.

The Y speed is stored in a special form. To calculate the correct speed, the Supervisor adds a value of 16, the Y speed is now encoded in a two's compliment format for the speed.

## #5CFO:HIT LIST

This list consists of 64 bytes of information. Every time a HIT command is issued, a list of sprites in collision with the test sprite is created at this location.

## #5D30:OFF SCREEN LIST

This list consists of 64 bytes of information. Whenever a ROOM or PARTCLEAR command is issued a list of all the sprites that are outside of the main sprite window is created at this location. The sprite numbers will range from 0 - 63. If the sprite is placed onto the screen as a permanent sprite, then the top bit will be set ie. sprite number + 128.

## #5D70:MISSILE HIT LIST

This list consists of 32 entries, each containing 2 bytes of information. The information is added to the list everytime a missile collides with a sprite. Whenever a missile is removed from the screen, the entry is removed from the list.

+0 Sprite number hit by missile
+1 Sprite number of missile

## #5DBO:ON SCREEN ADDRESS LIST

This list actually consists of three separate lists. Everytime a sprite is placed onto the screen, the address of the sprite information table (#3900 - #4900) for that particular s[rite is placed into the list. This allow the Supervisor to update only the sprites that are placed onto the screen. The list formats are as follows

#5DB0: Normal sprite address's

+0,+1 Address within sprite table, terminated with #FF, #FF

#5E32: Permanent sprites address's

+0,+1 Address within sprite table, terminated with #FF. #FF

#5EB4: All sprite numbers

+0 Sprite numbers ( 0 - 63 ), terminated with #FF

## #5F00:JUMP TABLE

This table consists of 8 entries, each containing 32 bytes of information. The table stores all of the speed offsets the jumps defined within the Designer program.

| | |
|---|---|
| +0 | Jump defined flag, 0 - not defined, 1 to 255 - defined |
| +1 | Number of stages within the jump |
| +2,+3 | First stage of jump |
| +4,+5 | etc |
| +30,+31 | Last stage of jump |

## #6000:SPRITE COORDINATE TABLE

This table contains relevant information about the position of the sprite on the screen. The layout of the table is rather complex. This layout was opted for because of the necessity for speedy operation.

This table is referenced in a non standard way. In machine code, the sprite number is transferred into the low register of a two byte register, the data is then referenced by setting the high register to the data type required.

Only normal sprite positions are maintained within this table. Platform sprites, Door sprites and Missile sprites are stored separately. If the *sprite x start position* has a value of 255, then the sprite is not currently stored on the screen.

| L = Sprite number. ( 0 - 63 ) | | L = Sprite number. Bit 7 set ( 128 - 191 ) | |
|---|---|---|---|
| H=0 | Sprite x start position | H=0 | Sprite feet left value |
| H=1 | Sprite x finish position | H=1 | Sprite feet right value |
| H=2 | Sprite y start position | H=2 | Plane detection |
| H=3 | Sprite y finish position | H=3 | Plane in which sprite is travelling |

This table is only half used, and so the missile positions are also stored within this table, although in a separate section

| L = Missile number. Bit 6 set ( 64 - 127 ) | | L = Missile number. Bit 6, 7 set ( 192 - 255 ) | |
|---|---|---|---|
| H=0 | Missile x start position | H=0 | Not used |
| H=1 | Missile x finish position | H=1 | Not used |
| H=2 | Missile y start position | H=2 | Plane detection for missile |
| H=3 | Missile y finish position | H=3 | Plane in which missile is travelling |

## #6400:PLATFORM / DOOR COORDINATE LIST

This list is similar to the sprite coordinate list described above, however, this list stores all of the information about the platform and door sprites. The platform and door sprite information is stored separately as they have different collision attributes to the normal sprites.

L = Platform number ( 0 - 63 )

H=0 Platform x start position
H=1 Platform x finish position
H=2 Platform y start position
H=3 Platform y finish position

L = Platform number. Bit 7 set ( 128 - 191 )

H=0 Platform feet left value
H=1 Platform feet right value
H=2 Sprite number of platform
H=3 Plane in which Platform is travelling

L = Door number ( 63 - 127 )

H=0 Door x start position
H=1 Door x finish position
H=2 Door y start position
H=3 Door y finish position

L = Door number. Bit 7 set ( 192 - 255 )

H=0 Door feet left value
H=1 Door feet right value
H=2 Sprite number of door
H=3 Plane in which Door is travelling

## #6800:SOUND EFFECTS TABLE

This table consists of 32 entries, each containing 8 bytes of information. Every entry describes the 8 autosound sound effects that may be programmed for the first 32 sprites.

+0 Fire missile sound effect number
+1 Jumping sound effect number
+2 Landing sound effect number
+3 Walking sound effect number
+4 Scenery bounce sound effect number
+5 Sprite bounce sound effect number
+6 Window bounce sound effect number
+7 Falling sound effect number

## #6900:BULLET TABLE

This table consists of 256 entries, each containing 5 bytes of information. Every entry describes the missile attributes for every single sprite image. If the first entry contains 0, then the image has not been defined to fire a sprite.

+0 Image to use as a bullet
+1 Missile X offset
+2 Missile Y offset
+3 Missile X speed
+4 Missile Y speed

## #6E00:CENTREGROUND COORDINATES LIST

This list consists of 256 entries, each containing 4 bytes of information. Every entry describes the x,y coordinate positions of all of the centreground scenery that is currently displayed on the screen.

+0 Centreground x start position

| | |
|---|---|
| +1 | Centreground x finish position |
| +2 | Centreground y start position |
| +3 | Centreground y finish position |

## #7200:ANIMATION TABLE

This table consists of 64 entries, each containing 20 bytes of information. Every entry describes one complete animation sequence.

| | | | |
|---|---|---|---|
| +0 | Animation defined flag | +10 | Start animation - down |
| +1 | Sequence number | +11 | Finish animation - down |
| +2 | Start animation - up | +12 | Start animation - down/left |
| +3 | Finish animation - up | +13 | Finish animation - down/left |
| +4 | Start animation - up/right | +14 | Start animation - left |
| +5 | Finish animation - up/right | +15 | Finish animation - left |
| +6 | Start animation - right | +16 | Start animation - up/left |
| +7 | Finish animation - right | +17 | Finish animation - up/left |
| +8 | Start animation - down/right | +18 | Start animation - stationary |
| +9 | Finish animation - down/right | +19 | Finish animation - stationary |

## #7700:SOUND COMMAND TABLE

This table consists of 32 entries, each containing 16 bytes of information, the information describes the sound command variables.

| | |
|---|---|
| +0 | Sound channel data (+128 clear sound queue) |
| +1 | Initial octave data |
| +2 | Initial tone data |
| +3,+4 | Duration ( - value repeat x number of times) |
| +5 | Initial starting volume |
| +6 | Noise data |
| +7 | Volume envelope |
| +8 | Tone envelope (Bit 7 set, repeat) |
| +9, to +15 | Unused |

## #7900:VOLUME ENVELOPE TABLE

This table consists of 16 entries, each containing 24 bytes of information. The information describes a defined volume envelope.

| | |
|---|---|
| +0 | Number of steps |
| +1 | Size of volume step |
| +2 | Pause time |

This is repeated another 7 times, making up a complete volume envelope.

## #7A80:TONE ENVELOPE TABLE

This table consists of 16 entries, each containing 24 bytes of information. The information describes a defined tone envelope.

| | |
|---|---|
| +0 | Number of steps |
| +1 | Size of tone step |
| +2 | Pause time |

This is repeated another 7 times, making up a complete tone envelope.

## #7C00 :GENERAL ROOM INFORMATION LIST

This list consists of a number of variables describing the information about room size, etc.

| | |
|---|---|
| +0 to +15 | Panel filename (designer use only) |
| +16 | Screen Y finish coordinate |
| +17 | Screen Y start coordinate |
| +18 | Screen X finish coordinate |
| +19 | Screen X start coordinate |
| +20 | Screen X width (bytes) |
| +21 | Screen Y height |
| +22,+23 | Screen start address |

## #7020: PALETTE TABLE

This table consists of 16 entries, each entry contains a palette colour, as defined within the Designer.

| | |
|---|---|
| +0 | Background colour |
| +1 to +15 | Palette colours |

## #7C30:KEY DEFINITION TABLE

This table consists of 10 entries, each entry contains data for the key definitions

| | | | |
|---|---|---|---|
| +0,+1 | Definition UP, key 1 | +10,+11 | Definition UP, key 2 |
| +2,+3 | Definition DOWN, key 1 | +12,+13 | Definition DOWN, key 2 |
| +4,+5 | Definition LEFT, key 1 | +14,+15 | Definition LEFT, key 2 |
| +6,+7 | Definition RIGHT, key 1 | +16,+17 | Definition RIGHT, key 2 |
| +8,+9 | Definition FIRE, key 1 | +18,+19 | Definition FIRE, key 2 |

The format of the data is low byte, Lower membrane input, high byte, Upper membrane input. Please refer to the technical manual for details of the keyboard membrane.

## #7C50:SOUND QUEUES

This table consists of 24 entries, 4 entries per sound channel, each entry consists of 18 bytes of information. The information informs the Supervisor of any pending sounds that need to be played.

| | |
|---|---|
| +0 | Sound mask |
| +1 | Octave |
| +2 | Tone |
| +3 | Volume |
| +4,+5 | Volume envelope address |
| +6,+7 | Tone envelope address |
| +8 | Tone repeat flag |
| +9 | Volume repeat flag |
| +10 ,+11 | Volume duration |
| +12 | Noise information |
| +13, +14, +15 | Unused |

## #7E00:SOUND CHANNEL DATA

This table consists of 6 entries, each entry is 32 bytes long. Each entry describes the details of one sound channel.

| | |
|---|---|
| +0 | Sound flag |
| +1 | Number of items in sound queue |
| +2,+3 | Duration of the sound |
| +4 | Volume envelope in use flag |
| +5 | Number of volume steps |
| +6 | Size of volume steps |
| +7 | Permanent volume pause |
| +8 | Updated volume pause |
| +9 | Current volume level |
| +10 | Volume stereo mask |
| +11 | Volume left/right output |
| +12 | Position within volume envelope |
| +13,+14 | Address within current volume envelope |
| +15 | Repeat volume flag |
| +16,+17 | Current volume envelope pointer |
| +18 | Tone envelope in use flag |
| +19 | Number of tone steps |
| +20 | Size of tone steps |
| +21 | Permanent tone pause |
| +22 | Updated tone pause |
| +23 | Current octave |
| +24 | Current tone |
| +25 | Position within tone envelope |
| +26,+27 | Address within current tone envelope |
| +28 | Tone repeat flag |
| +29,+30 | Current tone envelope pointer |
| +31 | Position within queue |

# WHAT IS NEXT ?

This is the first program from GLENCO SOFTWARE for the Sam Coupe. We are currently planning to write a number of additional programs to compliment SCADs. The most important of these packages is probably the SCADs Compiler.

The SCADs Compiler will convert your BASIC programs into super fast machine code. This has an unbelievable effect on your games programs. The Compiler can allow your BASIC programs to run up to 16 times faster than normal. The Compiler will also allow you to sell your games and give them away to friends without breaking any copyright laws. You could even start your own software house !!!

We are also considering a SCADs Music system. The Supervisor, at present, will allow you to create some amazing sound effects, but it has not been designed to allow you to play tunes. The music system would allow you to enter sheet music, using a special music editor and to let you play the music from within your games.

We are also considering a scrolling version of SCADs. The present version of SCADs will allow small objects to scroll, but not the complete screen. We are therefore considering writing a version of SCADs purely as a scrolling games Designer.

We may be running a SCADs USER GROUP. This group will be open to everybody who purchases a copy of BASIC SCADs. The idea of the group is to link together all of the people who own SCADs. We will periodically produce a SCADs magazine giving hints and tips on using the package.

As a member of the user group, we hope you will send into us any copies of programs you have written, as we intend to set up a SCADs Shareware group. The aim of the group is for you to send in a copy of programs you have written for inclusion in our Shareware lists, we will review the software in the User magazine. People will then be able to purchase your programs through the User group for a small fee. You, as the author of the program, will receive a royalty for every copy sold.

We hope this will bring about a large number of good quality arcade games for the Sam Coupe. Who knows. You may be able to sell your software as a full price software title !!

For further details on any of the above mentioned items, please look for additional information supplied with this package, or look out for any adverts we may be placing in the Sam press.

Even if you do not join as a member of the SCADs user group, please send us your registration form,this will enable you to receive news of any further products we decide to produce.

All of the above ideas will only become reality IF we get a positive response from you, the user, on what you want to see. If we feel this version of SCADs has not sold in the quantities we expect, through piracy or lack of interest, we will not produce any more programs for the Sam Coupe. This is not blackmail, it is economic sense. This program took over two years to design.

# ERROR MESSAGE CODES

1 Sprite Number is outside of command limits

2 Image Number is outside of command limits

3 Sprite has already been defined

4 X Coordinate outside of command limits

5 Y Coordinate outside of command limits

6 Sprite is off screen

7 Sprite can only be placed on ONE plane

8 Permanent sprites can not be toggled On to the screen

9 Sprite has not been defined

10 Sprite has not been placed onto the screen

11 Missiles can not be toggled ON to the screen

12 Room number is outside of command limits

13 Room has not been defined

14 Animation sequence has not been defined

15 Image number is not within defined sequence

16 Sprite is outside of its window

17 Jump has not been defined

18 Speed variable is outside of command limits

19 Drop variable is outside of command limits

20 Sound number is outside of command limits

21 Sound type is outside of command limits

22 Autosound sprite number is too high

23 Feet offset outside of command limits

24 Doors can not move

25 Edge attribute value outside of command limits

26 LENGTH command missing from statement

27 Sprite is defined as a missile

28 Sprite is outside of its window

29 Sprite is defined as a permanent sprite
30 Sprite is defined as a platform sprite
31 Sprite is defined as a door sprite
32 The sprite is currently on the screen
33 The sprite is currently off the screen
34 Image has not been defined
35 Sequence number is outside of command limits
36 Start variable is larger than the finish variable
37 Jump variable is outside of command limits
38 Channel data outside of command limits
39 Octave variable outside of command limits
40 Tone variable outside of command limits
41 Initial volume variable outside of command limits
42 Noise variable outside of command limits
43 Volume envelope variable outside of command limits
44 Tone Envelope variable outside of command limits
45 Sound has not been defined
46 Start and finish missile sprites has already been defined
47 Sprite already defined within missile range
48 Sprite number not within defined missile range
49 Missile delay is outside of command limits
50 Missile distance is outside of command limits
51 String is the wrong length
52 Printed text buffer overflow
53 Number of places outside of command limits
54 Start digit is outside of command limits
55 Sprite has not been defined to fire missiles
56 Ammunition value outside of command limits
57 Sprite window coordinates outside of command limits
58 Text must start on an even coordinate

59 Start window coordinate is greater than finish coordinate

60 X minimum coordinate is too low

61 X minimum coordinate is too high

62 X maximum coordinate is too high

63 X dimensions to small for sprite to fit inside window

64 Y maximum coordinate is too high

65 Y maximum coordinate is too low

66 Y minimum coordinate is too low

67 Y dimensions to small for sprite to fit inside window

68 Direction variable outside of command limits

69 Direction not defined within animation sequence

70 Missile explosion delay outside of command limits

71 Normal sprites already placed onto the screen

72 Input type outside of command limits

73 Panel can not be loaded while a room is being displayed

74 Screen file length error

75 Setinp string definition outside of command limits

76 Setinp string definition set not to jump, but jump sequence specified

77 Setinp string definition set to jump, but no jump sequence specified

78 Setinp string definition jump sequence has not been defined

79 Setinp string definition set not to jump, but change direction jump is defined

80 Setinp string definition set not to drop, but drop speed/acceleration defined

81 Setinp string definition set to drop, but drop speed/acceleration are not defined

82 Sprite is already being animated off the screen

83 Sprite is not defined as a platform sprite

84 Node sprite is being replaced outside of its window