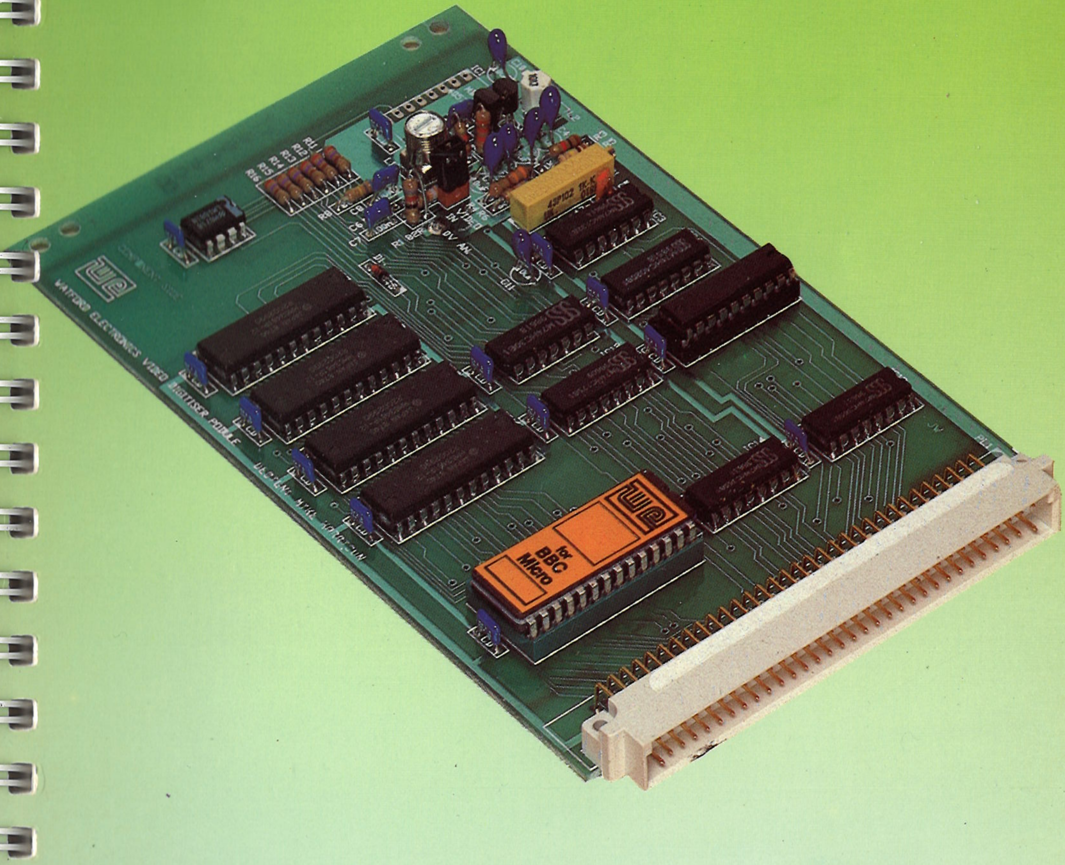
Archimedes Real-Time VIDEO-DIGITISER Operating Manual



Watford Electronics

Watford Electronics

Archimedes Video Digitiser

User Manual

Published in the United Kingdom by
Watford Electronics
Jessa House
250 Lower High Street
Watford
WD1 2AN
England

Telephone 0923 37774
Telex 8956095
Fax 01 950 8989

This book was computer typeset by
Ideal Software Consultants, 11 Hathaway Close, Luton, Bedfordshire .

Contents

1 Introduction

The Watford Electronics Video Digitiser is a very versatile add-on for the Acorn Archimedes range of computers, allowing video images from a variety of sources to be captured and manipulated.

Facilities are provided for grabbing images which can be displayed, rotated and/or scaled to any size and angle, saved to disk for future use, or printed out. It is also possible to grab sequences of pictures making up an animated sequence, and to obtain colour digitised images when using a video camera as the source.

The accompanying software disk contains utility programs, sample pictures, and several example programs which illustrate the use of many of the software facilities available.

The digitiser podule has 128K of on-board memory, which is used to store a picture, (this is referred to as 'digitiser memory'). Pictures are initially grabbed into this memory, and then transferred to the Archimedes' main memory (usually the screen), with optional scaling and/or rotation. It is also possible to use this memory as a general purpose picture store, as its contents may be transferred to or from disk or main memory.

1.1 Conventions used in this manual

Digitiser commands, system commands and listings are in a `mono-spaced` font.
Items inside <> are options to be replaced with the appropriate text. For example :

```
    *PicLoad <filename>
```

`<filename>` would be replaced with the name of the file to be loaded.
See section 3 for further information.

<> are also used to denote keys on the keyboard and mouse :

<left>	The left mouse button
<centre>	The centre mouse button
<right>	The right mouse button
<Ctrl>	The Ctrl key
<Return>	The Return key

etc.
Numbers preceded by & are in hexadecimal.

2 Installation

You must never install or remove podules when power is on!

If you are installing the digitiser into an A300 series machine, you will need to fit a podule backplane unless there is one already fitted. All A400 machines come with a backplane fitted as standard.

Remove one of the podule slot blanking plates from the rear of the machine by undoing the two screws. Fit the supplied blanking plate to the digitiser rear panel, so that the digitiser panel is on the left when viewed from the rear. (If you have a four slot backplane, the digitiser can be fitted on either side, and if you have another podule already installed, you don't need the blanking plate.) The podule should then be inserted so that the connector on the podule and the backplane are lined up, and then pushed firmly home to engage the connectors. The rear panel should now be flush with the rear of the case, and the screws can be replaced to secure it.

If your machine has an Arthur ROM version earlier than 1.1, ignore the rest of this section, and read section 17 instead.

Switch on the machine, and the message Watford Electronics Video Digitiser should appear with the usual sign-on messages. (If the machine is configured to go into the desktop at switch-on, the message will only appear very briefly before the desktop appears on the screen.)

If the message doesn't appear, type *RMReinit Digitiser. If this gives the error RM not found, check that the podule is correctly installed, and the power leads are correctly connected to the podule backplane. If you get the message No room in RMA, press <Ctrl>-<Break>, which should then display the sign-on message.

2.1 Connecting a video source

The digitiser will accept signals from most video cameras and video recorders which give a standard 1 volt video signal. The video source should be connected to the BNC socket on the digitiser's rear panel via an appropriate lead. Note that you must use the 'video out' connector on the camera or recorder, NOT the UHF (aerial) output.

When using a video recorder as the source, you will be able to digitise both off-air signals and tape recordings, although the former will usually give better quality. Poor quality recordings (this includes many pre-recorded tapes!) will be unusable on the digitiser, due to the poor quality of the synchronization signals.

If you don't have a video recorder, a tv tuner such as the Philips AV7300 may be used to obtain off-air signals for digitising.

If the video source being used (recorder or camera) has the facility to switch to monochrome output, this will generally produce cleaner images. When using off-air signals, you will usually find that image quality can be improved by mistuning slightly until the colour is lost, but most tuners and video recorders have some sort of automatic frequency control (AFC or AFT), which has to be switched off to enable this to be done - some units have a manual switch, others have a switch which disables the AFC when tuning flap or drawer is opened.

When a video source has been connected, the *See command can be used to display the incoming signal.

If the picture is dim or unstable, try removing the terminating link, LK1 on the podule (see section 16.3).

3 The digitiser module

This section contains general information about the digitiser software module.

All the driver software to control the digitiser is held in a ROM on the podule. The contents of this ROM are loaded into main memory as a relocatable module (RM) when the machine is reset (i.e. on power-up or <Ctrl>-<Break>).

The automatic loading may be disabled using *Unplug Digitiser, which sets a flag in CMOS RAM which will disable the module, and prevent it being loaded on subsequent resets or power-ups. You might want to do this to save memory when not using the digitiser. When *Unplugged, any image which was stored in the digitiser's memory will be lost.

*RMReinit Digitiser will re-enable the module after it has been *Unplugged. This may need to be done after installing the digitiser, moving it to a different podule socket, or if CMOS RAM as been corrupted or reset.

*ROMModules will show the current enabled / disabled state, and when enabled, the module name Digitiser will be listed by *Modules and *Help Modules.

Note for users of Arthur ROM versions earlier than 1.1
The automatic loading of the module will only occur under Arthur ROM versions 1.1 and later. If you have an earlier version, you will have to 'manually' load the module from the podule's ROM. See section 17 for information on how to do this.

The module provides access to the digitiser via key triggered grabbing, OS * commands, and SWI calls, which are described in detail in the appropriate sections of this manual.

3.1 * Commands

* Commands may be typed at the BASIC > prompt, the supervisor * prompt, or may be included in programs (using * or OSCLI in BASIC).

The following symbols are used to indicate the syntax of * commands :

`<n>`	A number
`<filename>`	A valid filename
`<spritename>`	A valid sprite name

`<options>`	One or more option letters.
`[ ]`	Indicates optional items
`|`	Indicates alternatives

Option letters may appear in any order. Usually, there must not be any spaces between option letters, but a few commands do require spaces, and these are indicated as such. All * commands, *Configure options and option letters may be in upper or lower case, or a mixture of both.

The digitiser module stores various configuration information in CMOS RAM, which may be altered using the *Configure command. Configurable options are described in the sections to which they apply. When the digitiser is first installed, or if CMOS RAM has been reset, the default settings are those listed with each option under 'CMOS Default'. The current state of configuration options can be found using *Status <option>. *Configure will change the options immediately, as well as storing them for later use.

Possible errors: The error messages which may occur are listed with each command. Some of the listed errors can only occur when certain combinations of options are selected. Filing system errors (Disc full, not logged on etc.) are not included, but may occur with any command which accesses files. The following possible errors also apply to most commands :

```
    Bad parameters, Bad number, Number too big.
```

* Command Examples (in BASIC programs) :

```
*Grab
OSCLI "Grab"
OSCLI "PicLoad "+name$
```

3.2 SWI Calls

SWI (Software Interrupt) is a machine code instruction which is used to access the Arthur operating system and its extensions (which include the digitiser software).

Many of the facilities provided by the digitiser software are accessed via SWI calls rather than * commands, as it is much easier to pass and return parameters, especially when calling from machine code programs. For applications where speed is important, SWIs are far more efficient than * commands, as they don't have to pass through Arthur's command line interpreter. SWIs have an additional advantage that they can be called from interrupt and event routines.

Although SWI is a machine-code instruction, BASIC provides access to SWIs via the SYS command. (See User Guide for more information on SYS)

Refer to the Programmer's Reference Guide for more information on SWIs.

Many digitiser SWIs use a register to pass several flags to indicate options, and where this is so, unused, or 'reserved' bits must always be set to 0. It is also advisable, where convenient, to set unused registers to 0 in order to ensure compatibility with possible future versions of the software.

Register bits are numbered in the usual convention of bit 0 being the least significant, bit 31 most significant.

Occasionally, one of the 32 bit registers are used to hold one or more bytes of information, in which case byte 0 is the least significant eight bits, byte 3 the most significant eight.

SWI Examples

Machine code

```
    MOV         R0,#0
    SWI         "Vdig_Grab"
```

BASIC

```
    SYS "Vdig_ScreenSwap",2
    SYS "Vdig_FastGrab",,,2   (same as 0,0,2)
    SYS "Vdig_VideoInfo" TO info
```

The 'X' (bit 17 set) version of the SWI can be used to avoid errors stopping a program :

```
 SYS "XVdig_Scale",65 TO e;f:REM Note semicolon, not comma!
 IF f AND 1 PRINT "ERROR - " ~!e; : SYS "OS_Write0",e+4
```

Bit 0 of f is set if an error occurred, e points to error number (1 word) and error message, terminated by 0

Note that the SYS statement sets unspecified registers to 0.

4 Display of digitised pictures

This section gives general information about displaying pictures, and how the digitiser makes use of the various graphics screen modes available.

The video digitiser hardware grabs images, which are stored internally as a 512x256 pixel image, with 64 grey levels per pixel. The Archimedes has various screen display modes, with varying numbers of colours and resolutions, and the digitiser software displays images in different ways, according to the screen mode selected.

4.1 256 colour screen modes

Due to the way in which the Archimedes' video circuitry works, the only way to display all 64 available grey levels is by using 256 colour mode, with a monochrome monitor. The quality of images displayed in this way is comparable to monochrome TV pictures, and for any serious image work, it is well worth getting a monochrome monitor if you only have a colour one. A more convenient alternative is to use a colour monitor such as the Philips CM8533 , which has both RGB and composite video (sometimes abbreviated to CVBS) inputs. If you connect the composite input to the Archimedes' monochrome video socket as well as using the usual RGB connections, you will be able to switch between colour and 64 shade monochrome modes easily using the monitor's RGB/CVBS switch on the front panel.

Owing to slight component variations between Archimedes machines, the screen pixel values required to get an even grey scale on an monochrome monitor vary from machine to machine. The digitiser software copes with this by storing correction factors in CMOS RAM, and uses these to calculate the displayed monochrome brightness of each displayed colour, which in turn is used to build an internal lookup table of screen colours to display for each of the 64 pixel intensities. The values of these correction factors are initially set using the program MCShades supplied on the disk, to determine the factors for your machine. Please refer to section 14.1 for instructions on using this program.

To use a normal (as opposed to high scan rate) monochrome monitor on an A400 series machine, you must first fit links LK13 and 14 on the main board, and connect the monitor to the Sync socket.

It is also possible to display 16 grey levels on both colour and monochrome monitors in 256 colour screen modes, and this can be useful if you want to add colour to an image, or use a software package which normally uses 256 colour modes. To select whether you want a 16 or 64 level display, There is a configuration option stored in CMOS ram, which may be altered using the *Configure Shades command:

Syntax : `*Configure Shades 16|64`

`*Configure Shades 16`
will produce 16 grey levels, which appear correctly on both colour and monochrome monitors
`*Configure Shades 64`
produces 64 grey shades on monochrome monitors only.

CMOS default is 64

4.2 16 colour screen modes

In 16 colour modes, digitised images are usually displayed using a palette which sets colours 0..15 to a scale from black..white. It is also possible to pass pixel values through a user supplied lookup table to obtain a good image using other palettes, or alter the mapping of the 64 digitised levels to the 16 screen grey levels.

Use of 16 colour modes has the advantage that good images can be obtained using both monochrome and colour monitors, and also that all the various digitiser display routines work much faster in these modes, allowing display of fullscreen images at up to 12.5 fields per second, or quarterscreen at 25 per second.

4.3 2 colour modes

Most (but not all) of the digitiser display routines will display images in 2 colour screen modes, using dot patterns of varying density to represent the different brightnesses. The dot patterns used are selectable from 6 pre-defined sets, or may be user defined using the pattern editor supplied on the disk. The patterns used are the same ones used for printer dumps, and are described in section 11.

Four colour modes are not supported.

4.4 Resolution

The 512 x 256 pixel image from the digitiser is automatically scaled to give the correct aspect ratio (the ratio of picture width to height) in the screen mode being used. High resolution modes 18 and 20 (using MultiSync type monitors) are supported, usually by duplicating vertical lines, although it is possible to grab a true 512 line frame in mode 20, as long as the image is stationary whilst grabbing.

5 Key triggered grabs

To make it easy to use the digitiser with painting or graphics packages (e.g. Clares' Artisan), a facility is provided to grab and display images when <Ctrl> and another key is pressed, while another program (i.e. the drawing package) is running. This powerful facility makes the digitiser compatible with most painting type (as opposed to object-oriented) drawing packages.

5.1 *Configure GrabKey

The key used to trigger the grab is configurable, the key code being held in CMOS RAM, and can be set using *Configure GrabKey.

Syntax : `*Configure GrabKey [<n>]`

`*Configure GrabKey`
on its own will prompt the user to press <Ctrl> and the required key. Alternatively,
`*Configure GrabKey <n>`
will set the key to the one which produces Ascii code <n> when used with <Ctrl> (e.g. 1 is <Ctrl>-A)

Note that the key-triggered grab software checks that <Ctrl> is actually pressed, so it's possible to use a key which normally doesn't produce a different code when used with <Ctrl> (e.g. <Tab>, <Return>, <Space> or number keys).

Keys which don't usually produce characters (i.e <Shift> <Alt> <Esc> <Scroll Lock> <Break> <Num Lock> <Caps lock> <Copy> and the cursor editing keys) cannot be used as the trigger key. <Ctrl>-A or <Ctrl>-<Tab> are usually the most convenient combinations to use. Remember that the control key combination selected will be disabled for normal use, as the key code will not enter the keyboard buffer. For example, if <Ctrl> A is selected, you will not be able to use <Ctrl> A in the Basic Editor. (This is why the grab key is configurable!)

CMOS default is 0 (<Ctrl> @)

5.2 *Configure GrabOptions

There are various configurable options to select whether you want to display a fullscreen picture, or display between 2 mouse pointer positions. Other options allow the palette to be automatically set up in 16 colour modes, and a zoomed or reduced section of a previously grabbed image to be displayed. Options are set using the *Configure GrabOptions command :

Syntax : `*Configure GrabOptions [F|W|M|C] [S] [S] [G] [L] [D] [S]`

The main options are :

F	- Fullscreen Grab
W	- Fill the current graphics window
M	- Plot between the last 2 points where a mouse button was pressed
C	- Plot picture between the last 2 graphics cursor co-ordinates.

These first 4 options are alternatives, and if none is selected, F will be used.

The other options are :

S	- Use section of picture previously marked by *Section
P	- Set palette if in 16 colour mode, or invert colours in 2 colour modes
G	- Don't grab - use the existing picture in digitiser memory
L	- Use pixel lookup table
D	- Disable key triggered grabs completely
A	- Apply *Smooth once before displaying (see section 10.1)

If no options are given, only F will be set.

In 2 colour modes, the P option is used to invert the shades, allowing the correct image to be displayed when the screen colours have been reversed using the palette. The dot pattern set used will be that set by *Configure PatternSet (see section 11.4)

Examples :

```
    *Configure GrabOptions FP
```
Fullscreen, set palette in 16 colour modes
```
    *Configure GrabOptions MA
```
Smooth, and plot between last 2 mouse click positions

CMOS default is F

Great care must be taken when using this facility, as programs don't generally expect things to appear on the screen unexpectedly! In general, never use this facility when there are pop-up windows on the screen, as a section of the picture will disappear when the window disappears or is moved. Also, avoid using key-triggered grabs whilst something is being drawn on the screen, (e.g. when 'rubber-banding' a line or shape). See also the notes on calling digitiser SWIs from background tasks in section 16.2.

The keypress will be ignored if the current screen mode is not a 2, 16 or 256 colour graphics mode, or if there is no incoming video signal.

There is more inforation on using key triggered grabs with specific art packages in section 18.

6 Grabbing and displaying images

See also section 7 which documents the facilities for displaying scaled and rotated images.

Display of a digitised image requires 2 actions - Grabbing the incoming video signal into the digitiser's memory, and then transferring it to the screen for display. Many of the digitiser display routines perform both of these operations.

6.1 *Bwpal

To display images in 16 colour screen modes (2,9,12,20), the colour palette has to be set up to display 16 shades of grey instead of 16 colours. Some commands (e.g. *Grab, *See) set up this palette automatically, but a command, *Bwpal, is also provided to do this for you.

Syntax : `*Bwpal [2|4|8]`

*Bwpal 2,4 or 8 sets palette to display only 2,4 or 8 of the 16 grey levels.
E.g. *Bwpal 2 will show the brightest 8 colours as white, the darker ones as black.

6.2 *Grab

The *Grab command grabs a field & displays it in the current screen mode, setting the correct palette in 16 colour modes.

Syntax : `*Grab [Q][P][O|E|I]`

Options

Q	Display quarterscreen
P	Don't set palette
O	Wait for an odd field
E	Wait for an even field
I	In mode 20 (using a MultiSync type monitor), grab both fields and display as a true 640*512 picture. Ignored in other screen modes.

Possible errors : `Can't in this mode, No video signal.`

6.3 *See

This command continuously grabs and displays fields until a key or mouse button is pressed. Screen bank swapping is used to reduce visible flicker, providing sufficient screen memory is available.

Syntax : `*See [P][M][O|E|I]`

Options

M	Use the current screen mode (otherwise uses mode 9)
P	Don't reset palette
O	Wait for an odd field
E	Wait for an even field
I	In mode 20 (using a MultiSync type monitor), grab both fields and display as a true 640*512 picture, ignored in other modes.

If the M option is used, or the current screen mode is 9, the screen mode will not be reset. This allows the mouse pointer (if enabled) to stay on the screen while pictures are being grabbed. Note, however, that because the grabbing software disables interrupts for some of the time, pointer movement will become sluggish if the mouse is moved quickly, especially when using the quarter-size (Q) option.

`*See` cannot be used in 2 colour modes.

When using certain combinations of size and screen mode, when the incoming video signal is interlaced, there will be a slight vertical jitter, due to alternate odd and even fields being grabbed. This can be avoided by using the O or E options, at the cost of slower screen update.

Note that when using the Q options for *Grab and *See, the image in digitiser memory will not be a complete one, as only half of the memory is used for speed reasons. There is more information on this in section 6.8

Possible errors : `Can't in this mode, No video signal, Bad MODE`

6.4 *ShowPic

*ShowPic displays the picture currently in the digitiser's memory, using the current screen mode.

Syntax : `*ShowPic [S][P][L]`

Options

S	Shows the section previously selected using *Section (or SWI Vdig_Section), zoomed to full screen size
P	Don't reset the palette
L	Use pixel lookup table

See the next section for information on *Section

Possible errors : `Can't in this mode`

6.5 *Section

This command allows part of a picture to be selected for use by key triggered grabs, *ShowPic, and SWI Vdig_Scale. The selected section will be expanded to fill the whole screen.

Syntax : `*Section [M][P]`

*Section displays the current picture from the digitiser's memory on the screen, and uses the mouse to select a section of it. A box is displayed over the picture, representing the currently selected section (which will default to the full screen if no section was previously selected), the size and position of which can be adjusted using the mouse.

The mouse controls the position of the corner of the box indicated by the pointer, which may be flipped to the opposite corner with <left>. A new image can be grabbed by pressing <right>, and <centre> exits, storing the new section. Pressing <Escape> will exit without changing the current section.

Options

M	Use current screen mode (mode 9 is used if M omitted)
P	Don't set palette

Possible errors : `Can't switch pointer on, Can't in this mode, No video signal`

SWI Vdig_Section &802CF

This SWI provides an alternative to the *Section command, and also allows the current section to be read.

On entry

R0	- 0 read, 1 write
R1,2	- x,y co-ordinates of first corner
R3,4	- x,y co-ordinates of opposite corner

On exit

R1..4	- Previous or new co-ordinates

Example

```
    SYS "Vdig_Section",,,640,512
```

Selects the bottom left-hand quarter of the image.

6.6 *MakeSprite

This command provides a simple means of creating a sprite from all or part of a digitised image.

Syntax : `*MakeSprite <spritename> [P][S][D][L][C]`

The current image in digitiser memory is shown, and its size may be adjusted using the mouse. The mouse buttons have the following functions:

<left>	Flips the image in x and y directions alternately
<centre>	Creates the sprite, and exits
<right>	Grabs new images for as long as it is held down
<Escape>	Exits without defining the sprite

Options

P	Don't set palette
S	Use section (enters *Section first to define section)
D	Display the current image size onscreen
L	Use pixel lookup table
C	Include palette data in the sprite

If S is selected, *Section is performed first to select the required part of the image (see section 6.5 for mouse buttons), before entering the usual MakeSprite screen. If the S and L options are both selected, the lookup will NOT be applied to the picture when selecting the section, only when adjusting the picture size.

The D option shows the current width and height of the sprite will be shown at the top left-hand corner of the screen. The text will be in white unless the P option is also selected, in which case it will be in the current text colour.

The C option includes a copy of the current palette in the sprite, which will be used if it is *ScreenLoaded.

Note that when grabbing continuously (by holding <right> down), the mouse response will become sluggish when the picture is small, due to interrupts being disabled during grabbing.

When the sprite has been created, it is selected as the 'current sprite', for plotting with PLOT &ED,X,Y.

Possible errors : `Can't in this mode`, `Can't switch pointer on`, `No video signal`, `No room to get sprite`, `No sprite memory`.

6.7 SWI Vdig_Grab &802C3

This SWI call grabs an image to the digitiser's memory, without displaying it. (for subsequent use with *ShowPic, SWI Vdig_Scale etc.)

On entry

R0 -	0 Grab any field
	1 Reserved
	2 Grab an even field
	3 Grab an odd field

Note that the interlace options will not wait for ever for an odd or even field : If the first field encountered is not the one requested, the next one will be grabbed regardless. This is to avoid hang-up problems if the picture is not interlaced.

Example
```
        SYS "Vdig_Grab",2
```
Grabs an even field

Possible Errors : `No Video Signal.`

6.8 SWI Vdig_FastGrab &802C4

This call will grab and display a picture as fast as possible in any 16 or 256 colour graphics screen mode, using special grab routines which have been highly optimised for speed. This call is particularly suitable for grabbing sequences of fields in rapid succession for display as a moving sequence.

Four picture size options are available : full or quarter screen, with or without scaling.

As the digitiser's internal horizontal resolution is 512 pixels, and the Archimedes screen is usually 320 or 640 pixels wide, scaling is used to fill the screen and maintain the correct aspect ratio. This scaling can be disabled, so the image is displayed pixel-for-pixel, and appears slightly squashed.

On entry

R0	Picture position on screen, or absolute address.	
R1	Target screen mode, or 0 to use the current screen mode.	
R2	Flags :	
	bit 0	0 For Fullscreen, 1 for quarter screen
	bit 1	0 To scale x axis up by 1.25, 1 To plot pixel-for-pixel
	bit 2	1 To change the end-of-line screen padding
	New value (in bytes) in R2 bytes 2,1	
R3	bits 1,0	00 Grab any field 01 Reserved 10 Grab an even field 11 Grab an odd field
	bit 2	Only relevant for fullscreen in mode 20, using a MultiSync type monitor : 0 Grab one field, duplicating lines 1 Grab both fields to display true 512 line picture (bits 1,0 ignored)

If bit 31 of R0 is clear, R0 is used as a byte offset from the top left-hand corner of the screen. When bit 31 is set, R0 is used as an absolute address (with bit 31 masked out) to store the screen image. The offset or address must be word aligned, except for quarter screen grabs in modes 2,9,10 or 13, where it may be byte aligned.

If R1 <> 0, Data for any target screen mode can be generated, regardless of the current mode. This is mainly of use when grabbing sequences of fields to memory. Note, however, that the grab routines have been optimised to work in their own screen mode, and if a low resolution mode is specified in R1, and the current mode is a higher resolution one, problems may be encountered due to the extra memory access caused by the higher screen update rate (this may appear as a vertically warped image). To avoid this, always ensure that the current mode uses the same, or a lower amount of screen memory than the target mode (refer to User Guide for amounts of memory used by each mode).

When displaying pictures which don't fill the full width of the screen, padding is normally added to the end of each horizontal line :

```
********************************
*<---image---><---padding--->*
*                             *
*<------screen width-------->*
*                             *
```

By setting bit 2 of R2, The amount of this padding can be altered, to enable image data to be dumped to a contiguous block of memory, particularly useful when grabbing sequences. The amount of padding must be specified in bytes (not pixels), and defaults to (ScreenWidth-PictureWidth). The amount of padding must be an exact number of words (i.e. a multiple of 4 bytes), except for quarter screen grabs in modes 2,9,10,13, where it may be a byte value.

Grabbing rates
This table shows typical field grabbing speeds in the various screen modes using the BASIC line :

```
REPEAT : SYS"Vdig_FastGrab",,,n : UNTIL FALSE
```

Speeds are in fields per second. Note that any significant delay between calls will slow these rates down, as will any significant amount of background processing (e.g. mouse interrupts)

	Full screen		Quarter screen	
	Scaled (n=0)	Unscaled (n=2)	Scaled (n=1)	Unscaled (n=3)
MODE				
2/9	12.5	12.5	25	25
10/13	6	7	12.5	12.5
12/14	7	10	16.7	16.7
15	3.6	4	10	10
20	5.5*	7*	10	10

* speed is approximately halved for interlaced grabs (R3=4).

Note that some mode/size combinations result in speeds which are odd multiples of the 50Hz (fields per second) rate (e.g. mode 12 scaled quarterscreen, which will grab every third field). If an interlaced signal source is used, this will result in slight vertical jitter, as alternate odd and even fields will be grabbed. This can be avoided by setting R3 to 2 or 3, so as to only grab even (2) or odd (3) fields. This will, of course, reduce the grabbing rate to the nearest even multiple of the field rate (12.5/sec in the above example).

When grabbing to a memory address other than the default screen (i.e.R0<>0), the address supplied is NOT checked for validity, to allow maximum flexibility (e.g. for writing to the screen in physically mapped RAM, or direct to I/O devices). Because of this, if an invalid address is supplied, an address exception or data transfer abort may occur.

Note that the quarter-screen grabs use only half of the digitiser's memory, so the image stored held in memory will not be a complete one, and this should be remembered when using routines such as SWI Vdig_Scale and *ShowPic. The image in memory after a quarter screen grab will appear split : the top half will contain the grabbed image, squashed to half vertical height, but full width, and The lower half will be the bottom half of the last full image which was grabbed or loaded.

SWI Vdig_FastGrab will not work in 2 colour modes, but if the current mode is a 2 colour one, and r0,1,2 are all 0, the call will be redirected to SWI Vdig_Scale to do a fullscreen grab and display using the default dot patterns (as set by *Configure PatternSet), otherwise a Can't in this mode error will result.

Examples

```
SYS "Vdig_FastGrab"
```
Grabs a fullscreen image to the current screen
```
SYS "Vdig_FastGrab",10280,,1
```
Grabs quarterscreen, displayed at screen centre (in mode 9)
```
SYS &802C4,buffer% OR &80000000,,5
```
Grabs a quarterscreen image to memory at buffer%
(using SWI number instead of name is a bit faster)
The next 2 examples are for mode 9 only.
```
SYS "Vdig_FastGrab",,12,1
SYS "Vdig_FastGrab",160,12,1
```
Grabs 2 fields, and 'mixes' them by displaying alternate lines from each.
```
SYS "Vdig_FastGrab",,,&F005
```
Displays quarterscreen image using alternate lines, stretching it to full height.

There are more examples in some of the programs on the disk.

7 Scaling and Rotation

This section describes the facilities for altering the size and/or orientation of an image before displaying it on the screen.

These facilities are provided via 2 SWI calls, SWI Vdig_Scale, for scaling, stretching and squashing, and SWI Vdig_Rotate, which will rotate to any angle, as well as re-scaling. Both these calls take the image currently in digitiser memory (or grab a new one if required), apply the necessary transformations, and then display the image onscreen. The image in digitiser memory is not altered (except when the smooth option is specified for SWI Vdig_Scale).

Pixel intensity values can be passed through a user supplied lookup table before being displayed. This allows such operations as contrast enhancement, pseudocolouring, thresholding and matching the image to non-standard screen palettes. Note that when using a user lookup table in 16 colour modes, speed is approximately halved (there is no difference in 256 colour modes). Lookup tables are fully described in section 8.

Images may optionally be plotted on the screen using the AND, OR, and Exclusive-OR plotting modes to combine the picture with the current screen.

7.1 SWI Vdig_Scale &802C6

This call extracts all or part of an image from the digitiser's memory, and displays it on the screen at any specified size, scaling up or down as necessary. The image may also be flipped horizontally and/or vertically.

SWI Vdig_Scale can display images in 2 colour graphics modes (modes 0,4,18) using dot patterns of varying density to represent the different grey levels. The dot patterns used are selectable from a fixed set, or user definable. The same patterns are used by the screendump software, and the method of selecting or defining patterns is described in section 11.

The size of the displayed image can be specified in 4 ways : fill the screen, fill the current graphics window, fill the area between the last 2 graphics cursor positions, or fill the area between the last 2 mouse positions where a mouse button was pressed. For the last 2 options, you can specify whether or not to clip the image to the current graphics window. Note that this window clipping is very fast, so parts of a picture can be 'painted' onto the screen by moving the graphics window around the screen. The last 2 options also allow part of the picture to be off the edge of the screen, in which case it will be clipped, without affecting the scaling.

On entry

R0	bits 1,0	00 Don't grab - use current picture in memory
		01 Grab any field
		10 Grab an even field
		11 Grab an odd field
	bits 3,2	00 Use whole digitised image
		01 Reserved
		10 Use section of image, set using *Section
		11 Use section of image specified in R1..4
	bit 4	1 For horizontal flip (mirror image)
	bit 5	1 For vertical flip (upside down)
	bit 6	0 Plot picture between last 2 graphics cursor positions (unless R0 bit 11 set)
		1 Plot to fill current graphics window (fullscreen if R0 bit 10 also set)
	bit 7	1 Use pixel lookup table

bits 9,8 - Plot mode :

00 Plot normally
01 OR with screen image
10 AND "   "   "
11 EOR "   "   "

bit 10	0 Clip image to the current graphics window
	1 Ignore the graphics window

bit 11 - Only relevant when bit 6 is clear

0 Plot picture between the last 2 graphics cursor positions
1 Plot between the last 2 mouse positions where a mouse button was pressed

bit 12 - Perform *Smooth before displaying

bits 18,17,16* - Pattern set number (see section 11.4 for pattern numbers)

bit 19*   1 To invert intensities (black<-->white)

* Bits 19..16 are only relevant in 2 colour modes.

If R0 bit 2 =0 and bit 3 =1, R1 to R4 specify a section of the image to use :

- R1 - X co-ordinate of first corner
- R2 - Y co-ordinate of first corner
- R3 - X co-ordinate of opposite corner
- R4 - Y co-ordinate of opposite corner

These co-ordinates may lie outside the normal co-ordinate range, in which case, the digitiser image will be repeated. For example, co-ordinates of 0,0 and 2560,2048 would display 4 copies of the image.

Examples

```
        SYS"Vdig_Scale",64
```
Display image filling current graphics window
```
        SYS"Vdig_Scale",&4D,0,0,2560,2048
```
Grabs and displays 4 copies of the image in the current graphics window
```
        SYS"Vdig_Scale",&4C,320,256,960,768
```
Zooms the centre of the image to double size
```
        SYS"Vdig_Scale",&20070
```
In 2 colour modes, displays image upside down using diagonal line patterns

There are several more examples of the use of this call in the example and utility programs on the disk.

7.2 SWI Vdig_Rotate &802CA

SWI Vdig_Rotate rotates and scales an image from digitiser memory and plots it on the screen.

The image may be rotated about its centre, or any specified point (including points off the edge of the image), and scaled up or down by a given factor, preserving the correct aspect ratio. The image is plotted so that the specified centre of rotation appears at the current graphics cursor position. The image will be clipped to the current graphics window if required.

SWI Vdig_Rotate will not work in 2 colour screen modes.

On entry

R0	bits 0,1	00 Don't grab - use current picture in memory 01 Grab any field 10 Grab an even field 11 Grab an odd field
	bit 2	0 Rotate about centre 1 Rotate about point in R3,4
	bits 3,4,5	Reserved (must be set to 0)
	bit 6	0 Plot the rotate point at graphics cursor 1 Plot the rotate point at centre of screen
	bit 7	1 Use pixel lookup table
	bits 9,8 - Plot mode :	00 Plot normally 01 OR with screen image 10 AND " " " 11 EOR " " "
	bit 10	0 Clip image to the current graphics window 1 Ignore the graphics window

R1	Angle to rotate in degrees (clockwise). R1 MOD 360 is used, and the angle may be negative.
R2	Scaling factor, multiplied by 4096 (e.g. 1024 is 1/4 linear size, 8192 is 2x) If R2=0 the image will be actual size (4096)

R3,4 are only relevant if bit 2 of R0 is set :

R3	X co-ordinate of the point on the digitised image to use as centre of rotation
R4	Y co-ordinate of centre

Examples

```
SYS"Vdig_Rotate",64,20
```

Displays fullscreen, rotated 20 degrees about centre

```
MOVE 0,0 : SYS"Vdig_Rotate",4,-10
```

Rotates 10 degrees anticlockwise about the bottom left-hand corner

```
SYS"Vdig_Rotate",64,45,2048
```

Half size, centrescreen, rotated 45 degrees about centre

```
MOVE1279,0:SYS"Vdig_Rotate",4,35,8192,1279,512
```

Magnify 2x, rotate 35 degrees about the centre of the right-hand edge, plotting the rotate point at the bottom right-hand corner of the screen.

The example program 'Rotate' on the disk shows some of the effects which can be obtained using this call.

8 Pixel lookup tables

Certain digitiser display routines (*ShowPic, SWI Vdig_Scale, SWI Vdig_Rotate, Key triggered grabs) allow pixel values to be passed trough a user supplied lookup table before being displayed. This is a very powerful and flexible facility, which has a variety of uses, including contrast enhancement, pseudocolouring, thresholding, and the matching of an image to suit an existing palette in 16 colour modes. Lookup tables are also used by the ColourGrab program to separately write to the red, green, and blue bits in displayed pixels to form a colour image.

There are four lookup tables, each 64 bytes long, three of which can be altered by the user.

16 colour lookup table
This is used in 16 colour screen modes when lookup is selected using the L option for *ShowPic, or the lookup flag bit for SWI Vdig_Scale and Vdig_Rotate. It holds the 4 bit logical colour values to be written to the screen for each of the possible 64 digitised pixel intensities. In 2 colour modes, it holds the number of the dot pattern to use for each intensity level. This table initially holds values from 0 to 15, giving the same display which would be obtained without lookup.

256 colour lookup table
This table is used when lookup is selected in 256 colour modes. It holds the byte values written to the screen for each of the 64 intensity levels. The relationship between screen bytes and logical colours is not as straightforward as in 16 colour modes :

Screen byte bits	7	6	5	4	3	2	1	0	
Logical colour bits	5	3	2	1	4	0			
Tint bits							1	0	
Red bits				3		2	1	0	(Default palette)
Green bits		3	2				1	0	(")
Blue bits	3				2		1	0	(")

This table initially holds a copy of the default intensity table.

256 colour intensity table
This table is used by all 256 colour display modes (including *Grab, *See and SWI Vdig_FastGrab) when not using a lookup option. It initially contains a copy of the default intensity table.

Default intensity table
This table holds a reference copy of the correct screen pixel values to display in 256 colour screen modes. Its contents are determined by the setting of the *Configure Shades option.
In 16 Shades mode, it holds the screen pixel values for each of the 16 displayable grey levels, i.e. bits 0,1 come from bits 2 and 3 of the intensity value, bits 5,3 and 2 from bit 4, and bits 7,6 and 4 from bit 5.

In 64 shades mode, it holds a list of screen byte values in ascending order of brightness when viewed on a monochrome monitor. The exact contents are determined using the correction factors set up in CMOS RAM using the MCShades program. This table is readable, but not writable by the user.

It is possible to define a 256 colour table in terms of the intensity table, so for applications which just need to define intensities rather than colours or byte values, you can just specify an intensity in the range 0..63, and the software will work out what the actual screen value should be.

Note that all lookup tables are reset to their default contents when the *Configure Shades command is used.

8.1 SWI Vdig_Tables &802C7

This call reads or changes the contents of a lookup table.

On entry
R0 - Reason code :

0	Reset lookup tables & intensity table to defaults
1	Define 256 colour lookup table via intensity table
2	Define 16 colour lookup table
3	Define intensity table, via default intensity table
4	Fill 256 colour lookup table with byte values 0..63
5	Define 256 colour lookup table as screen bytes
6	Reserved
7	Define intensity table as screen byte values
8	Read current tables

R1 - Address for tables :

When defining a table, R1 is address of user's new table (64 Bytes)

When reading tables (R0=8), R1 is address to copy current tables to (256 bytes) :

(R1)	-> intensity table
(R1)+64	-> 256 colour lookup table
(R1)+128	-> 16 colour lookup table
(R1)+192	-> default intensity table

Examples

The following example sets up the 256 colour lookup table to give a negative image:

```
DIM table% 64
FOR a%=0 TO 63 : table%?a%=(a% EOR 63) : NEXT a%
SYS"Vdig_Tables",1,table%
*ShowPic L
```

This one sets up a 16 colour table which squares the pixel values, emphasising the brighter parts of the image :

```
DIM table% 64
FOR a%=0 TO 63 : table%?a%=(a%*a%) DIV256 : NEXT a%
SYS"Vdig_Tables",2,table%
*ShowPic L
```

(The result is divided by 256 to give a value in the range 0..15)

For some more examples of using tables, look at the programs ColourGrab, Threshold and MCShades on the disk.

8.2 *TableSave

This command saves the current intensity table and user lookup tables to a file.

Syntax : `*TableSave <filename>`

The file format is 192 bytes of information, in the same order as returned by SWI Vdig_Tables with R0=8.

8.3 *TableLoad

*TableLoad Loads new intensity and/or pixel lookup tables.

Syntax : `*TableLoad <filename> [I|U][S[S][P][L]]`

Options
Sxxx Do *ShowPic afterwards xxx are ShowPic options, which would normally include L if loading a user pixel lookup table.

I	Only load intensity table
U	Only load user pixel lookup tables

Table files have an Acorn allocated filetype of &DFB (LookUp), and to make use of this, two system variables are set up by the digitiser module when it is initialised:

```
    Alias$@LoadType_DFB  TableLoad %0
    Alias$@RunType_DFB   TableLoad %0 SL
```

This allows tables to be loaded using *Load <filename>, and *<filename> will display the image currently in digitiser memory in the current screen mode, via the new table. Note that if the current mode isn't suitable for displaying images, the Can't in this mode error will result, but the new table will have been loaded.

Remember that loading a table file will usually overwrite the 256 colour intensity table (unless the U option is given), thus overriding the effect of *Configure Shades.

Possible errors : `Not a lookup table file`

There are a number of example table files in the 'Tables' directory on the disk:

Default	- default setting of 16 and 256 colour lookup tables.
Negative	- Negative versions of 16 and 256 colour tables.
Pal0_7	- For 16 colour mode palettes with colours 0..7 defined as black-white.
Pal7_0	- Colours 7..0 = black-white
Pal8_15	- Colours 8..15 = black-white
Pal15_8	- Colours 15..8 = black-white

9 Loading and Saving images

Digitised images may be saved to a filing system (i.e. ADFS, Econet) in one of two ways : The picture currently on the screen (having been displayed by one of the various digitiser display routines) can be saved, or the image currently in the digitiser's memory can be stored.

The first method is fine if you will only want to use the image exactly as it appears onscreen (e.g. for loading into an art package), or to use it on a machine which doesn't have a digitiser fitted, but the second method offers more flexibility, as you can reload the image into the digitiser's memory, and use all the facilities provided by the digitiser software (scale,rotate,smooth etc.) to change the way that the image is displayed.

Saving and loading screen images
Saving and loading of screen images can be done using the *ScreenSave and *ScreenLoad commands provided by Arthur's Sprite utility module, but as these commands are designed to be able to load and save both full and partial screen images, they are rather slow, especially when using floppy disks. The digitiser module provides much faster alternatives to these commands for use with fullscreen images, while using the same file format.

9.1 *FastSave

Saves the current screen image and palette in ScreenSave format.

Syntax : `*FastSave <filename>`

Possible errors : `Not a graphics mode`

9.2 *FastLoad

Loads a screen image and palette from a file previously saved with *FastSave or *ScreenSave.

Syntax : `*FastLoad <filename>`

Possible Errors : `Not a full screen ScreenSave file, bad MODE`

*FastSave files are compatible with *ScreenLoad, and *ScreenSave files can be loaded with *FastLoad, as long as the file is a full screen image rather than a section of the screen.

When using floppy disks, *FastLoad / *FastSave are about 5 times faster than *ScreenLoad / *ScreenSave.

Note that it is possible to redirect *ScreenSave and *ScreenLoad commands to use FastLoad and FastSave automatically, this being useful for software packages such as Clares' Artisan which use these commands to load and save fullscreen images.

This can be done using the commands :

```
*Set Alias$ScreenLoad FastLoad %0
*Set Alias$ScreenSave FastSave %0
```

Doing this will also allow images to be *FastLoaded using *<filename>.

Due a bug in Arthur versions up to and including 1.20 (Econet module v5.12, NetFS v5.14), the *FastLoad/*FastSave commands may not work on Econet when using certain fileserver types and versions, giving a no reply error.

9.3 *GrabSave

This command is provided to simplify the process of grabbing and saving screen image files.

Syntax : `*GrabSave <filename> [P][O|E|I]`

*GrabSave displays the image currently in the digitiser's memory, in the current screen mode, and then waits for a key to be pressed :

<Shift>	grabs and displays new images
<Return>	saves the currently displayed screen image as <filename>.

Any other key quits without saving.

Files are saved using the *FastSave command, and so may be loaded with *FastLoad or *ScreenLoad.

Options

P	Don't reset palette
O	Wait for an odd field
E	Wait for an even field
I	In mode 20 (using a MultiSync type monitor), grab both fields and display as a true 640*512 picture. Ignored in other modes.

Possible errors : `Can't in this mode, No Video signal`

Saving and loading digitiser memory
Two Commands are provided to load and save images to or from the digitiser's memory. The image is saved using a simple data compression system, which helps to reduce the amount of disk space required.

9.4 *PicSave

Saves the picture currently in digitiser memory. If there is not enough space on the disk, a disc full error will result, and only part of the file will have been saved: part of the screen will be filled with garbage if you try to re-load the incomplete file.

Syntax : `*PicSave <filename> [Z]`

If the Z option is specified, *Zit (see section 10.2) will be performed first, in order to reduce the size of the saved file by removing spurious spots in the image.
The size of *PicSave files is dependant on picture content, typically around 50-80K, and the absolute maximum size is 128K.
The Z option will reduce the file size by anything up to about 30K for noisy pictures, with very little visible picture degredation.

9.5 *PicLoad

Loads an image from file into digitiser memory, optionally combining the image file with the one already in memory.

Syntax : `*Picload <filename> [S[S][P][L]] [O|A|E|M|P|D|R|X|G|L|U]`

Options

Sxxx Do *ShowPic afterwards - xxx are *ShowPic options

O	OR the file with the existing picture
A	AND the file with the existing picture
E	Exclusive-OR the file with the existing picture

M	Average (mix) the 2 pictures
P	Add pictures
D	Subtract the new picture from the old one
R	Subtract the old picture from the new one
X	Multiply pixel values, and divide by 64
G	Replace pixel values higher than those in the old image
L	Replace pixel values lower than those in the old image
U	User defined operation

For option P, intensities greater than 63 are set to 63, for D and R, negative results are set to 0.

PicSave files have an Acorn-allocated filetype of &DFA ('Picture'), and the digitiser software sets up 2 system variables to make use of this :

```
     Alias$@LoadType_DFA PicLoad %0
     Alias$@RunType_DFA  PicLoad %0 S
```

This means that *Load <filename> will load <filename> into digitiser memory, and *<filename> will load and display it in the current screen mode.

If the S option is used, and the current mode is not suitable for displaying images, the Can't in this mode error will result. The new image will, however have been loaded into the digitiser's memory.

Possible errors : `Can't in this mode`

User defined operations
User defined logical or arithmetic operations between the loaded image and the previous one can be performed using the U option of *PicLoad, by copying 4 words of machine code into the user code area using SWI Vdig_UserLoadCode.

The code must be exactly 4 words long, padded with no-ops (e.g. BNV 0) if necessary. The code is entered with the old pixel in R0, the one from the file in R8, and should exit with the result in R8. Bits 6..31 of the incoming pixels will be clear, and the result must have bits 6 and 7 clear. R1,R2 and R14 may be corrupted (allowing BL to be used for subroutine jumps to longer code), R13 is the supervisor stack pointer. R10 bits 23..31 hold the column number of the current pixel, bits 0..7 hold the current line.

SWI Vdig_UserLoadCode &802D1
Copies 4 words of user code into the *PicLoad user code area for use with the U option.
On entry
    R0   points to user's code
On exit
    R0..4 corrupted

User code will remain in the user code area until the digitiser module is reloaded from the ROM (<Ctrl><Break>, or *RMReinit), but if the code is position dependant (e.g. includes a subroutine jump) remember that it may get relocated by *RMTidy.

Examples
```
DIM CODE% 16 : P%=CODE%

[ SUB R8,R8,R8,LSR#2 : ADD R8,R8,R0,LSR#2 : BNV 0 : BNV 0 ]
```
This mixes the new and old pictures in the proportion 3:1
or
```
[ TST R10,#1 : MOVNE R8,R0 : BNV 0 : BNV 0 ]
```
Only loads alternate lines of the new picture
or
```
[ EOR R1,R10,R10,LSR#24 : TST R1,#&20 : MOVNE R8,R0 : BNV 0 ]
```
Loads alternate squares in chessboard pattern
```
SYS"Vdig_UserLoadCode",CODE%
*PicLoad <file> US
```

***PicLoad / *PicSave File Format**
A simple run-length data compression system is used to encode the 128K 6 bit pixel values from the digitiser's memory (256 lines of 512 pixels). The image is stored left to right, top to bottom. For each byte in the file, bits 0..5 form a number, n the meaning of which depends on bits 6 and 7 :

bit 7 6	
0 0	Store pixel value n at current address in digitiser memory
0 1	Subtract 1 from the last pixel stored, and repeat it n times
1 0	Repeat the last pixel n times
1 1	Add 1 to the last pixel and repeat it n times

The first byte in the file will always have bits 6 and 7 clear. The file length is such that when decoded, exactly &20000 bytes are recovered. Byte values &40,&80 and &C0 are never produced, and are reserved for possible future expansion, so user generated files must not contain these values.

10 Image processing

Two commands are provided to enhance the image in digitiser memory.

10.1 *Smooth

Smooths the image by averaging each pixel with its nearest neighbours. This process is useful for removing noise from a picture, but at the cost of a slight blurring of small details. Repeated application of *Smooth makes the picture look out of focus.

Syntax : `*Smooth [S[S][P][L]] [<n>]`

Options

<n>	- If a number is specified, *Smooth will be applied <n> times
Sxxx	- If this option is present, the picture will be displayed using *ShowPic after the picture has been smoothed. xxx are the options for *ShowPic (S,P,L).

If Sxxx and <n> are both given, the action depends on the order of the parameters:

```
    *Smooth Sxxx <n>
```

Will display after each pass.

```
    *Smooth <n> Sxxx
```

Will only display after all passes have been performed.

Examples

```
    *Smooth 2 S
```

Smooths twice, then displays.

```
    *Smooth SP 3
```

Smooths 3 times, displaying after each pass, without resetting the palette.

When doing multiple passes, *Smooth takes about 0.8 seconds per pass. <Escape> can be used to exit at the end of each pass.

10.2 *Zit

This command removes spurious spots (usually due to noise) in the image in digitiser memory.

The effect on the image is almost unnoticeable, but it will greatly improve the effectiveness of the run-length compression used by *PicSave, and can reduce the size of the saved file by up to 30K (depending on image content).

The actual operation performed by *Zit is as follows :
For each group of 3 pixels in a line, if the outer pixels are the same, but different from the centre one, the centre one is replaced by the value of the 2 outer pixels.

11 PRINTING

The digitiser module contains a very versatile screendump facility, which although intended principally for printing digitised pictures, will print screens from any graphics mode. It will work with any dot matrix printer which is compatible with the Epson FX or LQ series. It is also possible to use some non FX-compatible printers (e.g. Epson MX series).

The screen (or any section of it) may be printed to any specified size, the only limitation being the width of the printer. It is also possible to do very large printouts by printing sections which can be joined to form the complete picture.

A variety of printed dot densities and dot patterns may be selected to give the optimum resolution and/or contrast for any particular picture, and the software will ensure that the printout has the correct aspect ratio, regardless of size, although the picture can also be stretched or squashed if required.

The Screendump is accessed using either the *PicDump command or the SWI Vdig_ScreenDump call. The latter provides more flexibility, as it allows the user to change the control codes sent to the printer. The following section details the various options available, and is applicable to both *PicDump and SWI Vdig_ScreenDump, which are described individually later.

The Epson LQ series printers (LQ800,1000,1500,2500) have a different vertical dot density to most other printers, so to avoid printouts appearing elongated, an option is available to correct for this.

The screendump was designed for use in 16 colour modes, but it may be used in all graphics modes. The following notes describe the action taken in other screen modes.

2 colour modes
If pattern set 0 (default) is selected, colour 0 (black) isn't printed, and colour 1 (white) is printed as solid black. If other pattern sets are selected, patterns 0 and 1 are used.

4 colour modes
If pattern set 0 is selected, logical colours 0,1,2,3 (black, red, yellow, white) are printed as white,light grey,dark grey,black. If other pattern sets are specified, patterns 0 to 3 are used.

256 colour modes
Same as 16 colour modes, with the top 2 bits of the logical colour ignored, i.e. the same shade will be printed for colours 0,16,32 and 48. The tint is also ignored.

It is possible to stop the printing before it has finished, by holding <Ctrl> down until printing stops. <Escape> can also be used, but this is not recommended, as it will not always leave the printer in the correct state to receive more data later (i.e. the line spacing and margin won't be reset, and it may be left waiting for more graphics data). However, if the printer is not online (disconnected, paper jam etc.), only <Escape> can be used to exit.

11.1 Orientation

The screen can be printed in any of eight possible orientations, the orientation being specified by a number from 0 to 7. The following illustration shows the printout which would be obtained when printing the letter R in each orientation:

```
    0           1            2           3
  ****         ****        *   *       *   *
  *   *       *   *        *  *         *  *
  ****         ****        ****         ****
  *  *         *  *        *   *       *   *
  *   *       *   *        ****         ****

    4           5            6           7
* * * * *   *     *      * * * * *     *     *
    *   *     * *   *    *   *       *   * *
    *   *       *   *    *   *       *   *
  * *   *       *   *    *   * *     *   *
*     *     * * * * *      *     *   * * * * *
```

11.2 Density

Any 8 pin graphics mode available on the printer may be used, and the software will automatically rescale to ensure the correct size, whichever mode is selected. The different modes result in different printed densities, and generally speaking, the higher the density, the darker, and usually more detailed, the image. Higher density printing is much slower (and usually noisier!).

The following printer modes are supported :

Mode	Description	Horizontal dots per inch	
0	Single density	60	
1	Double density	120	
2	Fast double density	120	
3	Quad density	240	
4	CRT graphics	80	
5	Plotter graphics	72	(Not on LQ series)
6	CRT graphics	90	
7	DD Plotter graphics	144	(Only on EX series)

Not all of these modes are available on all types of printer, so to find out if all these modes are available, consult your printer manual, in the section describing the ESC * graphics command, which will usually include a table of available modes. If you try using a mode which isn't supported by the printer, a blank printout will usually result.

11.3 Printed size

Print sizes may be specified in four ways :

1) Height and width. The printed image will be scaled so that the contents of the graphics window exactly fills the required size, and therefore may appear stretched or squashed if the aspect ratio of the window is different to that of the requested size.
2) Height only. The software will work out what the width should be in order to preserve the correct aspect ratio of the printed image, taking into account the shape of the graphics window, and the print density.
3) Width only. As above, the height is calculated automatically.
4) Neither height or width. In this case a default width of 4 inches is used, the height being dependent on the aspect ratio of the graphics window and the orientation.

The width and height are given in units of 1/72 of an inch (or 1/60" if using an LQ series printer). If a value of zero is given, the value will be derived automatically as described above. The *PicDump command also allows the size to be specified in whole inches - this will be assumed if a value less than 20 is given.

If the printed width is too big for the printer, the excess will be cut off at the right-hand edge.

Due to the scaling used, you will find that when dumping screens containing lines (as opposed to solid objects or digitised images), part or all of some lines will not be printed. This occurs when the number of printed dots is not an exact multiple of the number of screen dots, causing some dots to be skipped. This effect can be avoided by specifying a printed size where the number of dots is an exact multiple of the number of screen dots the the mode being used. The number of dots printed horizontally is a function of the width and printer density. The number of vertical dots is 72 per inch (60 on LQ series printers). If you want a pixel-for-pixel dump, printer mode 5 is recommended (mode 0/1 on LQ printers), as it has a 1:1 aspect ratio, so the printed size may be specified exactly in dots (see examples later).

11.4 Dot patterns

Shades of grey or colours on the screen are represented using different patterns of printed dots, and the set of patterns used is selectable from 1 of 6 pre-defined sets, or may be user defined. The pattern set to be used is specified by a number in the range 0 to 7 :

0 - Default set as selected by *Configure PatternSet
1 - Normal shade patterns
2 - Diagonal lines of varying thickness
3 - Vertical lines " "
4 - Horizontal lines " "
5 - Diamonds of varying sizes
6 - Lower density shades (recommended for quad density mode)
7 - Use screen characters 240 to 255 as print patterns

The last option allows user experimentation with screendump patterns, as the definitions of screen characters 240 to 255 are used as the printer dot patterns for logical colours 0 to 15. These patterns may be defined using VDU 23, or using the supplied pattern editor, PatEdit (See section 14.2).

When pattern set 0 (default) is selected, the actual set used depends on the value configured in CMOS RAM, which may be set using *Configure PatternSet <n>, where <n> is the set number (1..7). CMOS Default is 1

11.5 User defined dot patterns

This section also applies to the dot patterns used by SWI Vdig_Scale in two colour modes.

If pattern set 7 is selected, the current definitions of screen characters 240 to 255 are used as print patterns, a dot in the character representing a printed dot.

A command is provided to define these characters from one of the six internal pattern sets, so that they may be edited and/or displayed.

***Patterns <n> [S]**
This command reads screendump dot pattern set <n> (0..7) into characters 240..255

If the S option is supplied, the pattern set will also be displayed on the screen. Note that the printed patterns are rotated 90 degrees from the way they are shown onscreen. A white dot on the screen represents a printed dot.

The S option will not display correctly in 20 column or non-graphics modes.

This command can also be used to create user screendump pattern files for Clares' Artisan, by using

```
*Patterns n { > $.artisan.printpat }
```

(note the spaces inside the { })
If n is 7 (i.e. user defined patterns), the currently defined characters are redefined to be the same as they were before - this is to allow the above command to be used to create a user pattern file from the current user defined patterns.

A user defined pattern editor, PatEdit, is supplied on the disk - see section 14.2 for details.

11.6 Print quality

The optimum pattern set and printer mode to use will depend on the picture content, the printed size, the printer, and the condition of the printer ribbon, and is best found by experimentation.

Generally, quad density printing produces the best definition of detail, but not always the best range of grey shades. Pattern sets 2,3 and 6 are recommended when using quad density.

Small printouts (less than 4") will generally have either poor resolution, poor contrast, or both. This is a limitation caused by the size of the dots printed by matrix printers. By far the best way to get good small pictures is to do a large (full page) dump using lower printer densities (60..120 dpi) in conjunction with pattern sets 1 to 5, and then reduce the picture to the required size on a reasonably good photocopier. Results comparable to laser print quality are obtainable using this method. Pattern set 4 is especially good for this, as it is an enlarged version of the type of pattern used to print photographs in newspapers and magazines.

It has been observed that printer ribbons from different manufacturers can vary considerably in quality, some producing dark grey or purple rather than black, or uneven darkness, leading to poorer quality screendumps. It can also be worthwhile experimenting with different types of paper.

If you are planning to do a lot of graphic printing, the Epson EX 800 and 1000 printers are especially recommended, as they are very fast, partially because unlike most matrix printers, they print graphics bidirectionally.

11.7 *PicDump

*PicDump dumps the current graphics window to the printer.

Syntax : `*PicDump [W<n>] [H<n>] [D<n>] [P<n>] [O<n>] [M<n>] [I] [Q] [F<filename>|S[S][P][L]]`

Options

W<n>	: Printed Width
H<n>	: Printed Height
D<n>	: Printer density (0..7)
P<n>	: Dot pattern set (0..7)
O<n>	: Orientation (0..7)
M<n>	: Left margin, in characters
I	: inverts pixel brightness
Q	: Epson LQ type printer
F<filename>	: *FastLoad <filename> and print it
S[<options>]	: Do *ShowPic first, <options> are *ShowPic options (S,P,L)

All parameters are optional, and may appear in any order. The first 6 options take a number, which must immediately follow the option letter (without a space), and there must be a space between each parameter.

Options W and H
These specify the width and height of the printed image. If the value given is less than 20, it will be interpreted as being in inches, otherwise it is in units of 1/72" (1/60" on LQ series printers). Omitting either or both will have the effects described in section 11.3

Options D,P,O
These specify the printer density, pattern set number, and orientation respectively, the numbers being as described in the previous section. If omitted, default values of 0 are used. i.e. single density, default pattern set (as defined by *Configure PatternSet), 'normal' orientation.

Option M
This allows the printout to be spaced out from the left-hand edge of the paper. The value given is the number of character spaces, the size of which will depend on the printer's current character printing mode (condensed, elite, enlarged etc.), and will be 1/10 inch for normal (pica) mode.

Option I
This inverts the screen pixel values, i.e. colour 15 (usually white) becomes 0 (black) etc.
In 2 and 4 colour modes, the colours are inverted by default, and this will be reversed to 'normal' by this option.

Option Q
This option should be given when using Epson LQ series printers in order to get the correct aspect ratio, as these printers print 60 dots per vertical inch instead of the usual 72.

Option F
This prints the given picture file by loading it onto the screen using *FastLoad before calling the screendump. See the description of *FastLoad (section 9.2) for more information.

Option S
This prints the picture currently in the digitiser's memory, by performing *ShowPic before printing.
If this option is given, and the current screen mode is not a 16 colour graphics mode, mode 12 is entered before displaying the picture. If there is insufficient screen memory for mode 12, mode 9 is used. A Can't in this mode error will result if there is insufficient memory to enter mode 9. Remember that a mode change will cancel any graphics window which may have been set up!
The S may optionally be followed (without spaces) by *ShowPic options (S,P,L). See the description of *ShowPic in section 6.4 for further details.

It's possible to send the screendump to a file rather than to the printer (e.g. for printing on another machine) using the Arthur redirection system :

```
*PicDump  <options> { > dumpfile }
```

(note the spaces inside the {})
This will send the dump to file dumpfile, which may then be printed using
`*Print dumpfile.`

Note thet if the F or S options are given when doing this, the screen must be in the correct mode before issuing the command, otherwise the VDU codes used to change mode will be also be redirected into the file, and the wrong screen mode will be used, resulting in an incorrect dump.

For large dumps in high printer densities, the dump file will be very big (e.g. 350K for a full page quad density dump!)

Examples

```
*Picdump
```

Prints the current screen, 4 inches wide, using the default dot patterns in single density mode.

```
*PicDump Fpic M10
```

Displays the picture file pic, and prints it 10 characters (usually 1 inch) from the left-hand side of the page.

```
*PicDump SS O3 H6 P1
```

Displays the section of the picture in digitiser memory selected by *Section, upside down, 6 inches high, using a diagonal line dot pattern in double density mode.

```
*PicDump D3 P3
```

Prints the current screen using a vertical line pattern in quad density. This combination gives a high level of detail with digitised pictures.

```
*PicDump S W12 O4
```

Prints large 12"x15" picture from digitiser memory sideways on a wide carriage printer (e.g. FX100 etc.) using the default patterns.

```
*PicDump O4 H640 D5
```

Exact pixel-for-pixel dump (assuming no graphics window) to avoid missing dots when dumping text, line drawings etc.

```
*PicDump W320 D5
```

As above, but smaller, for 320 pixel wide modes (1,4,9,13) only.

The following BASIC program prints the mode 12 image file piccie in 2 strips, which can be stuck together to form a 20x16 inch picture :

```
10 MODE 12          : REM set mode to preserve graphics window
20 VDU24,0;0;1279;511;        :REM Window around bottom half
30 *PicDump W8 O4 P2 Fpiccie
40 VDU2,1,12,3      :REM Formfeed to top of next sheet
50 VDU24,0;512;1279;1023;     :REM Window round top half
60 *PicDump W8 O4 P2
```

Note that patterns 2 and 5 usually give the best results for large dumps.

11.8 SWI Vdig_ScreenDump (&802D0)

This SWI call is an alternative to the *PicDump command.

On entry

R0	bits 0..2	- Orientation
	bit 3	- Invert shades (0..15 = white..black)
	bits 4..6	- Shade pattern set number
	bit 7	- Reserved
	bit 8	- Should be set when using LQ series printers

R1	- Printed width (in 1/72", or 1/60" on LQ printers)
R2	- Printed height (in 1/72", or 1/60" on LQ printers)
R3	- If in the range 0..7, R3 is the printer density If R3 is greater than 7, it is assumed to be a pointer to a block of printer information (described later)
R4	- Left margin in characters, ignored if R3 > 7

Examples

```
    SYS "Vdig_ScreenDump",&40
```
Prints default size using pattern set 4
```
    SYS "Vdig_ScreenDump",&104,,,3
```
Prints sideways on an LQ printer,quad density
```
    SYS "Vdig_ScreenDump",9,576
```
Prints mirror image, shades inverted, 8" wide

Printer information block

If R3 is greater than 7, it is assumed to be a pointer to the following printer information in memory :

R3	- Codes to send at beginning of dump (16 bytes)
R3+16	- Codes to send at start of each line of dump (16 bytes)
R3+32	- Codes to send at end of dump (16 bytes)
R3+48	- Aspect ratio : horizontal dpi * &10000 DIV vertical dpi (1 word)

The codes at R6+16 are sent at the start of each line, followed by the low and high bytes of the number of horizontal dots to be printed,and then the graphics data. To illustrate the sort of data expected, below is a listing of the internal code table used when R3 is less then 8.

```
00 00 00 00  00 00 00 00  00 00 1B 6C  lm 1B 33 18
00 00 00 00  00 00 00 00  00 00 00 00  0A 1B 2A pm
00 00 00 00  00 00 00 00  00 00 0A 1B  6C 00 1B 32
dd dd dd dd
```

`lm` is the left margin, `pm` is the printer graphics mode, `dd` depends on the density, e.g `55 D5 00 00` for mode 0 (60 dpi).

The first line sets the left margin and line spacing, the second does a linefeed (initially, and after the previous graphics line) and ESC * to enter graphics mode, and the last provides the final linefeed, as well as resetting the linespacing and left margin.

All 16 bytes in each line are sent, so unused bytes should be set to 0 (NUL), and the graphics mode command must be the last thing on the middle line. The data must include the linefeed for each line of data (0A) (a linefeed is preferable to a carriage return, as it isn't affected by the state of the printer's auto-LF DIP switch). The density value is used to calculate the number of horizontal dots required to fill the required width, and to ensure the correct aspect ratio if the width or height isn't specified.

The left margin given in R4 will be ignored if a printer information block is supplied.

The program MXDump on the disk uses the printer information block to allow screendumps on Epson MX series printers.

12 Miscellaneous SWIs

This section documents digitiser SWI calls not mentioned in the previous sections.

12.1 SWI Vdig_VideoInfo &802C2

This SWI call returns information about the incoming video signal.

On exit

R0 returns one of 3 values :

-1	(&FFFFFFFF) If no signal is present
0	If a non-interlaced video signal is present
1	If an interlaced video signal is present

12.2 Vdig_VideoParams &802C5

This sets or reads 2 parameters used when grabbing:

On entry

R0	- Vertical delay in lines before starting to grab. The initial default is 34.
R1	- Timeout before giving 'no video signal' error. Initial default is &8000. The timeout period is approximately (R1)/50000 sec.

If R0 or R1 are 0, the respective parameter isn't changed.

On exit

R0	- New or current vertical delay
R1	- New or current timeout

New values will remain in force until the RM is reloaded (hard reset).

The vertical delay defines the start position of the 256 grabbed lines in the 312 line TV field. The default value of 34 gives a picture centred on the lines containing picture information on a broadcast signal.

Adjustment of the timeout is provided to stop grab routines returning a 'No video signal' error when there is no signal. This is mainly of use when switching between video sources, or tuning in an off-air source.

Example

```
        SYS "Vdig_VideoParams",,-1
```

Will cause grabs to wait almost indefinitely for signal rather than causing an error.

Important note

If the timeout has been extended significantly, it should be remembered that key triggered grabs are also affected, so if no signal is present when the trigger key combination is pressed, the system will hang up until either a signal is received, or <Escape> is pressed.

12.3 SWI Vdig_ScreenSwap &802C8

This SWI provides an easy to use method of screen bank swapping, a technique used to avoid flicker when displaying animated graphics. 2 screen banks are used, one being displayed whilst the next image is being written to the other. When the image is complete, the banks are swapped, giving an instant update of the screen. Swapping is usually done during the vertical flyback time (of the display, not the digitiser), so that the swap occurs when nothing is being sent to the monitor.
The screen bank written to by the Arthur VDU drivers, and the digitiser software, is referred to as the VDU driver bank.
See also the description of OS_Byte calls 112 and 113 in the Programmer's Reference guide.

On entry

R0	bit 0 -	0 If VDU driver bank is the same as the displayed screen bank, the VDU driver bank is set to be the non displayed bank, otherwise the VDU driver and displayed screen banks are swapped over. 1 The VDU driver bank is set to the displayed screen bank.
	bit 1 -	0 Perform swap immediately. 1 Wait until vertical flyback before swapping.

Screen swapping has no effect if the configured value of ScreenSize isn't big enough to hold 2 banks in the current screen mode.

Example
The following BASIC program is equivalent to the *See command.

```
  10 MODE 9 : *Bwpal
  20 REPEAT : CLS
  30 *GRAB
  40 SYS"Vdig_ScreenSwap"
  50 UNTIL INKEY$0 <> ""
  60 SYS"Vdig_ScreenSwap",1
```

The last line restores the VDU to normal (unswapped) operation, and if it were omitted, any characters printed would not appear on the displayed screen. This should be remembered if SWI Vdig_ScreenSwap is used in programs where <Escape> or some other error could occur - it's a good idea to put

```
    SYS "Vdig_ScreenSwap",1
```

in your error handler.

This SWI is also used in the example programs Rotate and Bounce2 on the disk.

12.4 SWI Vdig_Display &802C9

Copies a picture from main memory to the screen quickly, adding padding where necessary if the image width is less than the screen width. The main use of this call is for displaying picture data previously stored in memory (e.g. for animation).

On entry
R0 - Picture position on screen, or address to store data.
R1 - Address of image data to be displayed.
R2 - Length of picture data in bytes (defaults to the whole screen if 0).
R3 - Displayed bytes per line and / or padding value.
R4 - Delay factor
On exit
R1 - (R1 on entry) + data length

If R0 bit 31 is clear, R0 is used as a byte offset from the top left-hand corner of the screen. When bit 31 is set, R0 is used as an absolute address (with bit 31 masked out) to store the image. The address or offset must be word aligned.

Bytes 0,1 of R3 hold the length of the displayed line in bytes, or 0 to default to the full screen width.

If R3 bit 31 is set, bytes 3,2 (with bit 31 masked out) hold the amount of padding to add at the end of each line. If bit 31 is clear, the default padding of (screen width - line length) is used.
The value for bytes per line and picture data length must be a multiple of 16 bytes.
The padding value must be a multiple of 4 bytes.

If R4<>0, there will be a delay of (R4) field periods before displaying the picture. This is of use when playing back animations without causing display flicker. Setting R4 to 1 will wait for the next vertical flyback before writing to the screen.

For an example of the use of this call, see the program Animate on the disk.

The next four SWIs allow raw data to be dumped to or from the digitiser's memory, bypassing all scaling.

12.5 Vdig_ReadLine &802CB

Reads one or more horizontal lines of 512 pixels from digitiser memory.

On entry

R0	- Address to read data to (must be word aligned)
R1	- First line to read (0..255)
R2	- Number of lines to read (1..255, reads 1 if R2=0)

Each line read consists of 512 bytes, each containing a pixel value from 0..63, from left to right. Line numbers are from top (0) to bottom (255).
If R1+R2 > 256, reading will stop after line 255

12.6 SWI Vdig_WriteLine &802CC

Writes a line or lines of pixels to digitiser memory, Parameters are the same as Vdig_ReadLine.

Example
The following program turns the image in digitiser memory upside down.

```
10 DIM buffer% &20000
20 SYS "Vdig_ReadLine",buffer%,,256
30 FOR a%=0 TO 255
40 SYS "Vdig_WriteLine",buffer%+a%*512,255-a%,1
50 NEXT a%
60 *ShowPic
```

(Note that this isn't the most memory efficient way of doing this!)

12.7 SWI Vdig_ReadColumn &802CD

Reads one or more vertical columns of 256 pixels from digitiser memory.

On entry

R0 -	Address (may be byte or word aligned)
R1 -	Column number (0..511)
R2 -	Number of columns (1..512, reads 1 if R2=0)

Column data is stored as 256 bytes, from top to bottom

12.8 SWI Vdig_WriteColumn &802CE

Writes one or more columns, parameters are the same as Vdig_ReadColumn

Note that reading data by lines is considerably faster than reading by columns.

12.9 SWI Vdig_HardwareAddress &802C0

Returns the base address of the podule hardware.
On exit

R0 - Hardware base address.

12.10 SWI Vdig_Refresh &802C1

When the digitiser module is loaded, it sets up a background process which refreshes the digitiser's on-board memory. Pictures can only be kept in the digitiser as long as this refresh is active. This SWI call disables or re-enables the refresh process. If refresh is disabled for more than about 0.1 sec., pictures stored in the digitiser's memory may become corrupted.

On entry

R0 -	0	Disable refresh, releasing event vector
	1	Enable refresh
	2	Un-suspend refresh
	3	Suspend refresh temporarily

The first 2 options are intended for long-term disabling, as they claim and release the ticker event vector. The suspend / un-suspend options are much faster, and are intended for temporarily stopping refresh while accessing the digitiser hardware.

Note : SWI Vdig_HardwareAddress and Vdig_Refresh are intended for use by software which accesses the digitiser hardware directly, and will not normally be needed by the user.

13 Colour image grabbing

A program, ColourGrab, is supplied on the software disk, which allows colour images to be grabbed using a video camera (either monochrome or colour) and 3 coloured filters.

The image is grabbed three times, through red, green and blue filters, and these three primary colour images are combined to produce the full colour picture. The subject being grabbed must, of course, remain stationary while the 3 images are digitised.

Due to the automatic level control in the digitiser, and in most video cameras, the primary colour images will not be in the correct proportions, giving an incorrectly coloured image. The ColourGrab program gets round this by providing the ability to manually adjust the proportions of each colour, until the desired colour balance is obtained. The image may then be saved or turned into a sprite.

ColourGrab is in the Utils directory on the disk, and can be run using *Utils.ColourGrab. A camera icon will be displayed on the screen, the centre part of which indicates the colour filter to use, which will initially be red. Pressing <left> will grab and display the red part of the image, and change the icon to green, ready for the next filter. After grabbing using all 3 filters, the camera icon will return to red, and if required, you can repeat the grabbing process. If you want to re-grab one particular colour, use <right> to cycle through the 3 colours, and grab using <left> when the desired colour appears in the camera icon.

When a satisfactory image is obtained, press <centre>, which will take you to the colour balancing screen. A quarter-screen image will be displayed, and the mouse pointer shape will change to a square box containing a colour. This colour approximately indicates the tint to be applied to the image, and is calculated from the mouse position. The centrescreen position represents no tint, right of centre increases red, left decreases red, above centre increases blue, and below centre decreases blue. The colour in the box will change as the mouse is moved. To see the result of applying the tint to the image, press <left>. The new image will be displayed after a slight delay. You can try any number of tint values until the image looks right.

If you only want to display a section of the image, press <centre>. After a short delay, a fullscreen image is displayed, and the mouse pointer appears as a cross. A flashing rectangular box is displayed over the image, which indicates the part of the picture to be used. The mouse moves the corner of this rectangle marked by the cross, which may be flipped to the opposite corner by pressing <left>. Pressing <right> expands the box to cover the whole screen. When the desired section has been marked, press <centre> to return to the colour balance screen, which will then display only the selected part of the image.

It is also possible to alter the size of the displayed image - press <right>, and a flashing box will appear on the screen, representing the current size, which may be changed using the mouse. The current image size in graphics units horizontally and vertically will be displayed at the top of the screen. Press any mouse button to return to the colour balance screen, and display the image at its new size. If the image is to be made into a sprite, the size of the sprite will be the size set using this option

Large image sizes will take longer to display (approx. 1.5 sec. for fullscreen), which is why a quarterscreen picture is initially used, to speed up the balancing process. When the program is generating a new image, a watch icon is displayed during processing.

When the desired colour balance and size have been selected, press <Return>. The following menu will be displayed :

M	- Make Sprite
S	- Save fullscreen
B	- Return to colour balance screen
G	- Grab new image
Q	- Quit
*	- * Command

The first two options will prompt for a screen mode - type 5 for mode 15 (640x256), or any other key for mode 13 (320x256).

M Will ask for a sprite name, and create a sprite of that name from the image, which will be the size selected when colour balancing. The sprite will be selected as the 'current' one, for plotting with PLOT&ED. If you hold down a mouse button when <Return> is pressed after the sprite name, the sprite will be repeatedly plotted onscreen at the mouse position until all mouse buttons are released.

F Prompts for a filename, and then saves a full screen copy of the image (using *FastSave), together with the correct palette.

To create a file of a non-fullscreen image, Type *SNew, use the M option to create a sprite, then *SSave <filename>. This file can then be loaded to the screen using *ScreenLoad, or as a sprite with *SLoad.

B Returns to the balance screen, to allow the colour balance
or size to be readjusted.

G Restarts the program to grab a new picture.

Q Exits from the program.

Notes on colour grabbing
Colour images created by this method don't use the standard 256 colour mode palette, which is set up to display 2 bits of red, green and blue, plus 2 bits tint. Instead, ColourGrab creates a palette to obtain 3 bits of red and green, with 2 bits blue. This means that the normal convention of colour plus tint will not give the expected results when this new palette is in use.

Files and sprites created by ColourGrab contain the correct palette data for display. Additionally, There is a palette file ColPal on the disk, in the Utils directory (*ColPal to set the new palette).
The colour filters used may be photographic filters, coloured plastic, cellophane, or anything which is transparent, and reasonably accurate in colour (filter errors can be partially corrected for when colour balancing). Alternatively, coloured lighting could be used, as long as the colour light source provides the only illumination. If you have a set of cyan, yellow and magenta filters (100% density), as used for colour photographic printing, these can be used in pairs :

Cyan+Yellow=Green, Cyan+Magenta=Blue, Yellow+Magenta=Red.

14 Utility programs

These programs are all in the Utils directory on the digitiser software disk, and can be run using *Utils.<programname>, or alternatively they may be run from the desktop.

14.1 MCShades

This program is used to determine the correction factors which need to be used in order to obtain an even 64 shade grey scale on monochrome monitors in 256 colour screen modes. The correction factors are stored in battery-backed CMOS RAM, and so once set, will not need resetting unless the digitiser podule is relocated to a new podule slot, or CMOS RAM is reset or corrupted somehow.

When it is run, the program will display some instructions, and then display a grey scale on the screen, from black at the left side to white on the right. Moving the mouse will adjust the correction factors, and update the scale using the new values.
You should move the mouse to the position which gives the most even change from black, through grey, to white, minimising sudden vertical 'steps', or changes in brightness. By moving the mouse around the screen area, you will fairly quickly find the area which gives a reasonable scale, and the mouse should then be moved slowly around that area to obtain the best possible results.

It is quite probable that you won't be able to get rid of all the vertical brightness 'steps', but this shouldn't affect image quality too badly.

Pressing any mouse button during this process will grab and display an image using the new correction factors.

Once the optimum setting has been found, pressing <Return> will store the new values in CMOS RAM, and re-initialise the digitiser software to use the new values immediately (assuming 64 shade mode is selected - see section 4.1).

Needless to say, this program must be used on a monochrome monitor!.

14.2 PatEdit

This program allows the dot patterns used by the screendump and 2 colour mode displays to be edited. When PatEdit is run, blocks of user-defined characters 240..255 are displayed along the top of the screen, with an enlarged version of the current character being edited in the bottom left-hand corner.

<left> and <centre> are used to set and clear dots respectively, and <right> moves to the previous character if the pointer is in the left-hand side of the editing area, or to the next if in the right-hand side, the current character being indicated by the marker at the top of the screen.

Various editing facilities are available via the keyboard, the key functions being listed on the screen when the program is run.

Character editing

X	Flips the character horizontally
Y	Flips vertically
R	Rotates 90 degrees clockwise
C	Copies the last character selected to the current one

Set editing

<	Inserts a gap in the set at the current pattern, shifting the denser patterns to the left, to allow a new one to be inserted.
>	As above, but shifts towards the less dense end.
<delete>	deletes the current pattern, closing the gap by shifting up the denser patterns.
<copy>	as above, but shifts up from the less dense end,

Other facilities

L	Loads a pattern file
S	Saves the current set
D	Downloads a pattern set from the digitiser module
P	Prints a test strip using the current pattern set

0,1,2,3,4,5,6,7 keys select the printer density (remember that not all densities are available on some printers.

Q	Displays the current image from digitiser memory in 2 colour mode, using the current patterns. When the image has been displayed, press <Shift> to grab and display a new image, or any other key, or a mouse button to return to the editor.

Pattern set files saved by this program may be re-loaded via the program, or by using *Print <filename>. The files are also compatible with the Clares' Artisan user screendump pattern file, $.ARTISAN.PRINTPAT

PatEdit file format
The file consists of 16 VDU 23 sequences :

```
23 240 nn nn nn nn nn nn nn nn
23 241 nn nn nn nn nn nn nn nn
..
..
23 255 nn nn nn nn nn nn nn nn
```

`nn` is data for each line, as in VDU 23

Total length 160 bytes

14.3 MXDump

This utility allows screendumps to be done on Epson MX series printers. It is also provided as an example of using the printer information block with SWI Vdig_ScreenDump.

15 Example programs on disc

The following example programs are in the Examples directory on the disk, and each is briefly described here,. There is more information in comments in the programs. All are in BASIC.

Animate
Records and plays back sequences of images in memory. Uses SWI Vdig_FastGrab, Vdig_Display.

Interlace
Grabs and displays 640*512 images in interlaced format on normal (i.e. non MultiSync) monitors. Uses SWI Vdig_FastGrab, Vdig_Display, *Grab.

Patches
Grabs and displays images in random sequences of squares on the screen. Uses SWI Vdig_Scale.

PicBounce, PicBounce2
These two programs bounce moving images around the screen in different ways. Uses SWI Vdig_FastGrab, ScreenSwap.

PicFlood
Includes a procedure to flood fill a picture into an enclosed shape. Uses SWI Vdig_Scale, Vdig_Grab.

Rotate
Demontration of various rotation effects. Uses SWI Vdig_Rotate, Vdig_ScreenSwap.

Telly
Illustrates grabbing as a background task, grabbing quarterscreen images to the screen while other programs are running (BASIC/machine code). Uses SWI Vdig_FastGrab.

Threshold
Illustrates the use of pixel lookup tables to divide an image into just black and white. Uses SWI Vdig_Scale, Vdig_Tables.

VidParams
Alters vertical position of grabbed lines in TV field. Uses SWI Vdig_FastGrab, VideoParams.

16 Technical information

16.1 General

When a SWI or * command writes to the screen, the screen bank used is that used by the VDU drivers, (set by *FX112). This allows bank-switching techniques to be used, as with the normal OS drawing routines.

Whilst a field is being grabbed, FIQ and IRQ interrupts are disabled during the period between the request being issued, and the end of the field being grabbed. This will cause the system clock (NOT the real-time clock) to lose between 2 and 6 'ticks' (20-60msec) every time an image is grabbed. Fast repetitive grabbing will therefore make the clock lose significantly, as well as making keyboard and mouse response more sluggish. This is only likely to be a problem when displaying small pictures at high field rates, as the percentage of time spent grabbing (rather than writing to the screen) is higher.

There is a minor effect on the real-time clock, which is unlikely to cause problems: as the seconds count is maintained by software, the date, hours and minutes will always be correct, but the seconds count may get out of step, so the minute will not 'roll over' at 59 seconds.

16.2 Calling digitiser SWIs from background tasks

Most Digitiser SWIs (but NOT Vdig_ScreenDump!) may be called by interrupt or event routines, and although some of them enable interrupts whilst executing, they are not re-entrant, so to avoid problems when a SWI is called while another is being executed, an internal flag is kept to indicate when a digitiser routine is in progress. If a digitiser SWI is called by an event/interrupt routine when another is already in progress, the call will return with the error Busy (error number &800304). Obviously, only calls generated by interrupts or events can fail in this way.

A few points should be noted when calling digitiser routines from event code: When grabbing, FIQ interrupts are disabled for the duration of the field being grabbed. This can cause problems with I/O devices which use FIQs, normally the floppy disk and Econet. If a grab event occurs whist transferring data to or from floppy, the transfer will just pause for a moment, without any ill-effects. If a grab is performed when transferring data to or from Econet, The Econet operation will usually fail, giving a 'no reply' or similar error after a few seconds' timeout period.

These above effects have been observed using the version of Arthur current at the time of writing (1.20), and future versions may behave differently!

It should be remembered that some digitiser routines take a significant amount of time to execute (e.g. scale/rotate in high-res modes), so if they are called in rapid succession (e.g. by a timer event), there will be very little CPU time left available to foreground tasks.
A more obvious problem is that applications don't generally expect things to appear on the screen unexpectedly (i.e. put there by a background task), so great care should be used to avoid displaying pictures where this could cause problems (e.g. when there are pop-up windows on the screen)

An example program Telly on the disk uses the vertical sync. event to display images on the screen continuously as a background task.

16.3 Hardware information

The digitiser accepts a standard 1V composite video signal, and will adjust its input range over the approximate range 0.5 to 1.5 volts. This level adjustment, in conjunction with the internal black-level clamping ensures that the full input range of the analogue to digital converter is used. The link LK1 on the podule connects a terminating resistance, and may be removed if termination is not required. Note that you may experience problems when 'looping through' a signal via the digitiser, as the black level clamping circuit will tend distort or attenuate the colour burst, as the input is not buffered. For the same reason, clamping performance will be impaired if the source has a relatively high impedance.

An input filter is included to remove any colour subcarrier signals which may be present, and whilst this filter is quite effective, there will still be a small amount of patterning due to aliasing on strongly coloured scenes, and will be most noticeable in 320 pixel wide screen modes. It is therefore preferable to use a monochrome signal if possible.

The digitiser's on-board memory is dynamic, and is refreshed by a background process set up by the digitiser module, so if the module is killed, or interrupts diabled for long periods of time, any stored image will be lost or corrupted.

17 Early Arthur ROM Versions

If your system still has an old version of the Arthur operating system ROMs (version number less than 1.1), you have to load the digitiser module from the ROM using a special program supplied on the disk, in the Utils directory. It can be run with *Modload, or by clicking on its icon from the desktop.

The message Digitiser Module Loaded will be displayed if the module was loaded successfully. A hard reset (<CTRL><Break>), or *RMClear will delete the module from memory, so it will need reloading.

If the message Podule not present is displayed, check that the digitiser is correctly inserted, and that the power leads are correctly connected to the podule backplane.

If the error No room in RMA, or Can't set up workspace occurs, the size of the relocatable module area (RMA) needs to be enlarged to hold the digitiser software. This can be done by exiting from BASIC before using *ModLoad. Alternatively, you can change the default amount of memory reserved for the RMA on power-up, which is held in CMOS RAM. The command *Status RMASize will show the current configured size, and can be changed using *Configure RMASize n On an A300 series machine, the value of n should be increased by 4, or by 1 on the A400 series. Depending on the version of the Arthur ROMs in your system, you may need to increase the value by more than this - increase the size until *ModLoad works successfully. Remember that the value of RMAsize is only read on a hard reset, so press <Ctrl><Break> after changing the value, before trying *ModLoad again.

18 Using Art packages

These notes are in addition to those in section 5. Remember that you will have to use *Configure GrabOptions to set the required options before running the drawing program.

18.1 Clares' Artisan

Images can be grabbed directly to the Artisan screen, using the key triggered grab facility. The P GrabOption may be used to set the correct palette if required. For fullscreen images use the F GrabOption, or the M option for partial screen grabs.

Care is needed when using <right> to cancel an operation, as the image may also disappear!. This can be avoided by re-selecting one of the menus after grabbing. It is possible to make Artisan use the *FastLoad and *FastSave commands when loading and saving images. To do this, insert your Artisan working disk (not the original) in the drive and type the following (exit from the desktop if necessary) :

```
*ACCESS ART4 WR
*RENAME ART4 OLDART4
*BASIC
10 *SET Alias$ScreenLoad FastLoad %0
20 *SET Alias$ScreenSave FastSave %0
30 CHAIN "OLDART4"
SAVE "ART4"
```

This modifies the files on the disk, so that every time you run that copy of Artisan, loading and saving will be about 5 times faster.
If you want to use user lookup tables for the key triggered grabs within Artisan, add the appropriate *TableLoad command to the above program, as Artisan resets the machine, causing any previous lookup tables to be lost.

For some reason, you can't use <Ctrl>-<Tab> as the grab trigger key within Artisan.

18.2 Fairhurst Instruments' Arctist

Images can be grabbed directly to the Arctist screen using the key triggered grab facility, but you should only do this when the menu is not on the screen.
Remember to use *Configure Shades if necessary to set the desired 256 colour image display mode.

As with Artisan, screen loading and saving can be greatly speeded up by using:

```
      *SET Alias$ScreenLoad FastLoad %0
      *SET Alias$ScreenSave FastSave %0
```

before running Arctist.

18.3 ARM Paint (on the welcome disk)

Because this package only uses part of the screen, you have to use GrabOption M, which plots the image between the last 2 mouse click points.
A convenient way of using this facility is to use the rectangle option, which requires a mouse click at each corner, so if you draw a rectangle and then press the grab trigger key, the image will fill the rectangle.
It is also useful to use the P option as well, as setting the palette in ARM Paint is a little tricky!
The *FastLoad / *FastSave commands should not be used with ARM Paint, as it does not use fullscreen images.

19 Error Messages

Error numbers start at an Acorn-allocated base of &800300

`Can't in this mode` (&800300)
This occurs when an attempt is made to use an invalid screen mode (2 or 4 colour modes, or mode 7) for picture display.

`Podule not present` (&800301)
This error can usually only occur when using *ModLoad.

`No video signal` (&800303)
This occurs when a field grab is performed when there is no video signal being received by the digitiser. It may also occur when using certain video recorders or videodisc players in fast, slow-motion or pause modes. This error can be effectively disabled using SWI Vdig_VideoParams to increase the timeout period.

`Busy` (&800304)
This occurs if a digitiser SWI routine is called whist another is still executing. This can only occur when the call is made by a background task (e.g. an event).

`Invalid address` (&800305)
Occurs when an attempt is made to transfer data to or from an invalid memory address (e.g. using SWI Vdig_Readine etc.).

`Bad Parameters` (&800306)
Invalid option letters were used in a * command or *Configure option.

`Can't switch pointer on` (&800307)
Occurs if *Section or *MakeSprite can't enable the mouse pointer (usually because the WindowManager module is not active).

`Aborted` (&800308)
Occurs if *GrabSave is exited by a key other than <Return>.

`Can't set up workspace` (&800309)
Occurs if there is insufficient space in the RMA to reserve workspace for the module. This can only usually happen if *RMReinit is used to re-activate the module after it had been killed with *RMKill or *UnPlug. Quit from BASIC, or re-configure RMASize to reserve more space.

`Not a full screen ScreenSave file` (&80030A)
An attempt was made to *FastLoad a file which was not saved using *FastSave, or isn't a full screen (e.g. *ScreenSave used when a graphics window was active).

`Not a lookup table file` (&80030B)
An attempt was made to *TableLoad a file whose length was not 192 bytes.

20 SWI Calls and numbers

Number	**Name**
&802C0	HardwareAddress
&802C1	Refresh
&802C2	VideoInfo
&802C3	Grab
&802C4	FastGrab
&802C5	VideoParams
&802C6	Scale
&802C7	Tables
&802C8	ScreenSwap
&802C9	Display
&802CA	Rotate
&802CB	ReadLine
&802CC	WriteLine
&802CD	ReadColumn
&802CE	WriteColumn
&802CF	Section
&802D0	ScreenDump
&802D1	UserLoadCode

21 GLOSSARY

Aspect ratio
The ratio of a picture's width to its height. For the Archimedes screen and TV images, the ratio is usually 5:4

Interlacing
The technique used to display broadcast TV pictures, which are transmitted as 2 fields, 50 times per second, giving a frame rate of 25 per second. Each field consists of 312.5 lines, and as the fields are vertically spaced half a line apart, they combine to make a 625 line frame. The 2 fields are called the odd and even fields, the odd consisting of lines 1,3,5 etc. of the frame, the even contaning lines 2,4,6 etc.

CMOS RAM
The memory in the Archimedes which is maintained by a battery when power is off. It is used to store configuration data and options.

Relocatable Module Area (RMA)
The area of memory allocated by the Arthur operating system for storing software modules (RMs) and their workspace. The digitiser software is one such module.

Grey scale
A display consisting of all the displayable screen brightnesses from black, through grey, to white.

SWI (Software interrupt)
This is an machine-code instruction used to access operating system routines.

ROM (Read Only Memory)
A device which is used to permanently store programs and data.

Noise
A random disturbance to an electrical signal. In the context of digitised video, noise manifests itself as spots and 'snow' on the image.

Clipping
This is technique used when drawing objects on the screen - any part of the image which should lie outside the visible part of the screen or the graphics window is not displayed (clipped).

INDEX

Notes

Notes

Notes

Jessa House, 250 High Street, Watford, WD1 2AN, England
Tel: Watford (0923) 37774, Telex: 8956095 WATFRD, FAX: 01 950 8989