ARCHIMEDES COLOUR VIDEO DIGITISER

Pineapple Software

CONTENTS

All correspondence should be addressed to:-

Pineapple Software
39, Brownlea Gardens
Seven Kings
Ilford
Essex
1G3 9NL

Tel. 081-599 1476

Fax. 081-598 2343

Getting Started

This chapter is provided primarily for those people who want to start getting pictures on their Archimedes screens without having to read the whole of this handbook!

First you should install the podule into your Arohimedes computer. Instructions for this are supplied separately as they will vary aooording to the actual model of computer you have.

To test the hardware installation you will first need to exit the Desktop (if your oomputer boots up in this mode), by pressing f12. At the * prompt type PODULES <RETURN> and the presence of the Digitiser podule should be indicated when the list of podules is displayed.

The easiest way to test the rest of the features is by using the !DIGI application on disc 2. Place Disc 2 in the disc drive and if your computer is not already in Desktop type *DESKTOP <RETURN>.

Click with Select on the drive Icon and when the directory display appears double click with Select on the !DIGI file. (If you only have a floppy system you will also need access to the !System directory on your Apps1 disc). After a few seconds the DIGI Icon should appear on the Icon bar together with a window in the top right of the screen which may contain a moving image of your input video signal.

If no moving image is present then check that you are feeding the Digitiser with a suitable video signal. If you don't have a video signal then you can still test the software features using the demo video file on this disc. To load the video file ('Tortoise') into the Digitiser podule ram, drag the 'Tortoise' Icon to the 'Digi Source' window.

After a short while a picture of a butterfly should appear in the Digi Source window. This window shows the current picture stored in the podule ram. To digitise the image double click with Select on the Digi Source window. To utilise the many video processing options see chapter 10 of this handbook for full instructions on using this application.

Adjusting the Controls

If the moving picture shows that you have a video signal present then try adjusting the three controls on the Digitiser front panel for best results. (If you have loaded 'Tortoise' then restore the moving image by double clicking Adjust over the Digi Source window). The controls are actually very similar to those on the front of your TV set and should be adjusted in a similar way.

First adjust LIFT (Brightness) to make any dark areas in the picture appear black on the computer screen. Next adjust GAIN (Contrast) until peak white parts of the picture are just not 'clipping'. It is probably best to turn the Gain control to maximum and then gradually reduce it until full detail is seen in the white areas of the pioture. SAT (Colour) should be adjusted for best colour, and this will probably be best at near maximum setting.

Now freeze and Digitise the picture by double clicking with Select. After a few seconds a 'Digi Digitised' window should appear which can be expanded to full screen size if required. The setting of the three variable oontrols on the Digitiser can be adjusted more accurately on a trial and error basis by digitising a few pictures. To restore the 'Moving' image double click with Adjust over the 'Digi Source' window. The 'LIFT' should be set so that dark areas of the pioture appear really black on the computer soreen, and the 'GAIN' should be adjusted as high as possible without too muoh loss of detail in the white areas. There will be a small amount of interaotion between these two oontrols so it is best to adjust them by fairly small amounts once they are close to their correct settings.

Once set they should not require any further adjustment unless a video signal from a different source is used, and then it is likely that only the GAIN control will need adjustment. This control is the one nearest the side of the computer and hence the most readily adjustable.

Once the controls have been set to their best positions you can investigate the other features of the Digitiser software. Turn to section 10 of the handbook for details of other features of the 'DIGI' program and other applications supplied with the Digitiser.

INTRODUCTION to VIDEO DIGITISING

Video digitising can be regarded as a way of transferring images from a video source into a computers memory, in such a way that the picture can then be displayed on the computer screen, or analysed in any manner by the computer.

Because the video signal is in analogue form and has a frequency spectrum up to about 6Mhz it does not represent a signal that a computer can readily access directly via either the serial or parallel input ports. Because of this there has to be a certain amount of dedicated hardware involved in any device to enable a computer to convert the video picture into a form which it can understand.

There are various possible types of digitiser. In it's simplest form the digitiser can work at quite a slow speed. This means that only fairly simple hardware is required, but unfortunately this also means that only stationary images can be transferred to the computer memory. In order to be able to capture a moving image the digitiser hardware must be capable of storing a complete video 'frame'. To do this it must keep updating it's own memory with frames fifty times per second, until suoh time as the instruotion is received to ' grab', when it stops updating it's memory and retains the last image it stored.

Memory Requirements

The amount of memory required depends upon the resolution of the image stored and the number of bits used to store each pixel. As the image is eventually going to be displayed on the Archimedes screen, and the vertioal resolution of most screen modes is 256, it would seem sensible to use this as the vertioal resolution for the video ram. A slight disadvantage here is that the actual TV signal has 312.5 lines on each frame (a 625 line picture is made up from two frames). This means that if we only store 256 lines some of the picture will be lost. This is not a serious disadvantage as not all of the 312 lines oontain picture information, and only about 20 lines are actually lost, mostly near the bottom of the pioture.

The horizontal resolution could also be set to 256 pixels which would give an equivalent frequency response of around 2Mhz. This would give reasonable quality pictures, (a good video recorder has a frequency response up to about 3.0Mhz), but the Archimedes screen can provide a horizontal resolution of up to 640 or more pixels, so to get the best possible pictures more resolution would be an advantage. The Pineapple Digitiser actual uses 512 pixels horizontally which gives a resolution better than a domestic video recorder and not far short of the full PAL system.

Bits per Pixel

Another factor affecting the amount of memory used, and the eventual picture quality, is the number of bits used to store each pixel. The number of bits determine how many separate levels or shades can be reconstructed from the digital image stored in ram. In the analogue signal there are an infinite number of levels available, but a digitised image clearly has to have a finite number of levels to store the picture information. If we were to consider just a black and white image, then the number of levels used by professional video equipment is 256, which therefore needs 8 bits to store each pixel.

Colour Pictures

To store a colour image which later needs to be accessed by the computer, the most versatile approach to storing the image is to store each of the three primary red, green and blue images separately. This is achieved in the Pineapple Digitiser by decoding the PAL signal into it's R,G and B parts using an analogue decoder, and then storing each image separately in ram. The number of bits used is dictated by the capabilities of the Archimedes colour system. By using colour patterning to achieve more than the possible 256 colours, it has been found worthwhile to use 16 bits of total storage for each pixel. As 16 does not divide exactly by three, the allocation has been split into 5 bits for red and blue, and 6 bits for green. This actually produces a possible 65,536 colours!

The effect of using fewer bits is easily demonstrated using the *BITS command, and it will be seen that the number of bits can be reduced quite considerably before serious degradation of the picture occurs.

Once the Digitiser hardware has 'grabbed' the required image, the computer can then work at a speed of it's own choosing to read out each of the pixels forming the complete picture. It can transfer them to the screen as required, in any scale, orientation, colour etc.

By saving the pictures to disc in a form compatible with other software, many more possibilities become available for editing and printing of captured images.

Noise

Noise is not normally considered as a likely problem with video images these days, unlike the days of early TV sets when 'snow' was quite often a problem due to lower power transmitters and less sensitive receivers.

Unfortunately, the problem still exists when a static image is grabbed from a series of moving pictures. Because noise is a completely random phenomenon, it is not very visible on tv sets because each little spike of noise is only present in a particular place on the screen for an instant. Although the noise is present, it is not very visible as it is continually moving around on the screen. (This is of course a simplified view of things, and noise does become visible on tv pictures if it is a large enough amplitude.

With a static picture things are different. Any noise generated in the TV camera, or the transmission system, is grabbed along with the picture, and each noise 'spike' is then static on screen and clearly visible. It is possible with software routines to reduce the noise content although this inevitably takes considerably longer than the standard digitising routine. To get the best results from the Digitiser try to get the best possible noise free signal to feed it with, as it is capable of producing the highest quality pictures given a good input signal.

Cross Colour

Another form of interference which is more visible on a static image than a moving one is cross colour. This is an effect caused by the luminance component of the PAL video signal being falsely decoded by the chrominance circuits in the decoder as a colour image. It is caused by large luminance transitions which have a rise time similar to the 4.43Mhz colour subcarrier frequency. It is more often visible on domestic TV sets as a coloured moire patterning on people wearing striped shirts which happen to produce vertical lines on screen of a similar frequency to colour suboarrier.

There is little that can be done about this apart from using the R,G,B,S inputs rather than the PAL video input. Unfortunately, R,G,B outputs are only likely to be available from the more expensive video cameras, and not from domestic video equipment. The effect is however, not too serious and only occasionally visible on some types of picture.

Pixel Visibility

Because of the high resolution of the Digitiser, individual pixels will not be visible when displaying a full screen picture. The further a picture is 'zoomed in', however, the easier it becomes to see the individual pixels. This limits the usefulness of the zoom feature, (not just with this digitiser but with any eleotronic video zoom). As with noise, the visibility of the individual pixels can be masked using software techniques. A software routine oan smooth out the straight edges of the pixels and merge adjacent colours together to produce a high quality picture even when zoomed in quite considerably.

Software Requirements

The initial design philosophy for the software has been to create a series of very easy to use commands to access the most useful features of the digitiser. There are too many possibilties available, to be tied down by one piece of software, so the commands allow even the most inexperienoed programmer to create simple programs to achieve whatever he/she requires. The chapter on Basic programs shows the inexperienced programmer some simple techniques which can be used to include variables in the commands, and details some example programs to demonstrate the use of each command.

For the more experienced programmer there are SWI commands available to access the Podule Video Ram and access such features as individual video pixels. This makes it possible to produce programs to do virtually anything in the field of video analysing or display.

DIGITISER SOFTWARE

Once the video image has been captured in the podule ram some method must be provided to convert it into a viewable form on the computer screen. This is done using a software module which provides a whole range of *commands (and SWI's) to perform the necessary transferring of data from podule to screen. *Commands provide a very powerful method of accessing the podule as they may be used directly from the command line prompt,, or easily incorporated into Basic programs.

To make it easier to understand the effects of the various commands available, they have been grouped together here, and brief descriptions of each provided. Each command is described in greater detail in the alphabetically organised section.

An important point to note about the software module is that none of the *commands actually change the video image stored in the podule ram. (Apart from *LOADVIDEO which loads an image from disc into the podule ram). This means that whatever commands are used to produce an image on screen, you can always revert to producing the standard full size image again if required.

Group 1:- DIGITISE, AVERAGE, MOVING

These three commands have been grouped together because they are the only ones which actually transfer data from the podule ram to the screen. Most of the other commands simply change the way in which the first two of these work.

DIGITISE could be regarded as the fundamental Digitiser software command as this is the one which causes the picture to be drawn on the soreen. Most of the other commands (particularly groups 2,3 & 6)) are setup to simply ohange the way in which the picture is drawn, in one respect or another.

AVERAGE is a command which is particularly useful if you have a 'still' picture available as the Digitiser input source. This oould be in the form of a camera looking at a still photograph. Average does more than the simple Digitise command because it repeatedly updates the screen image with a new version of the input signal. By doing this it can greatly reduce any noise present in the camera or Digitiser hardware as any differences in the suocessive grabs will be averaged out. Parameters can be included in the *Average command to set both the way in which the averaging is done, and the number of times it is performed. Clearly this will only work with a statio input signal although there is nothing to prevent trying it with a moving input!

MOVING is a special command to allow viewing of the input video signal. As the name implies it provides a continuously updated picture which can be used to view the input video prior to grabbing a particular frame. This is particularly useful if you are using a video recorder as an input source as it means you don't require a separate TV to see what pictures are being fed to the Digitiser. A useful feature available with this command is the abilty to save whatever was on the computer screen before the moving display was started. This allows multiscreen images to be built up easily using *MOVING without corrupting pictures already stored on screen.

Group 2:- SCREENAREA, VIDEOAREA, SETSCREEN, SETVIDEO

These commands are grouped together because they can all be used to set the scaling of the image produced on screen when *Digitise (or *Average) is used. *Screenarea as the name implies, sets the actual screenarea to be used for displaying the picture. This is done by following the command with four parameters to set the left, bottom, right and top limits using standard Acorn graphics coordinates for a given screen mode. *Videoarea allows you to set in a similar manner, the actual area of the TV picture stored in the podule ram that will be displayed in the selected screen area. You need to imagine that the full size TV picture is always stored in the podule ram and that the coordinates are arranged such that the bottom left corner of the picture is 0,0 and the top right of the picture is 1024,1024.

By using these two simple commands any part of a TV picture can be displayed on any part of the computer screen.

An even simpler way of choosing screen and video areas is to use the *Setsceen and *Setvideo commands. These commands don't require the four parameters to specify the area, they actually draw boxes on screen for you, which can be manipulated in any way you choose using the Archimedes mouse. Once you have set the box to the required size and shape the routine is exited and the screen or video area parameters are automatically updated to the required values.

Group 3:- BITS, FLIP, IMAGE, MONO, NEGATIVE, PRIMARY

As with the previous group of commands for setting the scale of the picture, this group are all independant commands. This means that they may all be set quite independantly of each other without cancelling the effect of any others already set. For instance, you can specify a particular scale, a certain state for Flip, and then use Primary to only display the green component of the signal, all quite independantly.

The BITS command allows independant setting of the number of bits used to display each of the three primary colours (red, green or blue). When a video image is grabbed the podule ram automatically stores 5 bits of red information, 5 bits of blue information and 6 bits of green information. (Note that if you have the standard version with less ram, then only 4 bits of each colour are stored). The bits command is provided to allow less than the maximum no. of bits to be used when transferring the image to the computer screen. This can be useful to demonstrate the effect of 'contouring' on shaded areas of a picture, and can also be used to produce more acceptable pictures in 16 colour screen modes.

The FLIP command is used to 'flip' the picture either horizontally or vertically (or both) according to the parameters following the command.

IMAGE sets the way in which the picture is stored on the screen ram. It may either be stored straight to the screen (the default option), or it may be OR'ed, AND'ed or EOR'ed with information already stored on the screen.

MONO as the name implies produces a monchrome image on screen. It does this by adding together the red, green and blue images in the proportions 3/8R + 1/2G + 1/8B. This is quite close to the correct PAL proportions of .3R + .59G + . 11B and the calculations are considerably faster! If you are using a black and white video input signal then you should set the software to 'Mono' mode as the results will be considerably better than leaving the colour options enabled. Because of the way the palette is set for the monochrome mode, very good results can be obtained in 16 colour screen modes with a considerable saving in screen memory.

NEGATIVE is fairly obvious when considering a monochrome image, but not quite so obvious for colour images. The software works by inverting the red, green and blue images independantly so that white becomes black and black becomes white, but the intermediate colours are not so obvious. The parameters also allow inverting of just one or two colours giving even more interesting effects.

PRIMARY is a command which enables any one or two of the three primary colours to be switched off. This oan be quite useful when viewing colour pictures and you only want to see the information contained say, in the red signal.

Group 4:- SAVESCREEN, SAVESPRITE, SAVEVIDEO, SAVEAIM, SAVEFSI

This group of commands all provide ways of saving pictures to disc in one form or another.

SAVESCREEN is a command which does exactly what it says. The area of screen being saved can be specified, and this command is most useful when used to save images which will later be loaded to the screen in your own Basic programs.

SAVESPRITE saves a defined screenarea in a format suitable for use as an Acorn sprite. (The palette is saved with the data). This form of saving an image is most useful if you wish to incorporate the image into other Archimedes software. An image saved in this way can be edited and printed using Paint or loaded into many of the 'Art' and 'Publishing' packages available.

The disadvantage of the previous two methods described for saving the image is that once saved as a screen image, the pioture cannot be soaled or processed in any way using the module *commands. It can only be loaded back onto the screen again (possibly in a different place, but not in a different scale).

SAVEVIDEO is a special command which enables the complete podule ram to be saved to disc. In it's most complete form it uses a lot of disc space (256k) but does have considerable advantages over other forms of saving an image.

Because Savevideo saves the complete podule ram in a special format, all the module *commands can be used to process the image again. This is undoubtedly the best way to save an image if you don't yet know what size you will eventually want it to be on screen. Just make sure you have plenty of discs available!

SAVEAIM is a command which saves an image of the podule ram in a format suitable for use with the 'AIM' software available in the public domain. It saves a 256 * 256 resolution monochrome image which can then be loaded directly into AIM.

SAVEFSI saves a file which can be used by Roger Wilson's ChangeFSl program to perform video processing. Copies of this program are available from Roger Wilson at Acorn.

Group 5:- LOADSCREEN, LOADSPRITE, LOADVIDEO

These commands are the corresponding versions of group 4 which enable files on diso to be reloaded into the oomputer. LOADSCREEN allows any file saved using the Savescreen command to be loaded back to any specified point on the screen, and LOADSPRITE allows any spritefile to be loaded to a given soreen location.

LOADVIDEO will reload the podule ram with an image previously saved using Savevideo. This will not produce an immediate change to anything displayed on the computer soreen, of course, as a further *Digitise command will be neoessary to transfer the image to the soreen. Multipicture screens can easily be built up in this way by first saving all the required pictures on disc, and then reloading them one at a time, and transferring them in the required soale to the required position on screen.

Group 6:- FOCUS, NOISE, OUTLINE, SMOOTH

This group of commands is somewhat different to those previously described because only one of them can be selected at any one time. This means for instance that if you set up a *Noise option then any previously set Focus, Outline or Smooth options will be disabled. This is not a serious disadvantage as these are not normally commands that would require more than one to be set at a time. None of the other group oommands are affected so you can still set up soales, areas, etc.

FOCUS allows the displayed pioture to be defocussed by preset amounts. This is achieved by adding together a number of pixels surrounding the one being prooessed, and then dividing the result by the total number of pixels added together. The amount of defocus is controlled by the number of pixels added together to achieve the result. This does provide a considerable amount of noise reduction in the picture but only at the expense of ' softening' the image. It does also reduce the 'pixel visibility' to a certain extent when using zoomed in images.

NOISE provides a better way of reducing the noise in the picture without much apparent loss of focus. It still works by averaging together a number of pixels but it only performs this averaging if there are no sharply defined edges detected at that point in the picture. The edge detection is oarried out separately in both horizontal and vertical directions and various parameters may be specified to determine how the edges are detected and how the averaging is performed.

OUTLINE is a routine which uses the edge detection routines to simply draw the outlines of objects in the picture. All the processing in this group of commands **is** performed separately for the red, green and blue images and for colour pictures separate red, green and blue outlines are provided. To obtain a black and white outline simply set the monochrome option using *Mono.

SMOOTH is a routine to reduce the visibility of individual pixels when a picture has been 'zoomed in'. It uses an averaging technique similar to the other routines in this group of commands to smooth out the edges of the rectangular pixels by merging them together to make them less visible.

Group 7:- DEFAULT, FREEZE, UNFREEZE, PALETTE, RESTORESCREEN

This group of commands is really made up from those that don't fit into any of the other groups already described.

DEFAULT is a single command which sets all the other oommands to their default settings. If a *Digitise is issued after a *Default then a full screen picture of whatever is present in the podule ram will be obtained with 4 bits per colour, no flips, no noise reduction etc.

FREEZE and UNFREEZE are commands which affect the operation of the hardware in the podule. After an Unfreeze command the podule ram is in a state where it is being constantly updated with every single field of the video input signal. When a Freeze oommand is issued the hardware waits for the next field pulse and then 'freezes' the memory so that no further updating occurs. Many of the other *commands automatically use Freeze and Unfreeze to ensure that the computer does not try to access the podule ram while it is in the 'unfrozen' mode. For instance *Digitise and *Savevideo both autbmatically set the freeze option, and *Average and *Moving both continually issue 'freeze' and 'unfreeze' commands during their operation.

PALETTE is a oommand which sets the computers colour palette to the best condition according to the current soreen mode and the setting of the 'Mono' option. This is normally done automatically by the *Digitise command, but it can be useful to see the effects of instantly changing between the standard screen palette and the one preferred by the Digitiser software.

RESTORESCREEN is simply a command that can be used after *Moving to replace whatever image was stored on the screen area occupied by the moving image, after the moving image is no longer required. This allows the incoming video to be viewed without losing any part of a picture already stored on the screen.

PRINTING & **EDITING PICTURES**

There are no commands built into the software module for either editing or printing of pictures. The main reason for this is that Acorn already provide a very good medium for achieving both of these ends with the PAINT and DRAW programs which can be used from the Desktop.

The interface between the Digitser and the Paint program is achieved by saving a partioular area of picture as a sprite file using *Savesprite. It is strongly reoommended that this is done from a MODE 15 Digitised picture for best results.

If you save a sprite whioh is a full screen size MODE 15, and you only have 1Mb of ram, there is very little spare memory available when using Paint in MODE 15. This means that you will probably need to clear all software Modules from memory using *RMClear before booting up the Paint program.

Paint is extremely versatile for both editing pictures and printing them using any of the available Archimedes printer drivers. It would take too long to describe the full use of the program here so it is recommended that you refer to your Archimedes manual for details of how to use Paint (and Draw) for editing and printing sprites.

For those more experienoed programmers who can see that better printed pictures could be obtained by printing directly from the picture stored in the podule ram, rather than the image available on screen, there is a SWI command available to access pixels directly from the podule. This SWI could be inoorporated into a Basio program to enable you to write your own printer driver routine that would probably give superior results to one that works direotly from sprite or screen memory. Details of this and other SWI commands are available elsewhere in this handbook.

Screen Modes

Introduction

The effect of using the Digitiser with the numerous different screen modes available on the Archimedes is quite important and so it is worthwhile spending some time describing the attributes of each mode and the quality of picture likely to be obtained. The test mode of operation (*DIGITISE T) which generates a monochrome sawtooth, demonstrates very well the effects of *BITS and the visibilty of colour patterning on different screen modes. The sawtooth is the most difficult test for any digital video system and demonstrates how good the results can be in 256 colour high resolution screen modes. It also shows in 16 oolour modes, how an aotual picture can be quite acceptable when the test signal is obviously not very good at all!

The Digitiser works by storing the video image in it's own podule memory with a resolution of 512 pixels horizontally by 256 pixels vertically. The standard version uses 4 bits to store each of the red, green and blue signals giving a total of 12 bits per pixel. The extended version uses 5 bits of red, 6 bits of green, and 5 bits of blue giving a total of 16 bits per pixel. (Note that although 6 bits of green are available, the best results for colour images are usually obtained when the three oolours have the same bit settings).

Colour Patterns

Even with 4 bits for each oolour this gives 16 possible levels for each of the red, green and blue signals, which means a total of 4096 possible colours. Now the most number of colours obtainable with the Archimedes palette is 256 so some way has to be found to display all these extra colours on screen. The method of doing this is by using patterns for each colour to achieve the necessry extra shades. The best way to envisage this is by considering a 2 * 2 block of 4 pixels. By having different numbers of pixels switched on it is possible to obtain 4 different shades (not including black). If the pattern of the pixels is small enough, and the resolution high enough, and there are no obvious straight lines made up in the pattern, then the individual dots are practically invisible.

One problem that can arise with this system is that resolution is lost because the overall size of the blocks of pixels becomes considerably larger than the individual pixels. The Pineapple Digitiser overcomes this problem by not necessarily completing a complete pattern block for each and every pixel. If a particular colour pixel is required on screen and the system is halfway through transferring a different colour pixel to the screen, then the new pixel colour is immediately started at whatever point through the pattern block the system happens to be. This gives a resolution which is as good as the individual screen pixels and the very slight possible colour errors near to a oolour transition are too small to be visible.

The largest pattern that was considered usable for this system was an 8 pixel block spread over two consecutive screen rows, (4 pixels per row). This in itself would give 8 possible different shades, but by combining it with the fact that in the 256 colour mode the palette is also set up to give 3 bits of red, 3 bits of green and 2 bits of blue, we can see that our 6 bits of information for green can all be utilised. The 5 bits for red is easily achievable with only a 4 pixel pattern block and the 5 bits of blue can still be achieved with the full 8 pixel pattern.

With the standard 12 bit storage, the pattern never needs to be larger than 2 pixels for red and green, and 4 pixels for blue. On a mode 15 screen this makes any patterning effeots virtually invisible.

Monochrome operation

The situation in the monochrome mode is somewhat easier as the Archimedes palette can be set up to give a wider range of shades without requiring too large a pattern block. However, because the mono image is produced by adding together the red, green and blue signals, a resultant signal with more than 6 bits oan be obtained. This result is in fact limited to 7 bits which can all be used to produce the mono image on both 16 and 256 oolour modes. The way in which the mono image is obtained from R, G & B is to add them together in the proportions 3/8R + 1/2G + 1/8B. This is not exactly correct according to the PAL specification but gives very acceptable results without restricting speed of operation.

2 COLOUR MODES

Two colour modes are only capable of giving mono images, and these are limited to 3 bits (8 shades of grey).

4 COLOUR MODES

These modes are not capable of giving colour pictures, but a better monochrome image is obtainable using up to 4 bits (16 levels) of the sum of R,G & B.

16 COLOUR MODES

In these modes the colour image is limited to 3 bits red, 4 bits green and 3 bits blue. The palette is set up to use 1 bit for red, 2 bits for green and 1 bit for blue, the extra shades being achieved with colour patterns.

These are very good modes for producing monochrome images as the full 7 bits are utilised by producing a palette with 16 levels of grey, just as many as are possible in the 256 colour modes.

256 COLOUR MODES

In these modes the palette **is** not adjusted at all for monochrome images and only slightly for colour images. The colour palette is set to give 3 bits of red, 3 bits of green and 2 bits of blue. The default palette is used to achieve 16 grey levels for monochrome images.

Patterning allows the full 5 bits red, 6 bits green and 5 bits blue to be used for colour images.

MODE ATTRIBUTES

MODE	Text	Resolution	cols	memory
Mode 0	80 * 32	640 * 256	2	20k
Mode 1	40 * 32	320 * 256	4	20k
Mode 2	20 * 32	320 * 256	16	40k
Mode 3	80 * 25	Text only	2	40k
Mode 4	40 * 32	640 * 256	2	20k
Mode 5	20 * 32	320 * 256	4	20k
Mode 6	40 * 25	Text only	2	20k
Mode 7	40 * 25	Teletext	16	80k
Mode 8	80 * 32	640 * 256	4	40k
Mode 9	40 * 32	320 * 256	16	40k
Mode 10	20 * 32	320 * 256	256	80k
Mode 11	80 * 25	640 * 256	4	40k
Mode 12	80 * 32	640 * 256	16	80k
Mode 13	40 * 32	320 * 256	256	80k
Mode 14	80 * 25	640 * 256	16	80k
Mode 15	80 * 32	640 * 256	256	160k
Mode 16	132 * 32	1056 * 256	16	132k
Mode 17	132 * 25	Text only	16	132k
Mode 18	80 * 64	640 * 512	2	40k
Mode 19	80 * 64	640 * 512	4	80k
Mode 20	80 * 64	640 * 512	16	160k
Mode 21	80 * 64	640 * 512	256	320k
MODE 22	Not present			
MODE 23	144 * 56	1152 * 896	2	126k
MODE 24	132 * 32	1056 * 256	256	264k

This table may appear to differ slightly from that in the Archimedes User guide in terms of the horizontal resolution of some modes. This is because some modes are what Acorn term 'double pixel' modes. In these modes although the VDU drivers do not access the full horizontal resolution available, the Digitiser accesses screen memory directly and can make use of all the available resolution. One example would be modes such as 10 & 13 which will give identical digitised pictures but have different size text.

From the above table it can be seen that the best possible results should be obtained from mode 24. This is in fact true, but there is very little visible difference between this mode and mode 15 which is considerably more economical on memory. Mode 13 and mode 10 also give quite acceptable colour pictures while being very economical of memory usage. In general, with colour pictures, by far the best results are obtained with 256 colour modes rather than 16 colour modes. For example, mode 13 is far superior to mode 12, although they both use the same amount of screen memory. The loss of resolution on mode 13 is much less noticeable than the extra colour patterning required to achieve the colours on mode mode12.

For monochrome images mode 12 gives very good results, although if you want to have colour pictures and a monochrome picture converted from colour, on screen together, then you will need to use a 256 colour mode where the palette allows quite a good compromise between the two.

*COMMAND SUMMARY

***AVERAGE <repeats>[<type>]** Range: <0-128> <0-2>
***BITS [<R> <G> <B>]** Range: <0-5> <0-6> <0-5>
***DEFAULT**
***DIGITISE [<Shadow>[<Palette>]][<T>]** Range:<0-shadow><P> <T>
***FLIP [<H> [<V>]]** Range: <0-1> <0-1> ***FOCUS [<amount>]** Range: <0-6> ***FREEZE**
***IMAGE [<value>]** Range: <0-3>
***LOADSCREEN <filename>**
***LOADSPRITE <filename>**
***LOADVIDEO <filename>**
***MONO [<off/on>]** Range: <0-1>
***MOVING [<size> [<restore>]]** Range: <0-2> <R>
***NEGATIVE [<R> <G> <B>** [<M>]]
Range: <0-1> <0-1> <0-1> <0-1>
***NOISE [<coring> [(edge detection>]]** Range: <0-255> <1-3>
***OUTLINE [(coring> [<edge detection>]]** Range: <0-255> <1-3>
***PALETTE**
***PRIMARY [<R> <G> <B>]** Range: <0-1> <0-1> <0-1>
***RESTORESCREEN**
***SAVEAIM <filename>**
***SAVEFSI <filename>**
***SAVESCREEN <filename>**
***SAVESPRITE <filename>**
***SAVEVIDEO <filename> [<bits> [<resolution>]]**
Range: <filename> <0-2> <0-3>
***SCREENAREA [<left> <bottom> <right> <top>]**
Range: <0-1280> <0-1024> <0-1280> <0-1024>
***SETSCREEN**
***SETVIDEO**
***SMOOTH [<off/on>]** Range: <0-1>
***UNFREEZE**
***VIDEOAREA [<left> <bottom> <right> <top>]**
Range: <0-1024> <0-1024> <0-1024> <0-1024>

***AVERAGE [<repeats: (<type>]]**

RANGE: <0-128> <0-2>

DEFAULT: <0> <0>

No Parameters sets default values. (No averaging)

Average is a command which can be used to produce a higher quality image than *Digitise if a stationary video picture is available. It achieves a lower noise image by repeatedly digitising the same picture and averaging each new digitisation with previous ones. The method of averaging the pictures can be determined using the 'type' parameter.

Type 0 performs a simple *Digitise on the first pass and then for subsequent passes adds together the existing screen byte with the new pixel value and divides the result by two before storing to the screen.

Type 1 performs a simple *Digitise followed by the required number of passes where the ratio of screenbyte to new pixel added together is set by the number of passes as follows:

```
e.g. for 4 passes
           Pass 1      0 * Screenbyte +    1 * New Pixel
        Passes 2-4 3/4 * Screenbyte + 1/4 * New Pixel

     For 8 passes
           Pass 1      0 * Screenbyte +    1 * New Pixel
        Passes 2-8 3/8 * Soreenbyte + 1/8 * New Pixel
```

Note that for type 1 operations the number of passes should be a number in the range 2,4,8, otherwise the selected number of passes will be rounded down to the nearest of these values. Using a number higher than 8 will not give any further improvement to the image.

Type 2 is similar to type 1 except that the ratio of screenbyte to new pixel changes for each pass:

```
e.g.for 4 passes:-
           Pass 1      0 * Screenbyte +    1 * New Pixel
           Pass 2       1/2 * Screenbyte + 1/2 * New Pixel
           Pass 3       3/4 * Screenbyte + 1/4 * New Pixel
           Pass 4       7/8 * Screenbyte + 1/8 * New Pixel
```

No improvement will be achieved beyond 4 passes. Note also that the *Average command will not take any notice of any *Image settings.

***BITS [<R>** <G> **<B>]**

RANGE: <0-5> <0-6> <0-5>

DEFAULT: <4> <4> <4>

No Parameters sets default values.

This command sets the number of bits of each oolour to be used to display the image on screen.

The video ram always oontains the full number of bits available, (4,4,4 for standard version - 5,6,5 for extended version), but the number of bits actually used to create the image on screen can be set using this command.

The number of bits determines the number of levels of each partioular colour used to create the image. Because of the limitations of the screen, dot patterns have to be used to create the necessary number of levels for each colour even in the 256 colour modes. The notioeability of the dot patterns depends on the number of bits selected, the resolution of the chosen screen mode, and the number of colours available in that particular mode. Because of this the number of bits for each colour are limited to a maximum value less than the default for some screen modes. More information about this can be found in the section describing the different screen modes.

Some interesting effects can be achieved by limiting the number of bits, and in some low resolution modes a more acceptable image can sometimes be obtained in this way.

Changing the number of bits will affect monochrome images in a similar way to colour images. The monochrome image is actually produced by adding together the R,G, and B signals in the proportions 3/8R + 1/2G + 1/8B so the effect of just changing the number of bits of one colour will probably not be very notioeable in the case of a monochrome image.

<u>*DEFAULT</u>

No Parameters required.

This command sets all parameters to their default state. This corresponds to the state immediately after the software module is loaded.

Individual parameters may be set to their default state by issuing the relevant *oommand without any parameters.

It is worth noting that when a *Digitise oommand is issued and all the parameters are in their default state (and a Mode15 screen is used), then a special 'turbo' routine is used to display the image on the screen. This works about twice as fast as when non-default parameters are set.

***DIGITISE [<Shadow>[<Palette>]] [<T>]**

RANGE: <0-no.of shadow screens> <P>
RANGE: <T>

This command transfers the stored picture in the Podule video ram to the current or any nominated shadow screen. The picture is displayed according to the parameters set up by other star commands such as Videoarea, Screenarea, Flip, Image, Bits, Mono etc.If no parameters are specified then the picture will be stored to the currently displayed screen with the palette automatically set for the best results depending on the screen mode and whether the 'Mono' option has been selected.

If a numeric parameter is specified this will be taken to mean that the picture is to be transferred to a shadow screen. The number of shadow screens available will depend on the amount of configured screen memory and the mode chosen. A mode 15 screen for instance, will use 160k of memory so that if 480k is configured then there will actually be three screens available to store separate pictures at any one time. The default screen number is 1 so that if number 2 or 3 is specified after the *DIGITISE command nothing will appear to happen as the picture is acutally being transferred to a 'hidden' screen.

To switch the computers display to show a different shadow screen you can use the *FX113,<screen no.> command. This will instantly switch to any specified soreen.

If the 'P' option is speoified then the palette will not be affected by the Digitise command. This allows you to experiment with different palettes to produce some strangely coloured pictures! More information on the effect of the palette can be found under the section on screen modes.

If the 'T' option is used after the command then a 'test' mode is entered where an artificial monochrome 'sawtooth' waveform is generated and displayed according to all the other command parameters that have been set.

The monochrome sawtooth is the most difficult test signal for any digital video system to reproduce acourately, and the effects of the *BITS command is particularly notioeable here. The effect of the patterns used to generate the intermediate grey levels is also very noticeable using this test signal. Try it with different screen modes, numbers of bits, mono and colour modes, and looking at each primary colour in turn to see the effects of all these options on the final image. It clearly shows that for colour pictures the 256 colour modes are by far the best, but for monochrome pictures there is little to choose between 16 colour and 256 colour modes.

***FLIP [<H> [<V>]]**

RANGE: <0-1> <0-1>

DEFAULT: <0> <0>

No Parameters sets default values.

This command sets whether the image will be 'flipped' horizontally or vertically or both.

A zero for the parameter value gives normal orientation and a 1 gives a 'flipped' picture. A single parameter sets the horizontal flip state and restores the vertical state **to** normal. No parameters at all restores the default state of no 'flips' .

***FOCUS '<amount>)**

RANGE: <0-6>

DEFAULT: <0>

No Parameters sets default values. (No defocussing)

Focus is a command which can defocus the image by predefined amounts. Only the image stored on the screen is actually defocussed so that by specifying a smaller parameter value the image may be resharpened again.

The routine works by adding surrounding pixels to the one actually being transferred to the screen, and then dividing the result by the number of pixels added together.

The number of pixels added together for various parameters is as follows:

Parameter	Pixels
1	4
2	9
3	25
4	49
5	81
6	132

As the routine involves quite a large amount of processing for each pixel stored on the screen, the process slows down the *Digitise operation quite considerably, especially for the higher parameter values.

A parameter value of 0 or no parameter at all disables the routine.

*FREEZE

No parameters

This command is used to stop the incoming video updating the video ram and thus 'freeze' the picture being stored at that instant.

It is not necessary to use this command if the video is ' frozen' while watching the moving image on the Archimedes screen as the Freeze option is automatically set when exiting the *Moving routine.

The command is most useful when watching the incoming video on a normal TV set when a single key press can be used to initiate the command and 'freeze' the picture at that instant.

This gives the ability to 'grab' any single frame, whereas when grabbing while displaying the moving image on screen, only about one in fifteen frames can be frozen. This is because the video ram only updates about 15 times a second when displaying the moving image on screen rather than 50 times a second when in normal 'unfrozen' mode.

***IMAGE [<value>]**

RANGE: <0-3>

DEFAULT: <0>

No Parameters sets default value.

This oommand sets the way in which the pioture will be written to screen memory. The default value of 0 replaoes any image already on the screen with the new image.

A value of 1 will make the new soreen image the result of AND' ing the image already present with the new image.

A value of 2 will perform the OR function with the image already on screen, and a value of 3 will perform an FOR with the ourrent image.

***LOADSCREEN <filename>**

This oommand loads a previously saved area of screen to the bottom left hand corner of the currently specified screen area.

It is normally expected that the area will be loaded back into a screen of the same mode as that from which it was saved. It is possible, however, to load back to a different screen mode as long as that mode has the same number of colours as the one from which it was saved. If this is done it is likely that the size of the image will be incorrect in the horizontal direction due to different screen resolutions.

There is no protection against the image 'wrapping round' in the horizontal direction if the saved image is wider than the available space from the bottom left corner of the specified screen area to the right hand edge of the screen. Vertical ' wrap' is prevented and any part of the pioture that would be off the top of the soreen will be lost.

***LOADSPRITE <filename>**

This command loads a sprite image to the bottom left hand corner of the currently specified screenarea.

It would normally be more economical on disc space to use the *Savescreen, *Loadsoreen combination to perform this operation, but the command is available so that sprites can be edited with 'Paint' and then reloaded into a screen containing other digitised images.

***LOADVIDEO <filename>**

This command loads a previously saved image of the Video Ram back into the Podule Video Ram.

As explained elsewhere, using this method of storage does use a large amount of disc space although it does have the advantage that the picture is stored with the full resolution of the Video Ram. This means that reloading the image from disc is exactly the same as if the picture had been 'grabbed' from an incoming video souroe.

See *SAVEVIDEO for details of different file formats.

<u>*MONO </u>[<on/off>]

RANGE: <0-1>

DEFAULT: <0>

No Parameters sets default values.

This command sets whether the Digitise and Palette operations will be in monoohrome or colour mode.

A '1' parameter sets the mono option and a '0' reverts to colour mode. As explained elsewhere, any screen mode with less than 16 colours automatically defaults to monochrome operation but it does this without actually changing the setting of the mono flag, so that future operations in a mode with 16 or 256 colours will still be in colour.

***MOVING <size> [<restore>]**

RANGE: <0-2> <R>

DEFAULT: <1>

This command is used to display a moving colour picture on screen.

It's main purpose **is** to provide monitoring of the inooming video signal rather than to give an impressive colour display. The image is positioned at the bottom left hand corner of the current screen area and although there is protection against vertical 'wrap around' there is no protection from horizontal wrap if the left screen area corner point is too close to the right hand edge of the screen.

The size parameter gives three choices of display size. ' 0' gives a large image about 3/4 full screen size, a '1' gives a size of just less than a 1/4 screen, and '2' gives a smaller image still.

Currently only two sizes are supported, and these only on three screen modes. Size '1' is supported on Mode 15 and Mode 21 screens,and size '0' is provided on Mode 13 screens.

The 'R' option which can optionally follow the size is used to automatically replace the original screen image which has been overwritten by the moving picture. Even if this option is not specified, the original image may still be replaced by using the *Restorescreen command later.

The moving image routine can be exited by either pressing ESCAPE or any other key. If any other key is used then the character for the key pressed will be in the keyboard buffer on exit and can be removed as required using GET or INPUT.

When the routine is exited the video memory is automatically left in the 'freeze' mode so no further ' Freeze' command is required.

In these three 256 colour modes the image is displayed using 3 Bits of Red, 3 Bits of Green and 2 Bits of Blue which does not require any dot patterning to achieve the necessary colour shades.

The palette is automatioally set when the Moving image mode is initiated.

***NEGATIVE [<R> <G> <B> [<M>]]**

RANGE: <0-1> <0-1> <0-1> <0-1>

DEFAULT: <0> <0> <0> <0>

No Parameters sets default values.

This command allows any of the three primary colour signals to be inverted to obtain a negative image.

A fourth parameter allows a separate setting when Digitising in the monochrome mode.

This command needs little explanation as the results are fairly obvious. Some interesting effects can be obtained by only inverting one or two colours when displaying oolour pictures.

The separate settings of the colour parameters have no effect on the monochrome image, only the fourth parameter affects this mode of operation.

***NOISE [<coring level> [<edge detection>]]**

RANGE: <0-255> <1-3>

DEFAULT: <0> <2>

No Parameters sets default values (No noise reduction)

Noise is a routine which helps to reduce the visible noise in a displayed picture without notioeably degrading it in other ways. It works in a similar manner to *FOCUS in that it averages together the pixel being displayed with surrounding pixels to remove the noise. However, the routine only performs the averaging if it detects that the surrounding pixels are all within a specified amplitude range. If the routine detects that a transition with an amplitude greater than the 'coring level' is present then the averaging is not performed. This ensures that the picture sharpness is maintained, unlike the *FOCUS routine where it is deliberately reduced.

The routine works seperately in the horizontal and vertical directions and the averaging is performed either with the pixel either side of the pixel being displayed, or with the pixels above and below the central pixel, or both, depending on whether a vertical or horizontal transition is detected.

The transitions are detected by looking for a sequence of pixels (horizontally or vertically). The edge detection parameter specifies the number of pixels each side of the central pixel used for this sequence in the following manner. If the speoified number of consecutive pixels of a certain amplitude, are followed by the same number of pixels, which are all either greater or smaller than the preceding pixels by an amount greater than the specified coring level, then an edge has been detected. The SWI version of this routine allows the extra facility of being able to specify the number of successive pixels required to make an ' edge detection', separately for horizontal and vertical directions.

Because of the complicated processing involved in this routine it slows down the *Digitise operation quite considerably.

If a coring level of 0 **is** specified (or no parameter is present), then the Noise routine is disabled. note that this routine cannot be used at the same time as the Focus, Outline, or Smooth routines. If this routine is enabled then the others will be disabled.

***OUTLINE** **[<coring level> [<edge detection>]]**

RANGE: <0-255> <1-3>

DEFAULT: <0> <2>

No Parameters sets default values

Outline is a command which produces a picture consisting of just the outlines of the video picture stored in the podule ram. The outline is drawn wherever the video picture has a transition exceeding the specified 'coring level'. The edge detection is performed in the same way as for *Noise, and details of how this is done can be found on page 6-14.

As well as the parameters available with this command, the routine takes notice of other settings such as *Bits etc. which can significantly affect the results with pictures which may have outlines which are difficult to detect.

In colour modes outlines are drawn seperately for each of the three colours, red, green and blue. To obtain a black and white outline set the mono option using *MONO 1 which then adds together the R,G & B signals in the usual proportions before performing the edge detection.

A parameter of 0 or no parameter disables the routine.

***PALETTE**

No Parameters required.

This command sets the palette for any given screen mode and the setting will depend on whether the monochrome option has been selected.

Colour images can only be obtained in 16 or 256 colour modes and in the case of 16 colour modes the palette command sets up the 16 available colours to give the best possible colour rendering. If the monochrome option has been enabled using the *mono command then the palette will be set up entirely differenty to give 16 levels of grey.

With the 256 colour modes however, there is not much difference between the palette settings for colour and monochrome and it is quite possible to obtain reasonable colour pictures with the palette set up for mono operation and vice versa.

Modes with less than 16 colours will always have their palette set up to give the best possible monochrome picture, but by setting up your own palette you can achieve some interesting colour effects.

This command might appear to be unnecessary as the palette can be set automatically when issuing the *Digitise command. The reason for providing it is that a program may wish to set the palette before any Digitise commands are issued. This would enable text to be set to a certain colour which will then not change when the Digitise command is eventually used.

***PRIMARY [<R>** <G> **<B>]**

RANGE: <0-1> <0-1> <0-1>

DEFAULT: <0> <0> <0>

No Parameters sets default values.

This command sets which of the primary colours will be used to make up the screen image. It effectively allows any of the three colours to be switched off, giving eight possible variations (including no output if all three are switched off!) . A parameter value of 0 turns a colour on and a value of 1 turns it off.

If the input signal was pure white then the colours obtained for the different combinations would be White, Yellow, Cyan, Green, Magenta, Red, Blue and Black.

The command works for the monochrome option as well, removing the given signal from the calculation of the monochrome image from the equation:- 3/8R + 1/2G + 1/8B.

***RESTORESCREEN**

No parameters

This command is used to restore the screen image which has been erased by the 'moving' picture routine.

This enables the incoming video to be monitored for ' grabbing' purposes, while building up multiple pictures on screen, without having to worry about erasing part of the picture already on the screen.

Although this can be done by specifying the 'R' parameter after the *Moving command, it is sometimes convenient to be able to keep the 'frozen' final image from the moving routine until you decide to replace it with the original screen image using this command.

This command may give unpredictable results if issued without having previously used the *Moving command.

***SAVEAIM <filename>**

This command saves a file in a format suitable to be read by the 'AIM' software which is currently available in the public domain.

The format is 256 * 256 pixels monochrome, and the file is produced by averaging the 512 pixels available across the line to give the best possible results.

The monochrome image is produced as a result of adding together the R, G and B signals in the ratio 3/8R + 1/2G + 1/8B.

***SAVEFSI<filename>**

This oommand saves a file which can be used by the ChangeFSl software available from Roger Wilson at Acorn.

This software can perform various video processing functions using error diffusion dithering techniques.

***SAVESCREEN <filename>**

This command saves an area of screen to the current or specified filing system.The area saved is that currently defined using the *Screenarea or *Setscreen commands.

Of the three ways available to save images (*Savevideo and *Savesprite are also available), this uses the least amount of disc space for a given size of image. It is most useful for saving images which only need to be loaded back to the screen at the same size as the saved area.

***SAVESPRITE <filename>**

This command saves the specified screenarea as an Acorn Sprite Image. The image is stored with the palette information included in the file.

This routine is extremely useful as it provides an interface between the Digitiser and all other software which is compatible with Sprite Images. By using a file oreated in this way, the image can be edited with 'Paint' and then reloaded into the Digitser to be merged with other images if required.

This routine also provides access to printer drivers either directly or through the 'Paint' applioation. More details of this can be found under the section on printing.

***SAVEVIDEO <filename> [<Bits> [<Resolution>]]**

RANGE: <filename> <0-2> <0-3>

DEFAULT: <filename> <0> <0>

This command saves the complete image stored in the Podule Video Ram.

Although this method of saving does use a large amount of disc space it has the advantage that the picture is stored with the full resolution of the Video Ram. This means that when reloaded into Video Ram all the features of the software may be used to redisplay the image on the screen.

The two optional parameters following the filename can be used to reduce the amount of disc space required to store the image at the expense of image quality.

The first parameter allows less than the full 16 bits to be saved to disc. The options are numerical values 0,1 or 2. 0 - Saves 5 bits Red, 6 bits Green, 5 bits Blue.
1 - Saves 4 bits Red, 4 bits Green, 4 bits Blue.
2 - Saves 3 bits Red, 3 bits Green, 2 bits blue.

The second parameter allows the podule ram to be saved with half the horizontal or vertical resolution. This is most useful where the image is required to be stored to disc but it is known than when it is to be used later it will only be used at a scale of about half screen size. This means that if half resolution mode is used, the image will have a noticeable loss of quality if reloaded to full screen size, but the loss of quality will not be visible at half screen size or less.

0 - Full resolution
1 - Half horizontal resolution
2 - Half vertical resolution
3 - Half horiz and vertical resolution

The saving of disc space is quite substantial using the above techniques, and still enables all the functions provided by the other *commands to be used once the image is reloaded.

A full resolution image will use 256k of disc space. Option 1 for the bits will reduce this to 192k, and option 2 will reduce it to 128k.Using half resolution mode will reduce the size further by a factor of 2 for both horizontal and vertical modes.

By using all the reduction options the size can be reduced from 256k to only 32k, and if the image is only required at about 1/4 screen size or less then the loss of quality is only slight.

***SCREENAREA [<left> <bottom> <right> <top>]**

RANGE: <0-1280> <0-1024> <0-1280> <0-1024>

DEFAULT: <0> <0> <1280> <1024>

No Parameters sets default values.

This command sets the area of screen on which the image will be displayed. Any chosen size of image (set using *Videoarea) will be scaled to fit in the selected screen area.

If no parameters are specified after the Screenarea command then the default values for a full screen area are used.

Although the range of values shown above apply to most screen modes, modes which have a horizontal graphics resolution of 1056 pixels, (Modes 16,17,& 24) do not conform to the usual Acorn screen coordinates. In the case of these screens the maximum sizes for the horizontal values are 2112 instead of 1280.

*SETSCREEN

No Parameters required.

This command offers an alternative way of setting the screenarea parameters by displaying a variable size box on the screen which is controlled by the mouse.

Initially the size of the box is set to a default size of one quarter screen area, but subsequent calls will set the box to it's last size and position on screen. This enables multiple images of exactly the same size to be placed on screen easily.

Moving the mouse with no buttons pressed simply moves the box around the screen. Holding the left mouse button and moving the mouse changes the size of the box in a constant 5/4 ratio which maintains the correct picture aspect ratio.

Holding the centre mouse button while moving the mouse allows the size to change independantly in the X & Y directions.

The right hand mouse button is used to exit the routine and update the screen area coordinates. If Escape is used to exit the routine then the coordinates are not updated.

This routine will read the state of the VDU drivers on entry, enable them if disabled for it's own use, and then restore them to their previous state on exit. This allows the user to disable the VDU drivers if required without preventing the box from being displayed on screen.

*SETVIDEO

No Parameters required.

This command offers an alternative way of setting the videoarea parameters by displaying a variable size box on the screen which is controlled by the mouse.

When using this method of setting the videoarea you must always regard the screen as displaying the full area of the video ram. This means that you can only use the box to select an exact area of picture when the full picture is displayed on the full screen area. If a 'zoomed in' picture is displayed then positioning the box over a particular part of the picture will not select that part of the picture to the video area coordinates. Although this might appear to be a disadvantage it does provide a way of 'zooming out' which would otherwise be impossible.

One possible solution is to always keep a full size image of the video ram stored on a shadow soreen, whioh can be selected when a new video area is to be defined.

As with the SETSCREEN command moving the mouse with no buttons pressed simply moves the box around the screen. Holding the left mouse button and moving the mouse changes the size of the box in a oonstant 5/4 ratio which maintains the correct picture aspect ratio.

Holding the centre mouse button while moving the mouse allows the size to change independantly in the X & Y directions.

The right hand mouse button is used to exit the routine and update the video area coordinates.If Escape is pressed the routine will be exited without updating the coordinates.

This routine will read the state of the VDU drivers on entry, enable them if disabled for it's own use, and then restore them to their previous state on exit. This allows the user to disable the VDU drivers if required without preventing the box from being displayed on screen.

<u>*SMOOTH [<off/on>]</u>

Range: <0-1>

Default: <0>

No Parameters sets default option. (No smoothing)

This command works in a similar way to *FOCUS but detects the amount by which a picture has been 'zoomed in', and selects the best ratio of defocussing to suit the amount of zoom.

For pictures which are zoomed in to a large extent the routine can be fairly slow if a full sized image is being processed.

The routine cannot be used at the same time as *FOCUS, *OUTLINE or *NOISE.

*UNFREEZE

No parameters

This command is used to re-enable updating of the video ram by the incoming video signal.

Clearly, this will not affect any picture actually displayed on the computer screen, but will simply change the image that will produoe the next picture when *Digitise is used.

***VIDEOAREA [<left> <bottom> <right> <top>]**

RANGE: <0-1024> <0-1024> <0-1024> <0-1024>

DEFAULT: <0> <4> <1020> <1024>

No Parameters sets default values.

This command sets the area of the video ram to be used for displaying on the screen. By selecting a small area of video ram and a large screen area a 'zoomed in' image will be obtained. There is no limit to how far you can zoom in if small enough video areas are chosen, but the individual pixels become more and more visible the further you zoom in.

If no parameters are specified after the Videoarea command then the default values for a full video area are used. It should be noted that the default values are not exactly the maximum values. This is because the bottom row of the video ram and the right most pixel horizontally do not contain valid video pixels.

SWI COMMANDS

This section of the handbook gives details of all the SWI commands currently available in the Digitiser Module software. For newcomers to these commands some details of how to use them are included in the section on Basic programming, and more details can be found in the Acorn Archimedes manuals.

Digitise_GetPixel &40B00

On entry: R0 = Red bits (0-5)
R1 = Green bits (0-6)
R2 = Blue bits (0-5)
R3 = XAddress (0-511)
R4 = Y_Address (0-255)

On exit: R0-R4 & R8-R13 preserved
R5 = Red pixel
R6 = Green pixel
R7 = Blue pixel

This routine allows reading of the pixel values stored in the podule ram. The number of bits returned for each colour can be specified in R0-R2, and the address in the podule ram is specified in R3-R4. The values returned are always adjusted so that the most significant bit returned is in bit 7 of the byte.

Digitise_PutPixel &40B01

On entry: R0 = Red pixel
R1   Green pixel
R2 = Blue pixel
R3 = X_Address
R4 = YAddress

On exit: R0-R13 preserved
This routine allows pixels to be stored in specific locations in the podule ram. Input bytes should be adjusted so that the most significant bit is in bit 7 of the byte. For Red and Blue, bits 0-2 will be ignored and for Green, bits 0-1 will be ignored. Note that because the Digitiser works by transferring a complete row of pixels at a time to and from the podule ram, the overheads on this call are very high if a complete row is to be transferred. The next routine 'Digitise_PutRow' is far more efficient if more than a few pixels are to be stored to a given row.

Digitise_PutRow &40B02

```
On entry: R0 = Pointer to 1536 byte buffer
          R1 = Row number

On exit: R0-R13 preserved
```

This routine transfers a complete row of video pixels from a buffer set up by the user, to the podule ram. The buffer is in the form:- R1,G1,B1,R2, G2,B2...etc up to 512 * 3 pixels. The bytes should be in the same form as for PutPixel.

This routine is much faster than 'PutPixel' if more than a few bytes are to be transferred. The opposite routine 'LoadRow' is not necessary because 'GetPixel' keeps track of which row is currently buffered in memory and therefore does not suffer from the same speed overheads as 'PutPixel'.

Digitise_Display &40B03

```
On entry: R0   0 For normal display mode
               1 for test sawtooth mode.
          R1 = Shadow screen number
          R2 = 0 Pallette set
               1 Palette unchanged

On exit: R0-R13   preserved
```

This routine performs the same as *Digitise.

Digitise_VideoArea &40B04

```
On entry: R0 = Left coordinate (0-1024)
          R1 = Bottom coordinate (0-1024)
          R2   Right Coordinate (0-1024)
          R3 = Top coordinate (0-1024)

On exit: R0-R13 preserved
```

This routine is the same as *Videoarea

Digitise_ScreenArea &40B05

```
On entry: R0 = Left screen coordinate (0-1280/2112)
          R1 = Bottom screen coordinate (0-1024)
          R2 = Right screen coordinate (0-1280/2112)
          R3 = Top screen coordinate (0-1024)

On exit: R0-R13 preserved
```

This routine is the same as *Screenarea

Digitise_SetVideo &40B06

```
On entry: R0 = 0 Performs setvideo
             <>0 Read video area coordinates

On exit: R0 = Left coordinate
          R1 = Bottom coordinate
          R2 = Right coordinate
          R3 = Top coordinate
          R4-R13 preserved
```

This routine performs the same as *Setvideo with the additional ability to be able to read the actual coordinates set up by the routine. It is also possible to use this routine to just read the coordinates without actually performing *Setvideo.

Digitise_SetScreen &40B07

```
On entry: R0 = 0 Performs setscreen
             <>0 Read screen area coordinates

On exit: R0 = Left coordinate
          R1 = Bottom coordinate
          R2 = Right coordinate
          R3 = Top coordinate
          R4-R13 preserved
```

This routine performs the same as *Setscreen with the additional ability to be able to read the actual coordinates set up by the routine. It is also possible to use this routine to just read the coordinates without actually performing *Setscreen.

Digitise_Fileop &40B08

```
On entry: R0= Fileaction
          R1= Pointer to filename

          R0 = 0 Savescreenarea
          R0 =  1 Loadscreenarea
          R0 =  2 Savevideo ram
          R0 = 3 Loadvideo ram
          R0 = 4 Savesprite
          R0   5 Loadsprite
          R0   6 SaveAIM file
          R0 =  7 SaveChangeFSl file

On exit: R1-R13 preserved
```

This routine allows all the fileactions to be performed according to the value of R0 on entry.

Digitise_Setups &40B09

```
On entry: R0 = Action

          R0  = 0 Performs freeze
          R0 = 1 Performs unfreeze
          R0 = 2 Performs default R0
          = 3 Performs palette
          R0 = 4 Performs restorescreen

On exit: R0-R13 preserved
```

This routine performs a number of digitiser functions depending on the value of R0 on entry.

Digitise_Bits &40B0A

```
On entry: R0   Red bit setting
          R1 = Green bit setting
          R2 = Blue bit setting

On exit: R0-R13 preserved
```

This routine performs as *Bits

Digitise Flip &40B0A

```
On entry: R0    0 Horizontal flip off
             = 1 Horizontal flip on
          R1 = 0 Vertical flip off
                1 Vertical flip on

On exit: R0-R13 preserved

          This routine performs as *Flip
```

Digitise_Image &40B0C

```
On entry: R0 = 0 Store direot to screen
             = 1 AND's with existing screen byte
             = 2 OR's with existing screen byte
             = 3 EOR's with existing screen byte

On exit: R0-R13 preserved

          This routine is the same as *Image
```

Digitise Mono &40B0D

```
On entry: R0 = 0 Colour output
             = 1 Monochrome output

On exit: R0-R13 preserved

          This routine performs as *Mono
```

Digitise Moving &40B0E

```
On entry: R0    Size of moving image
          R1 = 0 Screen not restored
             = 1 Automatic restoring of screen

On exit: R0 = 1 If exited via ESCAPE key
             = 0 If any other key (still in buffer)
          R1-R13 preserved

          This routine is similar to *Moving
```

Digitise_Negative &40B0F

```
On entry: R0   0Red normal
               1Red inverted
          R1 = 0Green normal
             = 1Green inverted
          R2 = 0Blue  normal
             = 1Blue  inverted
          R3 = 0Mono  mode normal
             = 1Mono  mode inverted
```

On exit: R0-R13 preserved

This routine is similar to *Negative

Digitise_Primary &40B10

```
On entry: R0 = 0Red signal on
             = 1Red signal off
          R1 = 0Green signal on
             = 1Green signal off
          R2 = 0Blue signal on
             = 1Blue signal off
```

On exit: R0-R13 preserved

This routine is similar to *Primary

Digitise_Average &40B11

```
On entry: R0   No. of repeats
          R1 = Type
```

On exit: R0-R13 preserved

This routine is similar to *Average

Digitise_Focus &40B12

On entry: R0 = Amount of defocus

On exit: R0-R13 preserved

This routine is similar to *Focus

Digitise_Noise &40B13

```
On entry: R0 = coring level
           R1   No of H pixels used for edge detection (1-3)
           R2 = No of V pixels used for edge detection (1-3)
On exit:  R0-R13 preserved
```

This routine is similar to *NOISE with the additional facility of being able to specify the number of pixels each side of the central one, separately for H & V edge detection.

Digitise_Outline &40B14

```
On entry: R0 = coring level
           R1 = No of H pixels used for edge detection (1-3)
           R2 = No of V pixels used for edge detection (1-3)
On exit:  R0-R13 preserved
```

This routine is similar to *OUTLINE with the additional facility of being able to specify the number of pixels each side of the central one separately for H & V edge detection.

Digitise_Smooth &40B15

```
On entry: R0 = 0 No smoothing
                1 Smoothed output
On exit:  R0-R13 preserved
```

This routine works in the same way as *SMOOTH

Digitise_Sequence &40B16

```
On entry: R0 = Pointer to buffer
           R1 = No. of images required
           R2 = No. of frames between images
           R3 = 0 for 256 pixels by 200 rows
              = 1 for 256 pixels by 100 rows
On exit:  R0 = Next free address in buffer
           R1-R13 preserved
```

This routine allows a sequence of images to be grabbed and stored in a specified area of computer memory. Normally the gap between images is determined by the size of the grabbed image and the operating speed of the computer. If longer gaps are required then they may be extended by a specified no. of frames. You should ensure that the buffer is large enough to hold the specified no. of images as no checking is performed to prevent writing beyond the end of the buffer.

Digitise_Spritelnit &40B17

On entry: No entry parameters

```
On exit: R0 preserved
         R1 Pointer to Sprite control block
         R2 Pointer to Sprite
```

This routine initialises a sprite area to be used with the following SWI to enable a moving image to be used in a multitasking manner.

Digitise_SpriteRead &40B18

On entry: No entry parameters

```
On exit: R0 = 0 Do nothing
         R0 <> 0 New Sprite available
         R1 Pointer to Sprite control block
         R2 Pointer to Sprite
```

This SWI in conjunotion with the previous one enables a moving image to be displayed while allowing other tasks to be performed at the same time. This SWI should be inoluded in the multitasking loop and depending on the contents of R0 on exit a new sprite may be drawn using the relevant SpriteOp.

Digitise_SpriteMov &40B19

```
On entry: R0 = Size of moving image
             = 0 for 504 x 200 pixels
             = 1 for 252 x 100 pixels
             = 2 for 126 x 52 pixels
          R1 = 0 for normal operation
             = 1 to write sprite without unfreezing.

On exit: R0 = 0 Do nothing
            = 1 New sprite available.
```

This SWI is similar to the previous pair in that it enables a moving image to be obtained in a multitasking manner. It differs in that three sizes of moving image are available and in the way the sprite is drawn. This SWI does not actually produce a Sprite. It will normally draw the image straight to the soreen, but by using the relevant SpriteOp SWI, the image may be redirected into a Sprite and then placed on the soreen.

ERROR Nos. and MESSAGES

There are various dedicated errors which can be generated by the software module and these are detailed below along with their respective error numbers. The error messages should be self explanatory.

Error No.	**Error Message**
&00802E00	Pineapple Digitiser Podule not present
&00802E01	Parameters Incorrect
&00802E02	Parameter out of Range
&00802E03	Negative Width Value
&00802E04	Negative Height Value
&00802E05	Insufficient Parameters
&00802E06	Filename Required
&00802E07	Incompatible Mode
&00802E08	Not a Digitiser Screen File
&00802E09	Not available in this mode yet
&00802E0A	Not a video ram file
&00802E0B	Only in 256 colour modes
&00802E0C	Not in mono mode

WRITING BASIC PROGRAMS

This section is provided for those not very experienced at writing Basic programs, but who want to be able to write simple programs using Digitiser commands. It is not intended as a comprehensive Basic tutorial as there are many other books which cover this very adequately. What it does explain are a few techniques which are very useful particularly where variables are to be included as parameters to *commands.

Before attempting to write and run your own Basic programs you should set up your Archimedes computer to boot up in BASIC rather than the Desktop, and configure the RMAsize and Screensize options to the required amounts.

To make the computer boot up in Basic type:
*CO.LANGUAGE 4 <RETURN> from a * prompt.To set the RMAsize to 128k type:
*CO.RMASIZE 128K <RETURN>
and to set the screensize to 160k (for a mode 15 screen) type:
*CO.SCREENSIZE 160K <RETURN>

follow these configurations by a CTRL/BREAK.

You will find it easiest to write your programs using the BASIC EDITOR supplied on one of the applications discs. See the Archimedes handbook for details of how to use this program.

Because *commands are passed directly to the operating system by Basic programs they cannot normally contain references to Basic variables. Hence the line:-

```
          10 *SCREENAREA 50 150 600 75
```

is perfectly acceptable, but

```
          10 L%=50:B%=150:R%=600:T%=75
          20 *SCREENAREA L% B% R% T%
```

is not acceptable and would generate an error.

Clearly this is a big disadvantage as far as writing comprehensive Basic programs is concerned, and Acorn overcame the problem early on in BBC BASIC by introducing the keyword 'OSCLI' which allows Basic variables to be included in *command statements by converting the command line into a string. The syntax of this command is best shown by means of the previous example which could be written as follows:-

```
   20 OSCLI"*SCREENAREA "+STR$(L%)+" "+STR$(B%)+ "
        "+STR$(R%)+" "+STR$(T%)
```

Note that spaces must be included in the string just as if it was written out as in the first example. Although writing the command out as shown above works well it is somewhat cumbersome if it needs to be used frequently in the program. A better system is to use a procedure to include the OSCLI statement which would then enable multiple calls to be made using a single PROC statement with the variable parameters in brackets. This would be done like this:-

```
   10 L%=50:B%=150:R%=600:T%=75
   20 PROCscreenarea(L%,B%,R%,T%)


   1000 DEFPROCscreenarea(left%,bottom%,right%,top%)
   1010 OSCLI"*SCREENAREA "+STR$(1eft%)+" "+STR$(bottom%)+ "
         "+STR$(right%)+" "+STR$(top%)
   1020ENDPROC
```

A simple procedure can be written for every *command which is required in a particular program. This enables variables to be used easily, and also overcomes the other constraint of *commands which is that they may only be used at the end of BASIC program lines. Once converted to a procedure they can be used anywhere within a Basic line.

The following program is a complete example utilising two different *commands, which puts 16 copies on screen of whatever picture is stored in podule ram. The pictures are alternately flipped horizontally and vertically. It is assumed that the computer is in Basic and the Digitiser module is already installed before the program is run:

```
     5 MODE15
    10 H%=0:V%=0
    20 FOR Y%=0 TO 768 STEP 256
    30 FOR X%=0 TO 960 STEP 320
    40 PROCscreenarea(X%,Y%,X%+320,Y%+256)
    50 PROCflip(H%,V%)
    60  *DIGITISE
    70 H%=H% EOR 1
    80 NEXT X%
    90 V%=V% EOR 1
   100 NEXT Y%
   110 END
   1000 DEFPROCscreenarea(left%,bottom%,right%,top%)
      1010 OSCLI"*SCREENAREA "+STR$(left%)+" "+STR$(bottom%)+
       " "+STR$(right%)+" "+STR$(top%)
 1020 ENDPROC
 1030 DEFPROCflip(hor%,ver%)
 1040 OSCLI"*FLIP "+STR$(hor%)+" "STR$(ver%)
 1050 ENDPROC
```

Another way of incorporating variables into the Digitiser commands is to use the SWI version of the *command. The Digitiser software module provides a SWI version for all the existing *commands as well as some extra ones. These are mainly provided for use in assembly language programs, but BASIC provides a command called 'SYS' which gives access to those SWI routines. The syntax of the SYS command is as follows:-

```
   SYS"DigitiseCommand",R0,R1,R2...TO,var1%,var2%....etc.
```

The name following SYS can either be the SWI name in quotes or just the SWI number. R0,R1,R2 etc. represent the processor register that a particular variable will be passed to, and these will depend on the particular SWI command being used. The rest of the line including the keyword 'TO' is optional and is only required if the SWI routine is going to return a value to one of your BASIC variables. An example of this would be the SWI"Digitise_GetPixel" routine which returns pixel values from the podule ram.

To see which processor registers are used by a particular SWI routine check the section on SWI commands. The previous example program written using SYS would be as follows:-

```
   10 MODE15
   20 H%=0:V%=0
   30 FOR Y%=0 TO 768 STEP 256
   40 FOR X%=0 TO 960 STEP 320
   50 SYS"Digitise_ScreenArea",X%,Y%,X%+320,Y%+256
   60 SYS"Digitise_Flip",H%,V%
   70 SYS"Digitise_Display" 80
   H%=H% EOR 1
   90 NEXT X%
  100 V%=V% EOR 1
  110 NEXT Y%
  120 END
```

One command which needs special mention is *MOVING. As this is a continuous operation command, a way has to be provided to exit the routine and continue with the next line of the BASIC program. Two ways of leaving *MOVING are provided. One way is to simply press ESCAPE which exits the routine and generates an 'Escape Error' which can be detected with an ON ERROR routine.

To cater for the possibility that other keys might need to be used to exit the routine, and the fact that the program might need to detect which key was actually pressed to exit the routine, another method is also provided.

If a key other than ESCAPE is pressed during *MOVING then the routine is exited as before, 'freezing' the last video field, but in this case the key which was pressed is left in the keyboard buffer so that it's value may be easily read. E.g. .-

```
   10 *MOVING 1
   20 X%=GET
   30 PRINT CHR$(X%)
```

The above example exits *MOVING when any key is pressed and the key value is removed from the buffer in line 20. If you don't wish to know which key was pressed to exit the routine, then you should follow the *MOVING command with *FX21,0 to clear the keyboard buffer.

USING THE APPLICATIONS PROGRAMS

There are currently five applications programs inoluded with the Digitiser. Two of these are multitasking Desktop programs. Also supplied on the two discs are a number of 'Video' files which can be used to demonstrate the Digitiser software without the need for a video signal.

Both the multitasking applications will run in their own right with 1Mb of memory, but to achieve image transfer to other applications such as Paint will almost certainly require 2Mb.

The most powerful of the multitasking programs is !DIGI on disc 2 and this will be described first.

!DIGI

All the normal RiscOS guidelines are followed and for those familiar with multitasking applications most of the operations will be fairly straightforward. A full description is however given as there are some areas which may not be obvious.

Loading the Program

If you have a hard diso system then the first thing to do is to copy the !DIGI application from the floppy disc supplied, to your hard disc in the normal way.

For those with just a floppy system the !DIGI application can be run simply by double clicking with Select on the !DIGI file in the usual way. You will need access to the !System directory on Apps1 so it may be easiest to set up a disc containing this and the !DIGI application. If your Desktop is not set up to run in Mode 15 then a message will appear recommending that you change to Mode 15. The Digitiser software will operate in any screen mode but Mode 15 (or any higher resolution 256 colour mode) will give best results. The Mode may be easily changed by clicking with the Menu button over the Palette Icon.

After a few seconds the !DIGI Icon should appear on the Icon bar together with a window which may contain a moving picture (if you have a video signal connected to the Digitiser) , or just 'garbage' if no signal is present.

Clicking Menu over the !DIGI Icon will allow you to quit the application if required.

Closing the 'Digi Source' window will leave the Icon present on the Icon bar and the application can be restarted by clicking with Select on the !DIGI Icon.

Loading Video files

If you don't have a video signal available but still want to investigate the Digitiser features you can load a video file into the podule ram simply by dragging the file to either the !DIGI Icon or to the 'Digi Source' window. This will take a few seconds to load and then the image will be displayed in the Source window.

An important concept with the Digitiser is that the Source window shows the image that is present in the Digitiser Podule memory. All of the Digitiser software analysis commands (which generate Sprites) will operate on this image and once the image is unfrozen it will be lost unless it has been saved as a Video file.

Freezing and Unfreezing

If you have a video signal present and have adjusted the controls as described in section 1 you can experiment with grabbing images.

As has been described previously the 'Digi Source' window shows the current image in the podule ram. When the ram is 'unfrozen' the image will be constantly changing to show the changing input signal. To freeze any required image into the Podule Ram simply click over the Source window with the Select button.

To unfreeze the image again double click Adjust over this window. Note that when loading video images into the podule ram, the ram is automatically put into 'freeze' mode.

Digitising to produce a Sprite Image

There are two ways to start the Digitising process which generates a sprite image. Double clicking with Select over the Source window will start the process and after a short while a second window, the 'Digi Digitised' window will appear. This will initially be about a quarter screen size, but the actual digitising process will have created a full screen size sprite, and the window may be sorolled or expanded to full size to show the whole image.

Another important concept here is that this image is produced according to whatever parameters have been set up by the Menu options. This Sprite image will not be lost if the Source window is 'unfrozen', - it may still be saved as a Sprite file, but no more analysis may be performed on this particular image once the Source window contains a new image. (Unless it has been saved as a video file which can then be reloaded into the podule ram).

Menu Options - Saving Images

Clicking with Menu over the Source or Digitised windows will produce a list of options.

One of these options is 'Digitise' which is the second method of starting the Digitise operation and may be more convenient than double clicking when the menu options are displayed.

The-first option is 'Save' and works in the normal RiscOS manner. If you have just digitised an image then the default file type will have been set to 'Sprite'. To save the Sprite image simply drag the Sprite Icon to a directory viewer or to another multitasking application. If the Sprite name is a full file specification then it is possible to just olick on the 'OK' box.

The second option is 'FileType' and this gives four sub-options for saving the image in Sprite, Video, ChangeFSl, or AIM format. ChangeFSl and AIM options are to allow saving the image in a raw format suitable for use by these programs.

The video format has further options available desoribed fully in section 6-20.

Apart from the options for scaling the Sprite image, and the Sequence option, all of the other menu items are desoribed in section 6 of this handbook under their respective '*' command.

Scaling and Zooming the Image

One of the most powerful features of the Digitiser software is the ability to scale and zoom any portion of the image to any required size.

To digitise an area of image which is less than the full source window a box may be dragged in the Source window using the Select button. Once the box size has been set, it may be moved around the window by dragging with Adjust.

To remove the box simply drag a box to the full size of the Source window or just drag a tiny box. The box produced in this way may not be exactly 5:4 aspect ratio but when Digitised, the resulting sprite will be produced to the same aspect ratio, so that the image is undistorted. In this case the largest of the X or Y dimensions is used to set the Sprite size.

If the 'Distort' option is selected from the menu, then the chosen video area will digitise into a 5:4 Sprite and thus the aspect ratio may be deliberately distorted.

The final Sprite size may be set using the 'Image Size' setting which is a sub-menu from 'Options'. This allows a Sprite of any size to be generated and the apsect ratio may either be locked to 5:4, or set to any desired value.

A further method of setting the Sprite size is available by cropping the image once it has been digitised. This is achieved by dragging out a box over the 'Digitised' window in a similar manner to that used for the Video Area. Once the image has been oropped it is not possible to restore the full area other than by re-digitising the image. Note that a cropped image is saved as a sprite in the current screen mode. All unoropped Sprites are saved as Mode 15 images.

Sequences

One new option available with the !DIGI software is the ability to save a sequence of frames from a video input signal. It is not possible to grab every single frame of an image, but if the smallest image size is ohosen, then even with an Arm2 processor it is possible to grab approximately every third frame. For speed, colour resolution is limited to 8 bits. (256 colours).

The size of image may be set to Small, Medium, or Large and the number of images which can be grabbed into available memory, and the speed with which they can be grabbed, will depend to a large extent on the size chosen. Even with a 1Mb computer you can grab up to 33 small images.

The amount of memory that will be used by the resulting Sprite is shown, (and limited by available memory), so that if you only have floppy discs you can restrict your files to less than 800k!

The amount of time between grabs may be set by specifying the number of frames. The minimum setting allowed will depend on the size of image chosen. If the time specified is greater than 25 frames then multitasking operation is possible allowing, say, time lapse images to be grabbed while performing other tasks.

When the complete group of frames has been grabbed a new window will display as many of the images as possible. Clicking Menu over any of the images will allow the complete sequence to be saved as a Sprite. Individual frame nos. will also be displayed and individual frames may be deleted from the set. Double clicking Select will display that particular frame full size in the 'Digitised' window. This may then be saved as a single Sprite if required.

Using !DIGI with PAINT

It is worthwhile running the Paint application alongside the Digitiser as the ease of transferring Sprites into Paint by dragging them from one window to another enables many images to be visible at the same time. The name of each successive Sprite saved will automatically be appended with increasing letters of the Alphabet to avoid having to type a new name each time a new Sprite is appended into the Paint window.

Note that the !DIGI software allows direct memory transfer to suitable receiving applications if sufficient memory is available. If not, then a WimpScrap file is set up on the current diso to allow image transfer. Floppy users note that a disc with sufficient space available must be present in the drive if insufficient memory is available for the transfer.

!PINEAPPLE

This application is produced by Ace Computing and is also fully RiscOS based.The main difference is in the way scaling and zooming are achieved.

The size of the saved Sprite is determined by the Sprite save box which enables the size to be set using numeric entry.

The size of the Digitised window may be reduced to show just a certain part of the Sprite, and if the option to save a 'visible window' is used, then this portion of the image will be scaled to fit the Sprite size requested to obtain the final image. This allows 'zooming in' to be performed.

!MICCI

This non-multitasking application is produced by Chessfield Software and is particularly easy to use.

An enhanced version of this program is available directly from Chessfield Software which has the added ability to add anti-aliased text to the images, edit the video ram, and produce an audio-visual type display where mixing between a series of images can be performed. Details of this are available from:-
Chessfield Software, 61 Chessfield Park, Little Chalfont, Bucks, HP6 6RU.

Instructions are supplied on the ReadMe file which can be loaded into Edit and printed out if required.

MAINKEYS

This program is particularly useful if you are planning to write your own software as the effects of the various '*' commands can be easily seen. By examining the BASIC program file the BASIC programming techniques used can be studied.

When first started, the program should display a moving image in the centre of the screen, (this assumes that the Digitiser controls have been set as described in the ' Getting Started' section and a video signal is present).

Pressing the 'D' key will remove the moving image and display it full size on the computer screen. Pressing SPACE will go back to showing a moving image in the centre of the screen again.

All the various features can be tested by using certain keys on the keyboard to set them up, followed by the 'D' key to Digitise the current settings.

As an example, if the 'P' key is pressed, a box appears showing the current settings of the 'PRIMARY' parameters. New parameters can be entered and then 'D' pressed to show the results.

Two of the most useful keys are the 'C' and 'V' keys. 'C' allows you to set up the area of screen which will be used to display the image when digitised, and 'V' allows the area of the picture to be set which will be displayed. The areas are set using the mouse by simply moving the mouse around to move the area, holding the left mouse button depressed to change the size of the area in a fixed aspect ratio, and holding the centre mouse button to ohange the area in a variable aspect ratio. Pressing the right hand mouse button will exit the routine and set up the area, or pressing ESCAPE will exit the routine leaving the area unchanged.

The 'V' key selects a 'shadow screen' to set the video area, and if the shadow screen does not already contain the full image of the podule ram, then there will be a short wait while this loads. Once loaded, you can switch instantly from a full size picture to the actual soreen you are using to display your images.

The 'HELP' screen selected with the 'H' key also uses the shadow screen, so this may be selected without destroying any images you may be building up on the main screen. The help screen shows which keys perform which functions, and also shows the current settings. Settings may also be changed from this screen in the usual way.

The best way to use this program to build up different images on screen, is to use the 'G' key to 'grab' an image at the appropriate moment from the moving image. Then set up the screen area you want to display it in, (and video area if required), and then press 'D' to actually perform the digitising. You can of course set up any of the other options before doing this if you wish. In fact you can put lots of different copies of the same image on screen at the same time with different option settings to see the differences!

A summary of the keys used is given at the end of this section.

MULTIPICS

This is a very simple demo program which stores multiple images on screen either from a key press or automatically. This is also quite useful to show how a simple BASIC program can be written to utilise the Digitiser features.

After starting the program by double clicking on the ! MULTIPICS Icon you will be asked how many pictures across the screen you require. Start by entering 4 <RETURN> then also enter 4 for pictures down the screen. Next you will be asked if you want continuous display. Answer 'Y' and then enter '0' for seconds between frames.

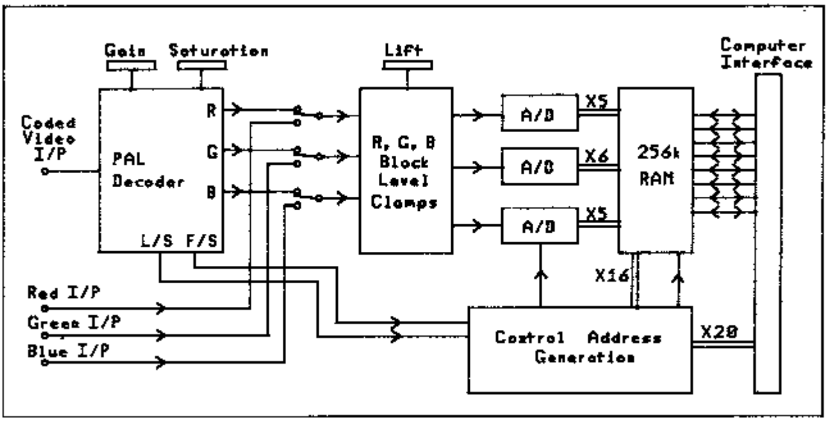
You should now get a continuous display of 16 pictures on screen which update every couple of seconds. You can enter values to get as many as 100 pictures on screen at a time if you wish. The other option for non-continuous display lets you grab pictures when you wish by pressing the SPACE BAR.

Key Summary for MAINKEYS

Use the HELP soreen ('H' key) for latest key funotions.

A	AVERAGE
B	BITS
C	SETSCREEN
D	DIGITISE
E	SCREENAREA
F	FLIP
G	FREEZE
H	Help Screen
I	IMAGE
J	VIDEOAREA
L	LOADSCREEN
SHIFT/L	LOADSPRITE
CTRL/L	LOADVIDEO
M	MONO
N	NEGATIVE
O	OUTLINE
P	PRIMARY
Q	Quit R DEFAULT (Default settings)
S	SAVESCREEN
SHIFT/S	SAVESPRITE
CTRL/S	SAVEVIDEO
ALT/S	SAVEAIM
SHIFT/ALT/S	SAVEFSI
T	DIGITISE (Test)
U	SMOOTH
✓	SETVIDEO
W	NOISE
X	DIGITISE P (Default Palette)
Y	FOCUS
Z	Clear Soreen
*	Enter * commands
SPACE	MOVING

HARDWARE BLOCK SCHEMATIC



This simplified block schematic shows the main parts of the Digitiser hardware. The input video signal is adjusted for gain and saturation and decoded into it's R,G & B components. At this point a video switch enables either these decoded R,G,B signals or external R,G,B inputs to be fed into an R,G,B clamp circuit. The point at which these signals are clamped is adjusted with the Lift (Brightness) control.

The three clamped signals are then fed into separate high speed 6 bit analogue to digital converters. The least significant bits of the red and blue images are discarded leaving a total of 16 bits available for storing in the dynamic ram.

The ram is organised as 512 * 256 * 16 bits, and the data lines are fed bi-directionally to and from the computer podule interface.

Control signals to feed addresses to the ram circuitry and the A/D converters are generated in the most complex part of the Digitser hardware. This is too complex to describe in detail here, suffice it to say that line and field synchronising signals derived from the PAL decoder are used to synchronise the ram addresses during video writing cycles, and computer address timing signals are used during computer ram accesses.

A description of the internal hardware adjustments and their effects on the Digitiser performance can be found under the section 'Hardware Adjustments'

HARDWARE ADJUSTMENTS

Although the Digitiser is very carefully lined up and tested before leaving Pineapple it is possible that the user may want to make his own adjustments. Only the adjustments that can be made without the use of an oscilloscope will be described here, those that require special test equipment are beyond the scope of this manual and you should refer to Pineapple Software if you require help in this direction. If you do inadvertantly misadjust any of the settings and can't return them to their correct positions, then if you return the Digitiser to us we will realign it for you at no charge.

Two preset potentiometers are available to set the horizontal and vertical shift positions of the picture. By shift positions we are referring to the part of the picture that is actually stored in the 512 * 256 pixel podule ram. Clearly the picture could be shifted around using the software commands, but all these hardware controls set the part of the picture that is actually stored in the podule ram. The vertical shift pot (VR5) sets which 255 lines of the picture are stored in ram. Each frame of video information actually consists of 625 lines divided into two fields each with 312.5 lines. About 24 of these 312.5 lines are not actually used to store picture information and this leaves 288 lines per field which actually contain the picture.

For convenience in organising the podule memory only 255 of these lines are actually stored and VR5 determines which line will be first to be stored. The range of adjustment is such that any consecutive 255 lines out of the 288 may be stored. The setting of this pot is initially arranged so that the 255 lines stored are those nearer the top part of the picture as this is more acceptable in terms of not losing the tops of peoples heads! In practice the height control setting of a normal TV set is usually arranged so that not all 288 lines are visible to prevent black areas at the top and bottom of the picture, and broadcasters allow for this when composing camera shots so that very little (if any) is lost by only storing 255 lines.

VR6 performs a similar adjustment in the horizontal direction and this is initially set so that there is equal loss of information on each side of the picture. Another adjustment, L6, is available which has a similar function to the 'width' on a normal TV set. Adjustment of this would allow the whole of the width of the picture to be stored in the 512 pixels available horizontally. However, for similar reasons to the reduced height setting (i.e. broadcasters allowing for overscanning on TV sets), this is initially set to complement the height value and give circular circles on pictures digitised with the default H/V scale settings.

L6 is also adjusted to minimise any patterning effects caused by the Digitiser clock oscillator beating with the PAL subcarrier oscillator and so this should be set carefully to ensure that no patterning results from it's adjustment.

All the above three adjustments can be made while viewing the picture in *Moving mode, but it should be repeated that they are unlikely to drift, and should really only be adjusted if you have a particular need to see the very top, bottom or edges of the picture.

The other five coils, C19 and VR4 can really only be adjusted with a colour bar input signal and an oscilloscope to view the results, so should only be attempted by someone who has experience at lining up a PAL decoder. For those who feel it necessary and need some guidance, a brief description of the function of each component is now given.

L2 is the subcarrier trap coil, it should be adjusted for minimum subcarrier while monitoring the right hand end of R19. L1 is a subcarrier bandpass coil and should be adjusted for maximum subcarrier while monitoring at the right hand end of R7. L3,4 & 5 and VR4 are controls for setting the balance and delay equalisation of the PAL delay line. They are adjusted for minimum 'twitter' while monitoring the red,green and blue decoder output signals. C19 adjusts the PAL subcarrier oscillator frequency and is best adjusted to reduce ' twitter' at the top of frame, i.e. adjsut this while looking at blue output in a frame sense on the 'scope.

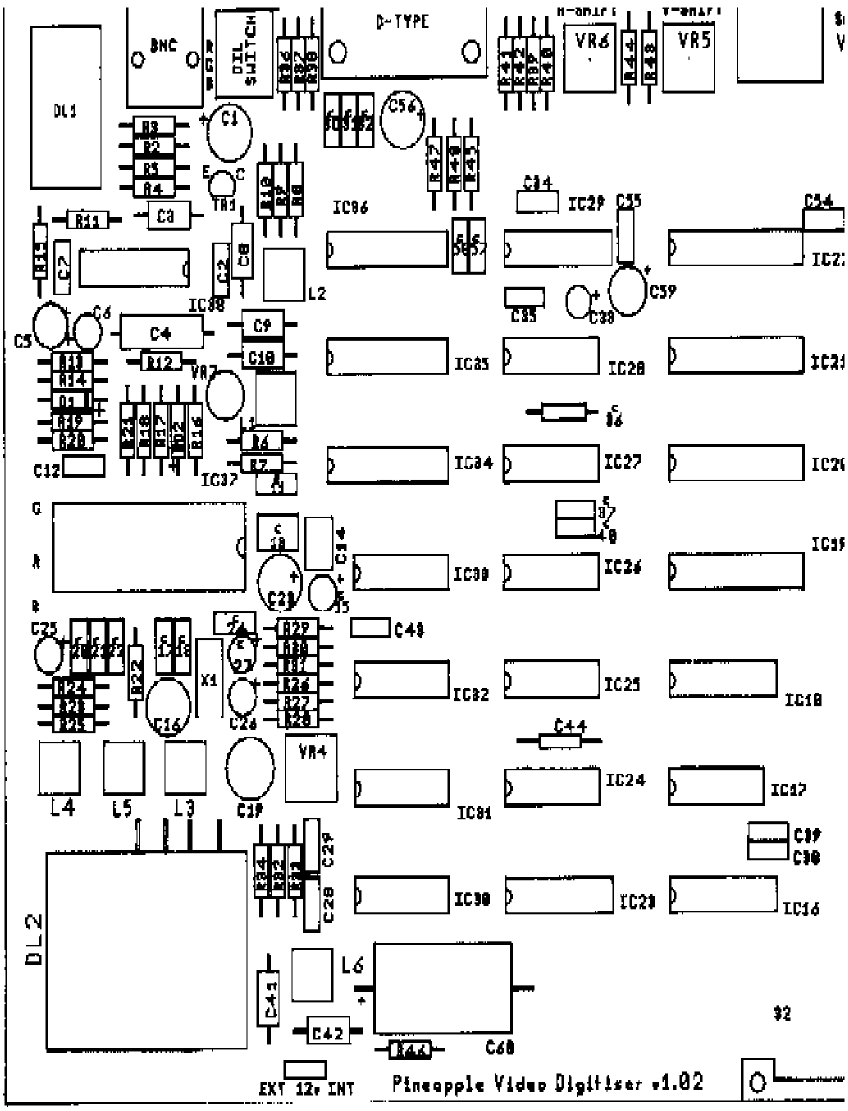
The only other adjustments on the board are provided by the four DIL switches. These switches allow the coded video and R,G,B inputs to be terminated if required. For those not familiar with the word 'termination', these switches simply connect a 75 ohm resistor between the input signal and earth. This is necessary with video circuits anywhere where the video is sent over long cables (longer than about 6 feet), to prevent reflections occuring in the cable. It also has the effect of halving the level of the input signal, usually from 2 volts peak to peak to 1 volt peak to peak.

The digitiser is quite happy working with PAL input signals from about 0.75 volts up to 2.5 volts pk to pk, as adjustment can be made with the gain control to allow for different input levels. Whenever possible though, it is recommended that a short video cable is used and the input signal is left unterminated, as the noise performance of the Digitiser is slightly better with a larger input signal.

When using the R,G,B inputs available on the 9 way 'D' type connector, sync pulses should still be fed into the PAL video input. The switch waveform to switch from the PAL video input to the R,G,B input can either be a simple +5v, or it can be a video speed switching waveform. This allows video switching between any two synchronised video signals where one is coded and the other R,G,B. Note that the gain control is not effective on the R,G,B inputs and external attenuators (probably just resistors in the 'D' type shell) must be used to set the R,G,B inputs to about .4v pk to pk.

HARDWARE SPECIFICATION

Coded Video I/P	Composite PAL video I/P via BNC connector.
	Min level 0.75 volts pk to pk.
	Max level 2.5 Volts pk to pk.
	High Z or 75 ohms switchable
	Wide range auto chroma control with manual adjustment from 0 to +15% normal.
R,G,B I/P's	Inputs via 9 way 'D' type connector.
	Pin 1 - Red
	Pin 2 - Green
	Pin 3 - Blue
	Pin 4 - Switch I/P (+5v)
	Pin 5 - Ext 12v input
	Pin 6 - 0v
	Pin 7 - 0v
	Pin 8 - 0v
	Pin 9 - 0v
	R,G,B I/P's 0.4 volts pk to pk (non-adjustable).
	High Z or 75 ohm switchable
	Sync I/P via BNC video I/P.
	High speed video frequency switch I/P switches between R,G,B and coded video.
Storage System	Standard Version
	512 * 256 pixel, 12 Bits per pixel.
	Organised as 4 bits Red, 4 bits Green, 4 bits Blue.
	Total podule memory 192k.
	Extended Version
	512 * 256 pixel, 16 Bits per pixel.
	Organised as 5 bits Red, 6 bits Green, 5 bits Blue.
	Total podule memory 256k.
12 volt supply	Provision is made for an 'external 12 volts power supply (250ma). This is fed via pin 5 of the 9 way 'D' type connector and switched by means of a link on the podule. This would allow the podule to operate on an A3000 system if a suitable external podule adapter box was available.
	N.B. Pineapple have now produced their own inexpensive adapter box for the A3000 which is connected to the computer via a ribbon cable and includes a 12 volt power supply. Please contact Pineapple for details.



DL1
BNC
R
G
B
DIL SWITCH
R36
R37
R38
D-TYPE
R41
R42
R39
R40
VR6
R44
R43
VR5
C1
C56
R3
R2
R5
R4
E
C
TR1
R10
R9
R8
R47
R46
R45
C34
IC29
C55
C54
R11
C3
IC36
R1
C7
C2
C8
L2
C59
C35
C38
IC38
C5
C6
C4
C9
C10
R13
R14
D1
R19
R20
R12
VR7
R21
R18
R17
R15
R16
IC35
IC28
R6
R7
C12
IC37
IC34
IC27
G
R
C14
C24
C25
R29
R30
R31
R26
R27
R28
C48
IC33
IC26
IC32
IC25
IC18
R24
R23
R25
R22
X1
C16
C26
C44
L4
L5
L3
C19
VR4
IC31
IC24
IC17
C29
C28
R34
R33
R32
C39
C30
IC30
IC23
IC16
DL2
L6
C41
C42
C60
$2
EXT 12v INT
Pineapple Video Digitiser v1.02