User 312

Picasso II

The Retargetable Graphics Board for your Amiga

Version 1.0

August 31 1993

Manual

USA Distribution by:

Expert Services
7559 Mall Road
Florence, KY 41042

Phone 606-371-9690
Fax 606-282-5942
BBS 606-282-5943

Trademarks

Trademarks are used without any mark.

Workbench™, Intuition™ und Amiga™ are registered Trademarks of Commodore Amiga Inc., West Chester, USA

Macintosh is a lizensiertes trademark of Apple Computer Inc.

Windows™ is a registered trademark of Microsoft Corporation, USA

DeluxePaint™ is a registered trademark of Electronic Arts

Written by: David Göhler

Layout by: David Göhler and Michael 'Mick' Hohmann with PasTEX on an Amiga 4000/040 & Picasso II

Printing: DoItYourSelf-Press,Hannover

Copyright Notice

Legal Aspects

Table of Contents

1. Introduction

Since the beginning of the computer revolution, graphic animations and simulations have been fascinating users. Attempts have even been made to create artificial worlds, and now as we approach the end of this century we are close to our goal. Never before have we been able to create graphics and animations so close to reality. Computers, of course, play an important role in these creations.

Special hardware is required for these creations. The Amiga is an affordable machine that is well suited for video productions. The only thing that has been missing from the Amiga is higher resolutions and more colors. Speed has not been an issue for the Amiga unless maximum overscan and maximum colors were being used. Considering this, we wish to congratulate you, you have just purchased a graphics card that will eliminate your graphic restrictions without losing speed. You are now the proud owner of a graphics card for the Amiga, which is an excellent value considering its performance, quality, and speed. We would like to thank you for trusting our product. Quality is our primary concern, and we are sure that you will not regret purchasing your Picasso II graphics board.

Should you have any complaints or comments, please write or call us, we want you to be completely satisfied with your purchase.

1.1. About This Manual

This manual will describe how to install your Picasso II graphics board and how to install and use the included software. In the back of this manual you will find technical data and information that will be useful to programmers. We have also provided a Fast Start chapter for those of you already familiar with the installation of hardware and software in your Amiga. If you are not familiar with installations, please read carefully the chapters following the Fast Start chapter.

1.2. System Requirements

The Picasso II graphics board requires Workbench 2.0 or higher for proper operation.

Since a blitter graphics chip is part of the Picasso II design, all the functions of the Amiga blitter are taken over by the Picasso II. This means that you

will be able to install the Picasso II into an Amiga with a 68000 cpu with out degrading system performance.

1.3. Monitors

t is very important to choose the right monitor. You will NOT be able to use a Commodore 1084 or 1084S, (the display will be blank), a multiscan monitor must be used, (a 14 inch or greater is highly recommended). Higher esolutions (starting at 1024x768), will require a monitor capable of displaying frequencies up to 57KHz, (Example: NEC 4FG). If you do not currently own a monitor, we would be happy to advise you as to which monitor would best suit your needs.

1.4. Credits

We would like to thank all of those people you have made this product possible and who have sacrificed a lot of their time and effort to make the Picasso II graphics board a successful product.

These people are (listed in alphabetical order):

Software

Michael Balzer
David Göhler
Georg Heßmann
Uwe Röhm
Thomas Sontowski
Stefan Sticht

Hardware

Andreas Bober
Klaus Burkert
Hubert Neumeier
Holger Wollny

English Manual Translation

Martha Bennett
Scott Bennett
Roland Buck
Brick Eksten
George Gibeau

As well as Angela Schmidt, Henning Schmiedehausen, Birgit Neumeier, Frank Diedrich, Arthur Andergg, Michael Hohmann. And everyone else who has helped develop this product.

2. What is Included

Please make sure that the following items are included with your package:

- 3 Disks
- The Picasso II Graphics Board
- A Short Monitor Cable
- This Manual

If any of these items are missing or damaged, please return this package to your dealer, or contact us directly.

The included short monitor cable is a high density 15 pin male to male cable. If you do not have an Amiga 3000 or an Amiga 2000 with a deinterlacer card, you will need another adapter. The adapter for the Amiga 23 pin RGB port is the same as the adapter provided with the Amiga 4000. The flickerFixer will either require a flickerFixer (from MicroWay) to VGA cable with a male high density 15 pin connector, or a 9 pin to high density pin 15 pin VGA adapter. If you do not own one of these adapters, contact your dealer or call us directly.

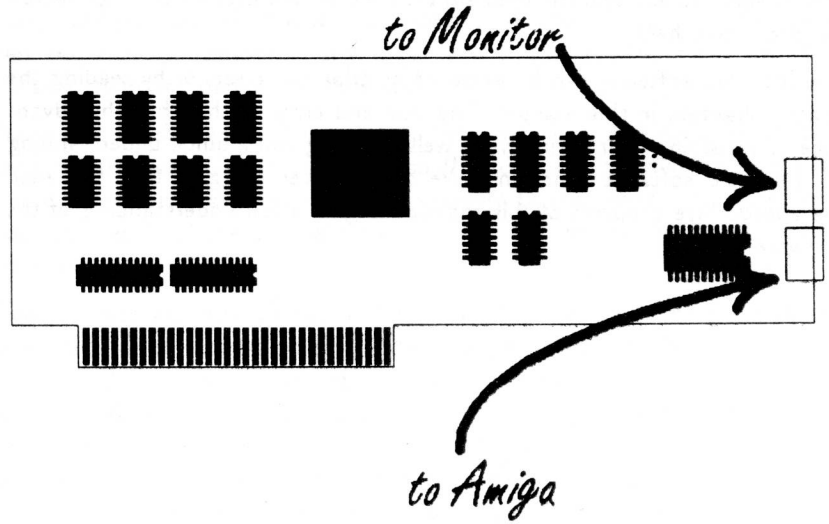
The Picasso II packaging is both recyclable and made of recycled material.

3. Fast Start

If you already have experience installing hardware and software, this chapter will provide all information necessary to install and use the Picasso II. If you do not have experience with these types of installations, please skip to the next chapter.

3.1. Hardware Installation

Open your Amiga and locate an empty Zorro II/III slot. The Picasso II utilizes a normal Zorro slot, NOT the video slot. Make sure to fully insert the Picasso II into the chosen slot. Connect the short monitor cable to the lower 15 pin connector of the board. The upper connector will be connected to the monitor. You will not damage anything by reversing the connections, it will become apparent when a Picasso II screen mode is selected that the connection is incorrect – the result will be a blank screen.



3.2. Software Installation

Software installation done will be using Commodore's Installer program. At the beginning of the program you will be asked to identify the level of your computer knowledge. Your choices are Expert User, Intermediate User, or Novice User. If you choose Novice, the program will install itself and not ask any further questions. Some programs will not be installed. If you choose Intermediate, the program will prompt with a choice of installing

various programs. If you choose Expert, you will be able to control fully the installation that can be dangerous in some instances. Before you choose Expert User, you MUST read the chapter describing monitor frequencies, resolutions, and fundamental monitor technology. This is important since the installation procedure will ask you questions that cannot be answered without the technical information contained within that chapter.

Upon completion of the installation procedure, your computer will automatically reset. The new screen resolutions should show up in the Preferences ScreenMode editor. Should this not occur, please read the chapter on troubleshooting. This chapter should help resolve any problems that may occur.

Unless otherwise specified, the installation procedure will install two commodities, ChangeScreen and StyxBlank, to the WBStartup directory. An AmigaGuide file is available to help you with ChangeScreen. As soon as you hit the Help key, (Workbench 3.0 or above), a window with all the ChangeScreen settings will appear, then simply click on the desired settings button for interactive help.

The included software can be explored by trial and error, or by reading the related chapters in this manual. The trial and error method has the advantage of satisfying your curiosity, as well as giving you a quick understanding of the basic software functions. We do, however, suggest that you read the appropriate chapters of this manual to gain a full understanding of the software.

4. Installing the Picasso II into the A4000

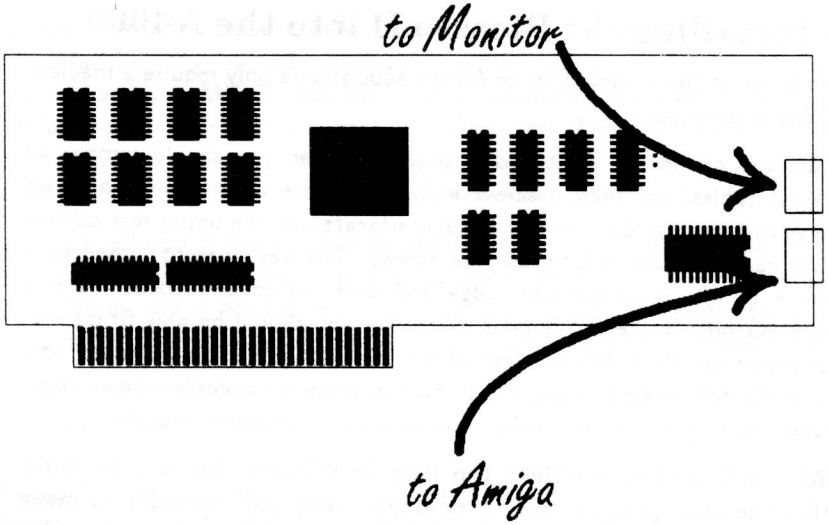
Installation of the Picasso II in an Amiga 4000 should only require a medium Phillips screwdriver.

Before you can install the board in your computer, you should remove all external cables and then unscrew and remove the cover. There are two screws that need to be removed which are located on the upper rear corners of the case. Be sure not to lose the screws. The best way to loosen your cover is to lift up on the back edge and slide the cover towards the rear of the machine. Once you remove the cover, you should be able to identify four expansion slots. On the rear of the machine you should also be able to identify four metallic tangs, each held in place by a single screw. These screws ensure that any expansion boards are held securely in place.

WARNING: Before installing the Picasso II board, be sure to touch either the case of the machine, or a grounding pad, in order to make sure that you are electrically discharged. If you do not do this, you run the risk of damaging any chips you might touch either on the Picasso II board or in your computer.

Decide which slot you will install the Picasso II in and remove the metal tang for that slot. Insert your Picasso II into that slot, making sure the board is securely inserted by applying pressure to the board. Use the screw you removed from the original tang to secure the tang of the board to the back of the machine.

Two connections must be made. First, you will need both the short cable provided and an adapter which gender changes the 23 pin RGB connector to a high density 15 pin connector. This adapter should have been included with your Amiga 4000. If you do not have this adapter, please contact your local dealer, or contact us directly. Install the adapter into the 23 pin RGB port, then insert one end of the short cable into the adapter and the other end into the connector of the Picasso II closest to the power supply of the Amiga. Next, connect the monitor cable into the remaining connector of the Picasso II. The Picasso II cannot be damaged by reversing these connections, but the board will not function correctly.



Before you replace the cover, please make sure that the Picasso II board is securely inserted and double check all cable connections. Next, replace the case using the opposite of the procedure described above, and replace all external cables previously removed. Power up the computer, if there are any problems, please refer to the chapter on troubleshooting in the back of this manual.

5. Installing the Picasso II into the A3000

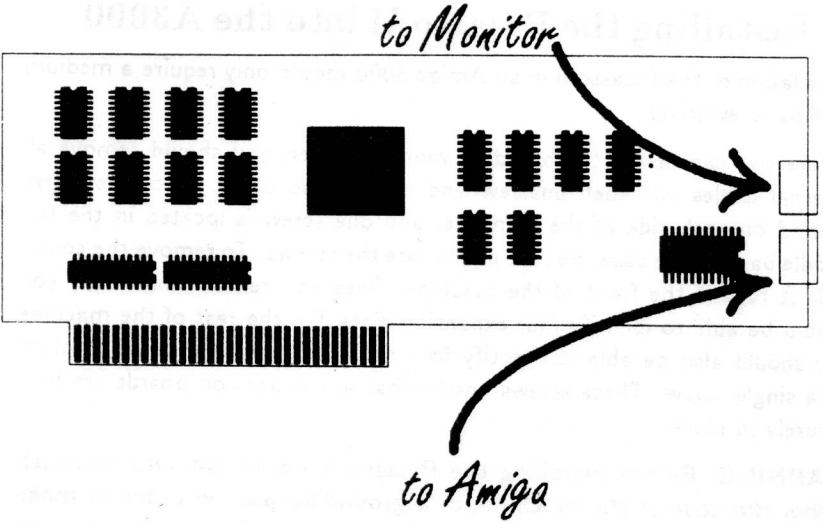
Installation of the Picasso II in an Amiga 3000 should only require a medium Phillips screwdriver.

Before you can install the board in your computer, you should remove all external cables and then unscrew and remove the cover. Two screws are located on each side of the computer and one screw is located in the top middle back of the case. Be sure not to lose the screws. To remove the cover, slide it toward the front of the machine. Once you remove the cover, you should be able to identify four expansion slots. On the rear of the machine you should also be able to identify four metallic tangs, each held in place by a single screw. These screws ensure that any expansion boards are held securely in place.

WARNING: Before installing the Picasso II board, be sure to touch either the case of the machine, or a grounding pad, in order to make sure that you are electrically discharged. If you do not do this, you run the risk of damaging any chips you might touch either on the Picasso II board or in your computer.

Decide which slot you will install the Picasso II in and remove the metal tang for that slot. Insert your Picasso II into that slot, making sure the board is securely inserted by applying pressure to the board. Use the screw you removed from the original tang to secure the tang of the board to the back of the machine.

Two connections must be made. First, you will need the short cable provided with your Picasso II. Insert one end of the short cable into the 15 pin VGA port of the computer and the other end into the connector of the Picasso II closest to the power supply of the Amiga. Next, connect the monitor cable into the remaining connector of the Picasso II. The Picasso II cannot be damaged by reversing these connections, but the board will not function correctly.



Before you replace the cover, please make sure that the Picasso II board is securely inserted and double check all cable connections. Next, replace the case using the opposite of the procedure described above, and replace all external cables previously removed. Power up the computer, if there are any problems, please refer to the chapter on troubleshooting in the back of this manual.

6. Installing the Picasso II into the A2000

Before you begin installation of the Picasso II in an Amiga 2000, (which does not have a built-in deinterlacer), it is important that you obtain either the correct adapter for the 23 pin RGB port, or the correct cable for the flickerFixer.

Installation of the Picasso II in an Amiga 2000 should only require a medium Phillips screwdriver.

Before you can insert the board in your computer, you should remove all external cables, unscrew five screws, and remove the cover. Two screws are located on each side of the computer and one screw is located in the top middle back of the case. Be sure not to lose the screws. To remove the cover slide it toward the front of the machine. Once you remove the cover, you should be able to identify seven expansion slots. On the rear of the machine you should also be able to identify seven metallic tangs, each held in place by a single screw. These screws ensure that any expansion boards are held securely in place.

WARNING: Before installing the Picasso II board, be sure to touch either the case of the machine, or a grounding pad, in order to make sure that you are electrically discharged. If you do not do this, you run the risk of damaging any chips you might touch either on the Picasso II board or in your computer.

Decide which slot you will install the Picasso II in, and remove the metal tang for that slot. Insert your Picasso II into this slot, making sure the board is securely inserted by applying pressure to the board. Use the screw you removed from the original tang to secure the tang of the board to the back of the machine.

Now, depending on your configuration, two connections must be made.

6.1. Amiga 2000 without a Deinterlacer

For the first connection, you will need the short cable provided and the adapter which gender changes the 23 pin RGB connector to a high density 15 pin connector. This adapter can be purchased from your dealer, or by calling us directly. Install the adapter onto the 23 pin RGB port, then insert one end of the short cable into the adapter and the other end into the bottom connector of the Picasso II. Now connect the monitor cable into

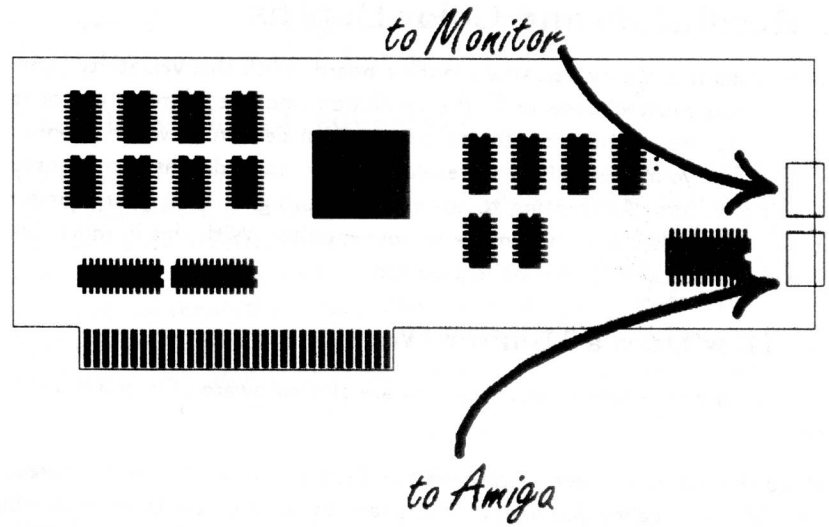
the top connector of the Picasso II. The Picasso II cannot be damaged by reversing these connections, but the board will not function correctly.

6.2. Amiga 2000 with A2320 Deinterlacer

For the first connection, you will need the short cable provided. Insert one end of the short cable into the A2320 and the other end into the bottom connector of the Picasso II. Now connect the monitor cable into the top connector of the Picasso II. The Picasso II cannot be damaged by reversing these connections, but the board will not function correctly.

6.3. Amiga 2000 with MicroWay flickerFixer

For the first connection, you will need the short cable provided and an adapter which gender changes the 9 pin connector of the flickerFixer to a high density 15 pin VGA connector, or a flickerFixer to high density 15 pin male cable. This adapter can be purchased from your dealer, or by calling us directly. Install the adapter on the monitor connection of the flickerFixer, then insert one end of the short cable into the adapter and the other end into the bottom connector of the Picasso II. Now connect the monitor cable to the top connector of the Picasso II. The Picasso II cannot be damaged by reversing these connections, but the board will not function correctly.



Before you close the case, please make sure that the Picasso II board is securely inserted and double check all cable connections. Now replace the case using the opposite procedure as described above, and replace all cables previously removed. Power up the computer and if there are any problems, refer to the chapter on troubleshooting in the back of this manual.

7. Resolution and Color Depths

The Picasso II is a very versatile graphics board. With this versatility come choices. You must choose both the resolution and the number of colors you want to use. With this information, you can determine which monitor will best suit your needs. Not all resolutions and color depths can be used with all monitors. Attempting to use resolutions higher than your monitor can support could result in damage to your monitor. With this in mind, the following technical information is provided.

7.1. How Does a Monitor Work?

If you know how a television works, you are already aware of how a monitor works.

Behind the monitor screen, a cathode ray beam is projected onto the screen at a high rate, called frequency. The beam consists of electrons that will light up, for a very short time, on the screen. These electrons are projected on the screen from left to right, row by row, very rapidly. The individual dots that light up on the screen are also known as pixels, and will fade very fast.

To avoid this weakness, the individual pixels that make up the screen have to be refreshed fast enough every second so that your eyes do not notice the fading action. The number of times the screen is entirely refreshed per second determines the monitor frequency. The higher the frequency, the better the picture will appear to your eyes. High frequencies will diminish flicker on your screen, which provide a more stable picture.

You have certainly noticed the flicker resulting from your Amiga's interlaced mode, running at 25 or 30 Hertz (Hz), in other words 25 or 30 pictures per second. A non-interlaced picture usually runs at 50 or 60 Hertz (Hz). Even at this high frequency your eyes will still notice a slight flicker. Computer experts have determined that a 70Hz or higher frequency is required to make the flicker unnoticeable to the human eye.

7.2. When Color is Added

The previous description only applies to black and white screens. Color monitors are more common these days, and the technology used is only slightly different. A color screen requires three beams of electrons: a green beam, red beam, and blue beam. Depending on the mixture of electrons hitting the screen, various shades of colors can be displayed. If all three beams hit the same spot, a white pixel lights up.

7.3. The Various Frequencies

The Picasso II is connected to a monitor at a certain frequency. The graphics card is responsible for sending information, which is displayed as graphics, to the monitor. The correct frequency, color mapping, and screen sizing information has to be transmitted to the monitor by the Picasso II. Your monitor cable has two connections responsible for the V-Sync (vertical sync) and H-Sync (horizontal sync). The H-Sync determines when the beam movement has to stop and jump to the next line on the horizontal axis. The V-Sync determines when the beam movement has to stop and jump back to the top of the screen on the vertical axis. This information must also be transmitted to the monitor by your graphics board.

Besides the previously mentioned frequencies, there is a vertical frequency that is important. The number of lines the monitor, in conjunction with the graphics card, can produce on the screen per second is called the vertical frequency. Let's assume that you have chosen as your ScreenMode, NTSC:High Res Laced, and you have a deinterlacer installed in your machine. With overscan you will now achieve a resolution of 720x480 pixels, a picture frequency of 60Hz, and the vertical frequency resulting from this is 28.8kHz (480 lines x 60Hz = 28800Hz).

Unfortunately all monitors cannot display every vertical frequency. A normal 14" monitor usually has a maximum vertical frequency of 38kHz. This is enough to display 800x600 pixels on the screen at 60Hz. If you want to display 1024x768 pixels at 70Hz, you would have to buy a better multiscan monitor. These monitors can easily display up to 57kHz vertical frequencies.

At the end of this manual you will find a table listing the different resolutions and frequencies that the Picasso II outputs.

7.4. Segmentation

Amiga 2000 owners are limited to 8 megabytes of fast RAM, unless they own an accelerator card with its own 32-bit memory.

Should you wish to install the Picasso II in an Amiga 2000 with 8 megabytes, you will not be able to access the memory on the Picasso II since it needs its own memory range to address the Video RAM. A quick solution would

be to remove some of your Fast RAM, which is usually done in two or four megabyte increments. This solution is usually not very acceptable.

Therefore we should take a different approach, segmentation. In this process the Video RAM is divided into 16 partitions. This memory range will not be used normally, but instead put into the Input/Output area of the Amiga. The advantage is that you can use your graphics board with a Amiga 2000 expanded to 8 MBytes.

However, some disadvantages result from segmentation. By using memory segmentation, the programmer has to choose the right segment before drawing into the video memory. Since this is controlled by the I/O area of the computer, the process of drawing lines, text, or other objects is going to slow down.

All but one of the additional programs that come with your Picasso II will not work with segmented memory. Only the intuition driver will work in this mode. The intuition driver will be explained later in this manual.

For programmers, the segmentation solution involves more complication. For this reason, segmentation should only be considered a temporary solution. We will try to modify our programs to make them more segmentation compatible. We cannot, however, guarantee that other software developers of 24-Bit-Software will adjust to our segmentation mode.

Should you have the problem described above, we suggest that you invest in an accelerator board. This would allow you to increase the amount of memory in your Amiga 2000 past the 8 MByte limit. Therefore your Picasso II would be able to run without restrictions.

Should you, however, use your Picasso II with segmented memory, you will have to move a jumper on the board. Please read the chapter on Jumpers and Segmentation.

7.5. 24 Bit At 1280 x 1024 Pixels

Besides the picture frequency, measured in Hertz (Hz), and the horizontal frequency, measured in kiloHertz (kHz), there is a pixel frequency, also called video bandwidth, which is measured in MegaHertz (MHz). This value indicates how many pixels per second are displayed on the screen. Sometimes, this value indicates the number of bytes that are read from the Video RAM and displayed on the screen. This number depends on the organization of the memory and the video card.

Cards with DRAM (Dynamic RAM) operate in such a way that 1 byte equals 1 pixel. Besides DRAM you can use VRAM (Video RAM), which is more expensive but allows for a higher pixel frequency. Most graphic cards available to consumers use 1 MByte of DRAM on the card.

How is the pixel frequency calculated? Vertical Frequency x Pixels per Line x 1.1 = Pixel Frequency. The last term, 1.1, represents the time it takes the video beam of a monitor to move from the bottom right corner to the top left corner of the screen. Typical results using this calculation are: 60, 70, 80, or 90MHz. Values slightly above 90MHz are usually the boundaries for cards using DRAM.

The Picasso II has 1 MByte of DRAM that is enough to display 640 x 480 pixels in 16.8 million colors, also called TrueColor mode. This is possible as shown by the following equation:

$$640 Pixels * 480 Lines * 3 Bytes/Pixel = 921,600 Bytes$$

which means that nearly 1 megabyte of RAM is being used. To obtain 16.8 million colors 3 Bytes/Pixel are used.

The following calculations will determine what the pixel frequency would be:

$$72 Hz * 480 Lines * 1.1 \approx 38,000 Hz$$

this corresponds to a horizontal frequency of 38kHz. In order to calculate the pixel frequency, there has to be included in the calculation the amount of Bytes per Pixel (BpP) and multiply it by 1.25. This will result in the following calculation:

$$38,000 Hz * 640 Pixels * 3 BpP * 1.25 = 91,200,000 BpS$$

BpS stands for Bytes per Second, which indicates the pixel frequency. In other words, a little over 90 million Bytes per Second (BpS) are being transferred to the monitor. This results in a video bandwidth, or a pixel frequency, of 90MHz.

For the next higher resolution, 800 x 600 pixels, you would require 2 MBytes as shown by the following equations:

$$72Hz * 600Lines * 1.1 \approx 47.5kHz$$

and

$$47,500Hz * 800Pixels * 3BpP * 1.25 = 142,500,000BpP$$

By only using 60Hz the following equation is the result:

$$60Hz * 600Lines * 1.1 = 39.6kHz$$

$$39,600Hz * 800Pixels * 3BpP * 1.25 = 118,800,000BpP$$

If someone told you that they could display 1280x1024 Pixels in TrueColor, it means that they would require 4 MBytes of memory. But the question remains, what horizontal and pixel frequency are they using? A small calculation will give us the answer. Lets take the maximum pixel frequency or video bandwidth, which we determined to be around 90 MHz, and divide it by the amount of pixels, the bytes per pixel, and the factor of 1.25, to calculate the horizontal frequency:

$$\frac{90,000,000Hz}{1280Pixels * 3BpP * 1.25} \approx 18,700Hz$$

With this horizontal frequency of about 19kHz you can only use older 14 inch monitors which are quite slow. VGA frequencies only begin at 31kHz, however, older 14 inch monitors do not have the resolution to display 1280 Pixels. Now lets determine which scan rate would be required:

$$\frac{18,700Hz}{1024Lines * 1.1} = 16.6Hz$$

16Hz flickers twice as bad as the PAL interlaced resolution without even considering the fact most monitors cannot display 16.6 Frames per Second, which is very slow.

In addition, you should decide when using 4 MBytes, whether or not you are using the right CPU in your computer. Using good programming, the Zorro

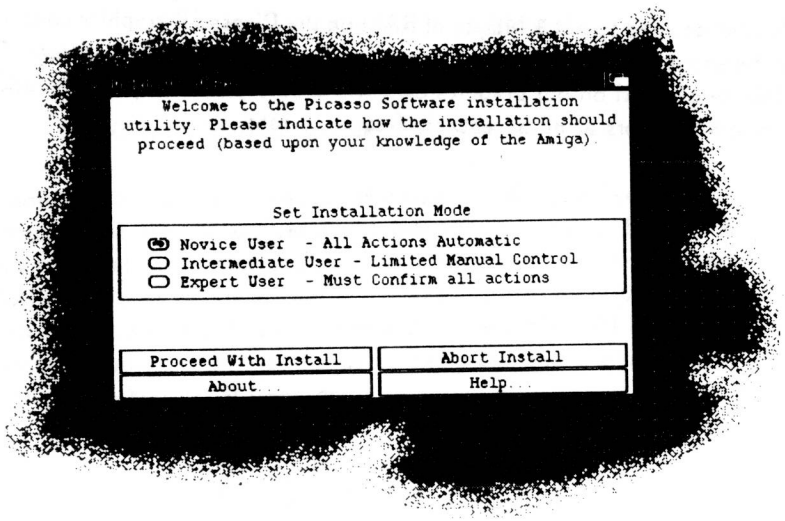
II bus can only transfer 3 MBytes per Second. Even using a 68040 processor this technical limitation can not be changed.

Go ahead and make the previous calculations. By making the previous calculations you can determine easily whether the graphics cards offered on the market are really promising true performance. In addition to all the preceding material, calculations involving transfer rates for animations are quite interesting. For a 320x200 pixel animation in TrueColor, you will require 4.8 MBytes of memory per second to be transferred into your video RAM in order to display 25 Frames per Second.

We have decided to put 1 MByte of RAM on the Picasso II graphics card, this will be enough for the highest non-interlaced resolution with 256 colors. The 1 MByte of RAM on our graphics card will enable you to display resolutions with lots of colors at an acceptable frequency rate.

8. Software Installation

The installation of the software is quite easy. Insert disk 1 and click twice on the Installation icon, Install English. After clicking twice on this icon, the standard installation requester from Commodore will appear.



The installation does not modify any of the important system files such as S:User-Startup, S:Startup-Sequence, or DEVS:Mountlist. The installation process only copies a few files into the correct directories.

Among these files are:

The Village library	to the Sys:Expansion drawer,
The Picasso Monitor file	to the DEVS:Monitors or Sys:WBStartup,
The PicassoSwitch	to the Sys:WBStartup,
StyxBlank	to the Sys:WBStartup, and
ChangeScreen	to the Sys:WBStartup.

If you are not familiar with your Amgia, please click on Novice in the installation requester. The program will then install all programs needed without asking any further questions.

If you are familiar with the Amiga you can click on Intermediate User, which will give give you the choice to install or not install certain programs which are included with your graphics card.

If you are an Expert User you can choose to click on Expert. You will be asked questions that will determine every aspect of the installation procedure. This mode will even allow you to set the kiloHertz frequencies which your monitor can handle. Be careful that your selection is not too high, or possible damage may occur to your monitor.

Once you start the installation procedure, you may encounter a problem identifying the buttons we discussed previously. They may be missing and you will have to change your system font to Topaz 8 in order to correct this problem.

In order to change the system font to Topaz 8, you must go into the preferences font editor and change your system font to Topaz 8. This will default the system to Topaz 8.

9. The Intuition Driver

Of what use would the best graphics card be if you could not use it with every program? In fact, you would want to use as many colors and as high a resolution as possible, as often as possible. This requires software to combine both the graphics card and computer to make it look as if they were always designed for each other.

Unfortunately, Commodore has not yet enhanced the operating system to use modular graphics cards, like the Picasso II, without any problems. The modular graphics card environments, like the Apple Macintosh and Micro-Soft Windows environments, are not yet available for the Amiga line of computers. Therefore, the Picasso II solution is not yet 100% compatible, even though we have tried to put as much effort as possible into increasing the compatibility of the board.

In order for the graphics card to conform to the system closely, we have used two mechanisms that the Amiga operating system uses. The card and its functions are being controlled by a library, called the village.library. The usage of new resolutions is performed though a monitor file, called Picasso.

9.1. village.library

The village.library has two tasks:

1. It is the link between the card and the intuition driver.
2. It chooses the correct resolution for the monitor, and only allows the user to use resolutions that are possible with the monitor in question.

The village.library must be located in Sys:Expansion to function properly. Usually the installation program copies the library into this directory, Sys:Expansion, so that you should not have to worry about it.

Important: In your S:Startup-Sequence you must be running the command Binddrivers. If you do not do this, other Picasso II programs will not be able to locate village.library and your graphics card will not function properly. The standard Startup-Sequence usually does this automatically, but if you have altered your Startup-Sequence and removed the Binddrivers command you will have to replace it for proper operation.

In the preceding chapter regarding the technical aspects of a monitor, we found that there are monitors that can display vertical frequencies ranging

from 38kHz to 64kHz. The icon of the village.library has a line in the Tool Types that reads,

```
MONITOR=38kHz
```

which you can see by single clicking on the icon and selecting information from the Workbench pull down menus, or by selecting AMIGA-i. If you own a monitor which can display 57Khz and above, you can modify this line to read,

```
MONITOR=57kHz
```

Important: Make sure your monitor can display these higher frequencies or you can damage your monitor. Village Tronic Marketing and Expert Services will not be responsible for any damage which may occur to your monitor resulting from a higher frequency selection. After making this change, you will have to reboot your computer in order to activate this higher frequency.

Possible vertical frequencies are: 38, 50, 57, 64 kHz. If you enter a different value you will experience a system error when starting your system. Therefore you must use only one of these four frequencies.

The entry,

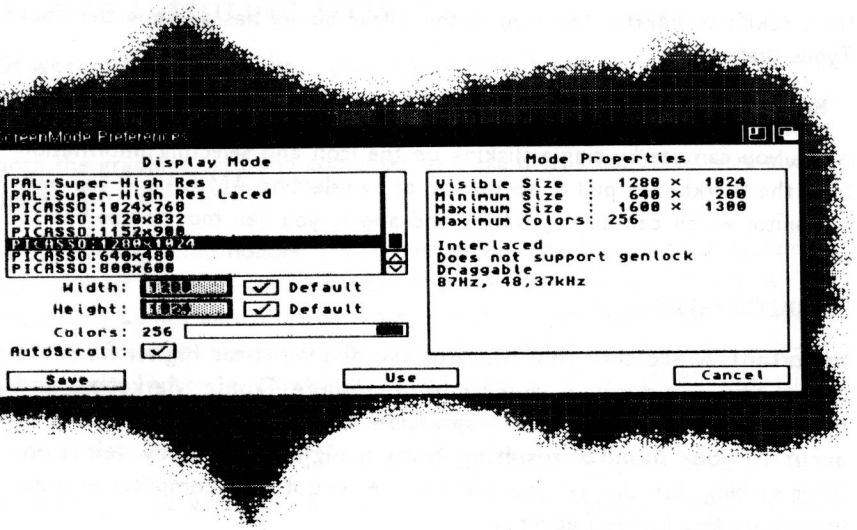
```
PRODUCT=...
```

must absolutely stay the same or your library will not be linked to the operating system.

9.2. Picasso-Monitor File

Without the intuition driver of your Picasso, your board would be difficult to use. Like other graphics cards, you would need programs that would specifically be written for your graphics card. This would be expensive and not a very good solution.

Therefore, we have chosen to expand the important operating system routines in such a way that they will work in conjunction with your Picasso II. This required much work, but the results are more than satisfactory. To activate the Picasso driver, you only have to double click on the Picasso icon. Then by double clicking on the program ScreenMode, which is found in the directory Sys:Prefs, you will be presented with a listview displaying the new resolutions available:



The ScreenMode Program with the Picasso Resolutions

What the resolution will be that you can choose from is determined by your monitor and how high a vertical frequency it can display. With better monitors you will have more choices since they are capable of higher resolutions.

A few programs, ones which require operating system version 2.0 and above, will also allow the user to choose what screen mode to use. This is usually done using the same procedure described above.

For programs which are programmed to run on operating systems version 1.2 or 1.3 we have provided a program called ChangeScreen, which will be described later in this chapter. This program will allow you to promote older screen modes to the new screen modes.

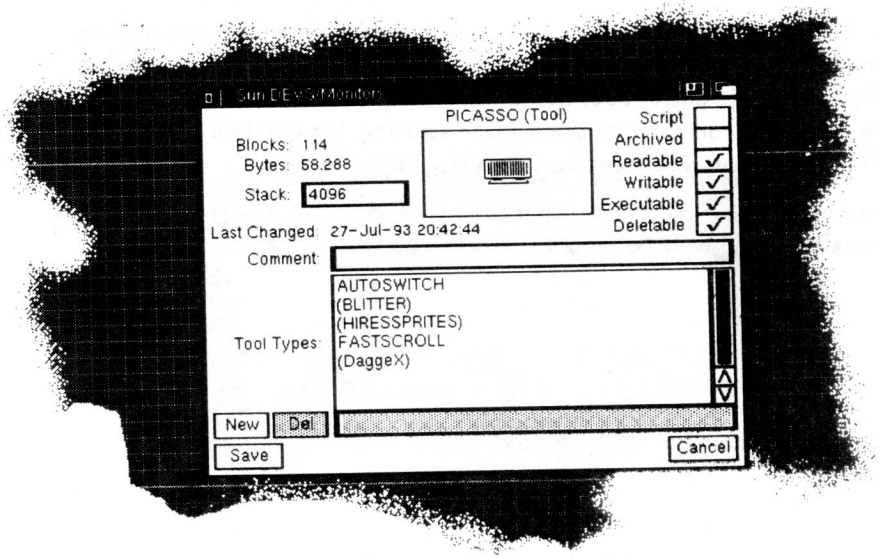
With our new monitor file you will get more than just better resolutions. You will also be able to open screens with 256 colors, even if your machine does not have the AGA chip set. You will have to be running operating system 3.0 or higher to do this, however. Also, when using these new higher resolutions no chip memory is required, leaving it free for the other customs chips to use, for things such as disk buffers.

Just like the library, the Intuition driver has four settings which can be adjusted by clicking on the icon and selecting information from the Workbench pull down menu. First there is:

```
AUTOSWITCH
```

If AUTOSWITCH is listed in the Tool Type list, the software will automatically switch between the Amiga screen and the Picasso II screen depending on what is being displayed. To disable the AUTOSWITCH function, all you have to do is put parenthesis around it. This would look like:

```
(AUTOSWITCH)
```



After single clicking on DEVS:Monitors/Picasso and pressing AMIGA-i

If you are running Workbench 3.0 or above, you can choose which kind of mouse pointer you want to use. This is determined by the line

```
HIRESSPRITES
```

If HIRESSPRITES is active and not in parenthesis, you will be using a small, high definition, mouse pointer. If HIRESSPRITES is missing, or in parenthesis, you will be using a normal 16 pixel wide LoRes mouse pointer that will be displayed in double size. It is recommended when working in

high resolution you use the larger mouse pointer since the smaller one will be difficult to see on the screen.

The third entry is,

BLITTER

which allows the Intuition Driver to use the Amiga Blitter. This is useful when your machine does not have a fast CPU, such as a 68030 or higher. Using this function creates a disadvantage, the screens will have to be loaded into ChipMem.

The fourth entry has to be used with caution, it is called,

FASTSCROLL

This function speeds scrolling considerably, however some programs may be incompatible. When using this function, all the planes of the screen are being scrolled. Since the blitter does not have to work as hard when all planes are being moved, as compared to single plane scrolling, the Intuition Driver with FASTSCROLL will work considerably faster.

Remember that whenever you change any of these settings that we have discussed you will have to reboot your computer to activate them.

10. Programs

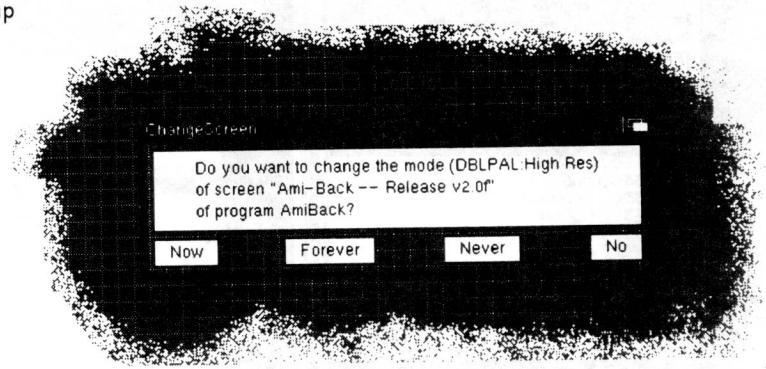
10.1. ChangeScreen

Some programs won't allow the user to determine the resolution or the amount of colors that should be displayed on the screen. For that reason we have enclosed the commodity ChangeScreen which will allow programs like those to run with your Picasso II board.

The installation procedure copies ChangeScreen to your directory `Sys:WBstartup` so that it will be active all time. If you only need it occasionally, it would be best to move ChangeScreen to a different directory and run it only when needed.

10.1.1. Function

Each time a program starts its own screen, the following message will come up



The 4 gadgets (elements to click) have the following meanings

Now: if you click NOW, the screen will only be customized one time for the Picasso II board. Before this happens, another requester will come up asking for the desired resolution. Next time you start the program which asks for such a screen, the same requester will appear.

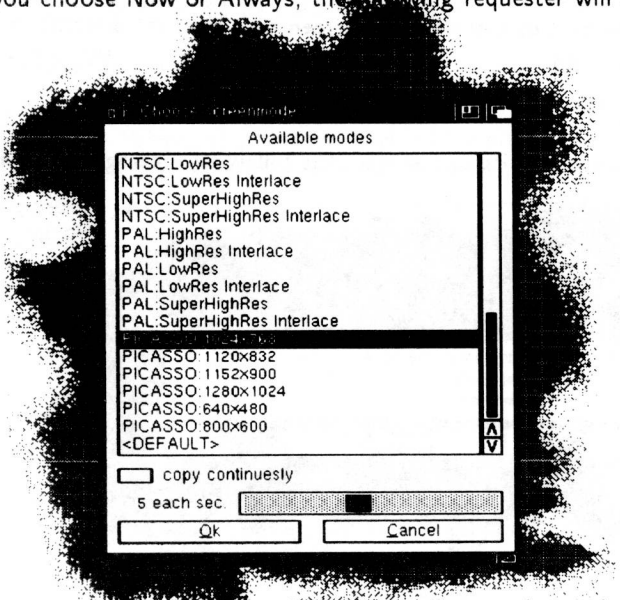
Always: if you click on ALWAYS, you can determine a permanent screen promotion for the specific program you are starting. When opening the same program again, this requester will not appear. The screen will immediately be changed to the

same resolution without asking. If you click on always, you will get the choice to enter the resolution you want.

Never If you click on NEVER, ChangeScreen will never come up with the requester again asking whether to modify the screen or not, the screen will alway be in its default mode.

No If you click NO, the next time you start the program the requester will come up again, but this time the resolution will not be affected.

f you choose Now or Always, the following requester will appear:



You can choose a new resolution according to the choices that appear in the window. This resolution does not have to be a Picasso resolution. If you do not want to ChangeScreen a resolution, click CANCEL. After choosing a resolution, click on OK to switch to that resolution. The program will automatically adjust the screen if the screen that comes up is smaller than the one you chose.

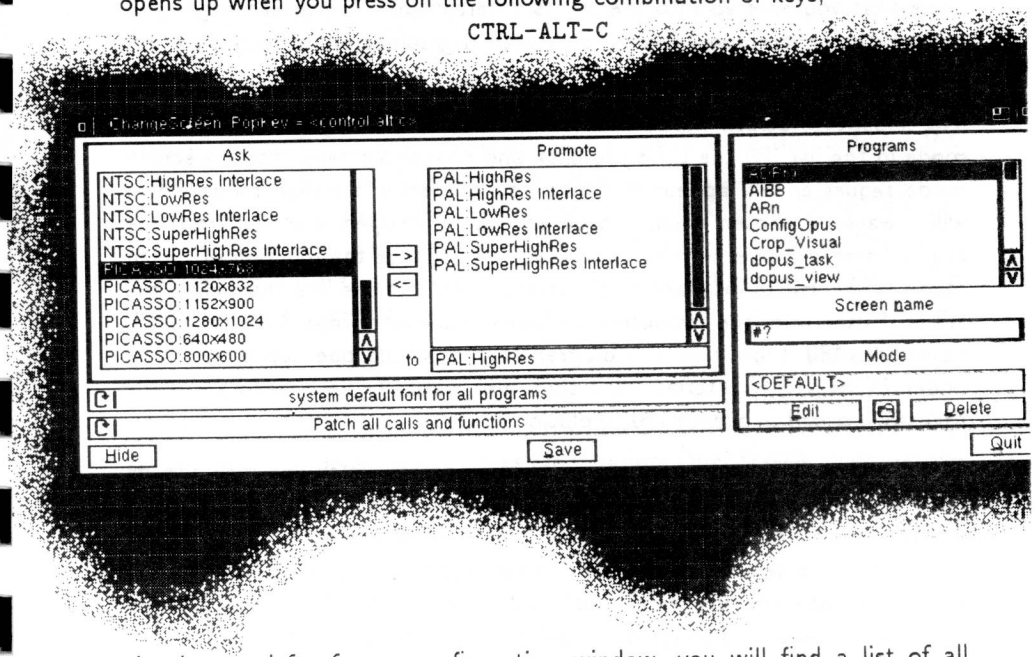
Underneath the resolution box you will find two additional gadgets that can be clicked. If you click on COPY CONTINOUSLY, the screen will not be changed to a different resolution, in fact, the screen will be opened in the background and copied to chip ram, whose contents will continuously be copied into the Picasso's II screen. Some programs do not use system

functions and directly access the screen memory, examples of such programs are older DPaint versions, these type of programs would require you to click on the option continuous copy. Many animation programs also belong in that category. The other choice underneath the resolution box allows you to indicate the number of copies per second, in other words, how many times your screen should be renewed per second.

10.1.2. The Configuration Window

Of course, even after carefully setting your options, some programs will still not work properly. This is when the configuration window comes in handy. The configuration window allows you to modify the settings which you entered when you clicked on Always or Never. The configuration window opens up when you press on the following combination of keys,

CTRL-ALT-C



At the top left of your configuration window, you will find a list of all available screen modes. Over this window you will see the words "Ask" when a program wants to open a screen that is in the list, ChangeScreen will ask whether to change the screen or not. Generally this is the task of the ChangeScreen program, however in some instances it would not be useful to have this prompt coming up since some programs will automatically open up a screen in the screen mode (ie. Picasso:800x600). Obviously the Picasso

II graphics board supports such screen resolutions, and no major changes would have to be made.

In such instances you can move a screen mode from the left list to the middle list, marked automatic. All you need to do is click on the desired screen mode in the left list and move it over to the middle list by clicking on the arrow pointing to the right. Now that you have added a screen mode to the middle list, any program that needs this screen mode will automatically open up this screen without prompting you.

In addition to the function just described, the middle list has a second function as well. Underneath the middle list you will find a line in which after clicking on the screen mode in the middle list, the same screen mode will appear. This screen mode is also called substitute screen mode. If the value that you entered into the box is different from the screen mode you clicked, the program will force the Picasso II graphics board to choose this substitute screen mode.

The substitute screen mode can be modified by clicking first on a screen mode in the list. By the time you click and choose a screen mode, a screen mode requester will appear in which you can enter a screen mode which will always be chosen when a program wants to open a screen from the original screen mode. In other words, you can change the function of screen mode NTSC:HiRes interlace to Picasso:640x480, this is like channeling the NTSC:HiRes interlace resolution to become the new resolution called Picasso:640x480. From now any programs which would open up a resolution under NTSC:HiRes interlace will then automatically be opened from the Picasso II board in a 640x480 resolution.

The far right list has a series of names of programs which you may already have modified or channeled by clicking on Always or Never. Underneath that line is a line titled Screen. In this line, you will find the name of the screen when you have selected a program name in the box above it. Underneath that screen title you will see the resolution in which the chosen program will function. Instead of displaying a resolution, the words '<DEFAULT>' can be seen which means that you must have clicked on Never one time when prompted.

The new resolution can be changed in this window anytime by clicking on a program and by clicking on modify. A screen mode requester will appear in which all the screen modes and the entry '<DEFAULT>' will be listed. Choose one of the screen modes and click on OK.

Of course you can also erase entries from the right list by simply choosing the entries and clicking on DELETE.

Between the two gadgets, MODIFY and DELETE, you can see a gadget with the symbol device. By clicking this gadget, you can enter programs into the right list without necessarily starting them. After clicking on the device symbol, you will see a data requester open up in which you can select the desired program. Because ChangeScreen doesn't know what the screen title will be, you need to enter it into the program list yourself. When entering the title, you can use the Amiga DOS sequence #?. If you only enter the #? as a title, all the screens of the selected program will be changed and channeled into the new screen mode.

Under both lists, the one in the middle and the far left, you will find two cycle gadgets. The first gadget under these two lists can choose between Topaz 8, which is useful for older programs, or the standard system font for all other programs, especially newer ones. Many older programs will assume that without selecting a font, you are wanting to use the font Topaz 8. After Workbench 2.0, this rule is not true anymore since the fonts for the workbench, screens, and other uses must be set. If you select Topaz 8 for older programs, all your screens that the Picasso will generate will display Topaz 8. This will help in many instances to restore menus and text, and allow you to have a much better way of reading your screen, even if you have selected a larger screen font.

The second cycle gadget can choose between modify all new screens or prompt only after OpenScreen(). By choosing the first setting, all the screens that will be opened will be prompting the user for the screen resolution. The second setting will only prompt if the old function, OpenScreen, is being used and not the OpenScreen tag list which is being used after Workbench 2.0, which means that only older programs will prompt if a screen has to be channeled or not.

10.1.3. Menus

All settings can be loaded and saved. This can be done with the menus project/open and project/save. By choosing any of these menu options, a data requester will appear asking for the data file that you wish to either save or open.

Usually ChangeScreen will save those settings into a data file called ENVARC:ChangeScreen.prefs. This happens, for instance, automatically whe-

never you click on a modification screen such as Never or Always. The settings will be saved as normal text which you can edit in a text editor. This is not recommended, since you can introduce inconsistencies by modifying through a normal text editor which can later be interpreted wrongly by ChangeScreen.

For that reason, we are not going into any further detail to describe the data files which are being saved or opened. By clicking on About a window will appear in which you can see the name of the author, the entry date, and a version number of the program. This could be useful in case you want to inform us of a mistake or error, or to determine whether you are using the newest version of the program or not.

The function Hide will close the window without ending the program. Since it is run as a commodity, you will always be able to retrieve your program by hitting the key combination CTRL-ALT-C. This key combination can be altered for your specific needs, which we will describe in the next paragraph. When clicking Quit, you will not only close the window, but you will also end the program.

10.1.4. Tool Types

ChangeScreen can be started from your workbench or from a shell environment. The parameters for ChangeScreen can be found under tool types or shell parameters.

The parameters are;

`SETTINGS`	the name of the configuration data file
`NOICON`	the configuration data file without saving any icons
`ONLYOLDSTYLE`	whether OpenScreen() should be patched or not
`FORCETOPAZ`	old programs should be using Topaz 8
`CX_PRIORITY`	commodity priority
`CX_POPKEY`	command keys to open the system requester for your graphics setting
`CX_POPUP`	whether or not to open up the setting window at startup

Behind SETTINGS you can indicate a data file name in which your settings will be looked for. This way you can use greater or fewer data files with SETTINGS, and while operating, can change those data files as necessary.

Usually ChangeScreen will save its configuration always with an icon. Should you not want this to happen, then indicate NOICON in your setting.

Should only old OpenScreen() routines be prompted for new screens, use the function ONLYOLDSTYLE. This function will work well with, and make sense to use together with, FORCETOPAZ. In other words, screens that use Topaz 8 (usually older screens) will also be prompted for the resolutions.

CX_PRIORITY, CX_POPKEY, and CX_POPUP are the standard tool types which each commodity understands. The Commodore manuals that come with your computer will give you more insight into this.

10.1.5. Interactive Help

Should the previous explanation not suffice to give you a clear picture about the procedure required to make your Picasso II board compatible with your programs, or should you loose your manual, you can always get help by hitting the help key. This is going to open a window of the program Amiga Guide with which you can obtain more information regarding these proceedings.

Help on how to use Amiga Guide can also be obtained by hitting help a second time after opening Amiga Guide. Since you can completely guide your way around Amiga Guide with your mouse and menus, you should not have too many problems with this part.

10.1.6. Warning

IMPORTANT: ChangeScreen is very important, however sometimes it won't be system compatible. We cannot guarantee that every program will work flawlessly with your Picasso II board.

Many programs expect that you open up a very specific screen and not one that is any bigger. You have to try out and modify your screens until your program will work and be compatible with the Picasso II board. Should you not be successful the first time, try the copy mode. Using this mode may cause your Amiga to produce a guru meditation. We cannot be held responsible for any damages that may result from trying various screen modes.

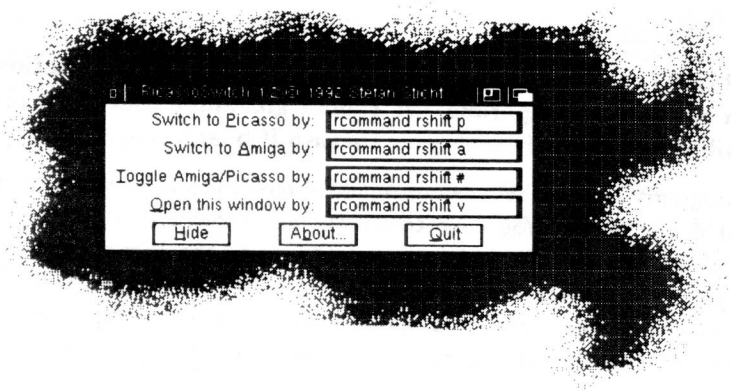
10.2. PicassoSwitch

It may sometimes be useful to switch between Amiga graphics and Picasso II graphics. Generally this is not necessary because the intuition driver will automatically take care of this procedure.

For test reasons, or after getting a guru meditation, it might be helpful. For this reason we have included a program called PicassoSwitch. It is a commodity that allows the user three different functions. These functions can be activated with the following key combinations:

To Picasso	(Shift Amiga P) switch to PicassoII GFX
To Amiga	(Shift Amiga A) switch to Amiga GFX
Toggle	(Shift Amiga T) toggles between Picasso and Amiga modes

These key combinations can be adjusted to your personal needs. To do this, single click on the PicassoSwitch icon and select information from the Icon menu of Workbench. If the program is already running, a window with the commodity exchange will appear. This configuration window looks like this:



The settings in this window are only temporary settings. After resetting the computer, or switching it off, these values will be restored to their original settings. To reset permanently the key combinations you must save the new Tool Type settings of the PicassoSwitch icon.

10.3. StyxBlank

In addition to the software described previously, we have included a screen blanker that will blank your screen after a certain amount of time. This will blank your screen and prevent monitor burn in. This will only happen if you do not click on a key or move your mouse for a specified amount of time.

The use of a screen blanker is easy to understand if you understand the aging process of a monitor. The older a monitor gets, the weaker the pixels will light up. Over the years if you use one picture repeatedly, it may happen that this particular picture will be "burned in" the screen. In order to avoid, or at least delay this process, screen blankers have been developed. These will darken the screen after a certain amount of time in which no keyboard or mouse activity is detected. The pattern you will see on the screen will help you determine if the monitor is still on. Specific parts of the screen will not be burnt in due to the constant movement of the image over the entire screen.

The simple program that we have included to do the screen blanking for you is called StyxBlank. It recognizes three parameters that you can set from either the icon Tool Types, or by manually entering them from the keyboard.

These 3 parameters are:

`LINES`	number of lines on the screen.
`SPEED`	time period between drawing to consecutive line in thousandths of a second
`BLANKTIME`	time in seconds until the sreen blanker will activate if you do not use the mouse or keyboard.

for instance, the command may look like this:

```
run StyxBlank LINES=200 SPEED=2 BLANKTIME=120
```

If after 2 minutes (120 seconds), there is no keyboard or mouse activity, a blank screen will appear and 20 lines will move around at an interval of 2 thousandths of a second.

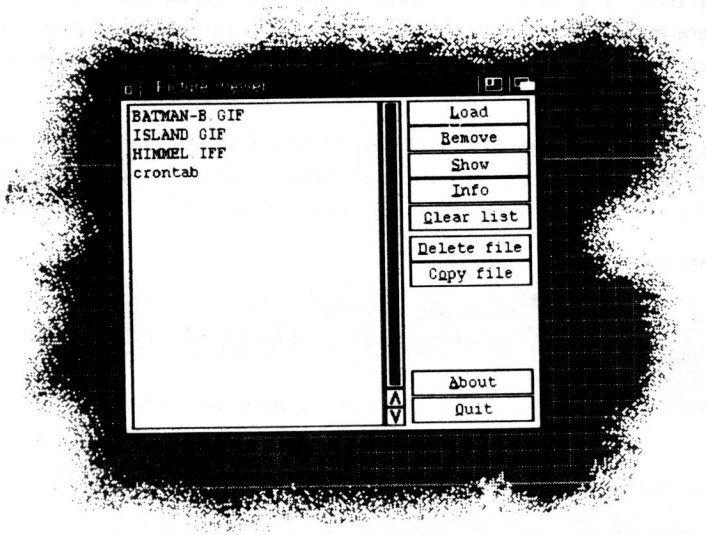
You can exit the program by hitting CTRL - C.

0.4. IntuiView

ntuiView only functions well in conjunction with other programs. You ould use IntuiView to display pictures, text, archives or any other formats at are available from your Workbench. IntuiView possesses a very easy nd configurable analysis system that can be expanded to suit your needs.

itially **IntuiView** was developed to help CLI novices facilitate the usage f picture displaying programs. Thanks to its flexible concept, you can use **ntuiView** for other uses.

After starting the program from either the Workbench or the Shell, you will ee the following window:



The left side of the window contains the list of file names. The right side of the window contains a series of gadgets that indicate various actions that can be performed.

10.4.1. Usage

To display a file, whatever the format and content, click on the name in the listview list and then click on **show**. **IntuiView** will try to figure out the format of the file and start the program to display the file. There are two different ways to add file names to the list:

- click on **load**, the file requester will appear, and you can choose a file
- click on the Workbench icon of the file and drag the icon into the **IntuiView** window
- choose **load** from the menu, which will make a file requester appear. This function can also be achieved by hitting **AMIGA-L**.
- simply press the **L** key to make the file requester appear.

To erase a file from the list, just click on the gadget **remove**, or press the **R** key. This only functions when you have highlighted one of the items in the list. With **remove**, you are only deleting the file from the list, and not physically removing it from your disk.

By selecting the **info** gadget, you can obtain helpful information about the highlighted file. Information displayed will include the complete file name, the length of the file, the date that it has last been modified, and the file attributes.

The gadget **clear list** has a similar function as **remove**. The difference is that by clicking **clear list**, all entries will be cleared from the list. Again, the files will not be physically removed from your disk.

A more dangerous function happens when you click on the gadget **delete**. When you select **delete**, a requester appears asking whether you really want to delete the file(s) you have selected. If you confirm your choice, the file(s) and their icon(s) will physically be removed from your drive. Use caution when using this function.

Should you like a picture so well that you want to copy it, use the gadget **copy**. The program **Copy** must be resident in the **C:** directory for this function to work properly.

The gadget **about** will bring up a window on your screen that will display the author of the program, the creation date, and the version number. This is interesting in case you want to inform us of an error or if you want to know if you have the latest version.

The gadget **quit** will allow you to exit the program.

Instead of clicking to select a file, and then selecting the gadget **show** to display or view the file, you can also double click on the file name that will have the same effect as **show**.

If you do not like using your mouse, you can also access all the functions through your keyboard. To do this, you will be using the up and down arrow keys, as well as the return key. With your up and down arrow keys you can move your selection bar to the desired entry, and press the return to activate your choice.

10.4.2. The Configuration

A configuration file is required for IntuiView to recognize specific file types, and determine which program to use to view them. This configuration file contains the information that IntuiView requires to display properly your files. This file is usually found in ENVARC:Intuiview.prefs, and can be edited with any text editor you wish.

An example of such a configuration file follows, which will help you understand the explanations on the next pages.

```
IFF: // used for comments
#?,FORM????ILBM#?,bin:viewiff %s C
#?,FORM????8SVX#?,bin:sound %s

GIF: // there are two formats
#?,GIF87a#?,bin:ViewGIF.dcc %s C
#?,GIF89a#?,bin:ViewGIF %s C

JPEG: // the format is
#?,??????JFIF#?,bin:viewJPEG %s

MPEG: // the format are
#?.mpg,#?,bin:playMPEG %s
#?.mpeg,#?,bin:playMPEG %s
#?.play,#?,bin:play %s

TeX:  // the format is
#?.dvi,#?,tex:bin/ShowDVI %s

Archive: // also for .lha files
#?.(lha|lzh),#?,bin:lha >t:temp v %s,bin:most t:temp
#?.zoo,#?,progs:bin/zoo >t:temp v %s,bin:most t:temp

Exec:    // programs can also be started
#?,??'03'F3#?,run %s
```



```
Default: // if nothing else it may be text
#?,#?,bin:most %s
```

Lines with fewer than two commas, and empty lines, are considered comment lines. You can put as many explanations as you wish into this file, as long as your lines do not contain more than one comma. Lines with two or more commas can be split up into three categories:

```
<file-pattern>,<byte-pattern>,<action>,<action>,...
```

The file pattern is the easiest to understand. Most files can be recognized by their file extension. For instance, lharc files have the extension .lha or .lzh. When TEX translates a file, it will have an extension of .dvi. For any file that has a certain extension, we will use the abbreviation #? followed by the extension. An example would be #?.lzh, which would indicate any files ending in .lzh should be considered. If you look at our example configuration file above, in the case of all files ending in .dvi, we are telling IntuiView to display those files using the program ShowDVI, which can be found in the subdirectory TeX:bin. The extension %s will automatically place the file name behind the command, ShowDVI.

Should you want to view a file called Hello.dvi, then this .dvi extension will be recognized and the line

```
TeX:bin/ShowDVI %s
```

will be interpreted as

```
TeX:bin/ShowDVI Hallo.dvi
```

Sometimes a file will not be recognized. This could be because some of your pictures might not have the proper extension to the file name. For instance, IFF pictures usually have an extension .pic, however, sometimes they have an extension .iff that could cause confusion in your configuration file. In such instances you can use the byte pattern approach. In other words, we are going to tell the configuration file to look at the first 80 bytes in your file to determine what program to start.

For IFF-ILBM files, the letters FORM and ILBM will always be part of those 80 bytes. In between those two word sequences, you will find four bytes which for us are of no use and are different from file to file. We can proceed with the following example:

```
#?,FORM????ILBM#?,BIN:viewiff %s C
```

his line will certainly recognize an IFF-ILBM file, whatever the extension ay be. Since the extension does not matter (in this instance), you do not eed to indicate the extension before the first comma. What action has to e executed can be found after the second comma. In this case, the program iewIFF is being started, and the name of the file is automatically being nserted behind ViewIFF.

s you can see from the example called **Exec:**, you can also include byte alues instead of characters. A good example would be:

```
#?,??'03'F3#?,run %s
```

This line means that extensions to the file names can be ignored, however, he first two bytes in the file are also obsolete and the next two bytes should ave the values $3 and $F3. In other words, you are indicating to IntuiView hat all executable programs will have these two bytes at the beginning of he file. **IntuiView** can consequently be used to start your programs by either double-clicking or hitting **return** whenever you have selected a file o run.

10.4.3. ToolTypes

In case you do not want the configuration to be placed in the directory **ENVARC:**, you can modify this. In your icon **IntuiView**, there is a Tool Type with the name **CLASSFILE**. A possible entry to this file would be:

```
CLASSFILE=S:My_Definition
```

You can also create two different icons, one for viewing files, and the other for modifying the files to the required program. Just let your imagination go wild.

A second Tool Type has the name

```
WINDOWTITLE=My_Display_Screen
```

With this Tool Type you can use your own window titles so that you can differentiate between versions of the **IntuiView** configuration files.

10.4.4. Opening Your Shell

You can also open **IntuiView** from your Shell, you do not need to use the Workbench. If you start IntuiView with ? as a parameter, you will see the following line displayed:


```
FILES/M,C=CLASSFILE/K,T=WINDOWTITLE/K:
```

Just like in Tool Types, you can enter a name of a configuration file after **CLASSFILE**. If you do not enter a file name, the program will default to **ENVARC:IntuiView.prefs**. After **WINDOWTITLE** you can enter any text that will be displayed in your window as a title. It will be displayed as text at the top of your window.

The parameter **FILES** will allow you to indicate certain *patterns* or file extensions. These will be responsible for displaying, in your list, all those files that end in the extension you have entered after the parameter files.

for example:

```
IntuiView #?.c readme CLASSFILE s:Classes T=Test
```

This line would make all files that end in **.c**, as well as **readme**, appear in the list. The configuration that will be chosen in this case would be found in S: and be named **Classes**. Your window would have the title, **Test**.

10.5. Picture Viewing Programs

Since we know that everyone does not own programs such as **Art Department Professional** or **ImageFX**, we have provided several programs that allow you to view pictures. These pictures can be in the following formats, IFF-ILBM, GIF, or JPEG format.

You need to determine which program to use to display the various formats. The programs responsible for displaying the pictures are called **viewers**. Viewers can be started from either the Shell or Workbench, and will recognize several parameters that are similar among image viewing programs.

It is easier, however, to use the program **IntuiView**. IntuiView will automatically determine the image format and then use the correct program to display it (see also 40). As previously mentioned, IntuiView can be run from either Workbench or the Shell.

10.5.1. ViewIFF

The most prominent picture format on the Amiga is the IFF-ILBM format. Most programs are capable of displaying and saving **IFF pictures**.

The viewer called **ViewIFF** will read all formats currently being used with the exception of HAM pictures. This includes 8 bit and 24 bit plane pictures. Should you have pictures with different bit planes, you should first convert them. A program in the Public Domain called PPM Tools is a good example of a conversion program.

ViewIFF has a few parameters that can be listed by using ? as a parameter:

```
FILENAME/M/A,C=CENTER/S,W=WAIT/K/N,R=RESOLUTION/K/N,
B=BEHIND/S,I=INFOONLY/S
```

After **FILENAME/M/A**, you can enter as many file names as you want. For instance, you can enter **#?.iff**, which would make **ViewIFF** display all the files that end in .iff.

The option **CENTER**, if indicated, will center images on your screen that are smaller than your current display. This can be abbreviated with a **C**.

Usually **ViewIFF** will wait for a mouse click before displaying the next picture. However, if you would like a continuous slide show, you can use the parameter **WAIT**. By specifying the amount of time, W=XX, **ViewIFF** will display a pictures every XX seconds.

Some pictures are bigger than the largest resolution you can display on your monitor. **ViewIFF** always tries to open a screen that is as large as your horizontal and vertical picture resolution. For example, if a picture is 600x800 pixels, **ViewIFF** will try to open a 1120x832 screen. If this resolution is not supported by your monitor, **ViewIFF** will return an error message, and will not display the picture. **ViewIFF** can be forced to display a certain resolution no matter what your picture size is. To do this, you must use the parameter **RESOLUTION**, and specify the *width* of the pixel resolution you want to use:

```
ViewIFF Picture.iff RESOLUTION=1024 CENTER
```

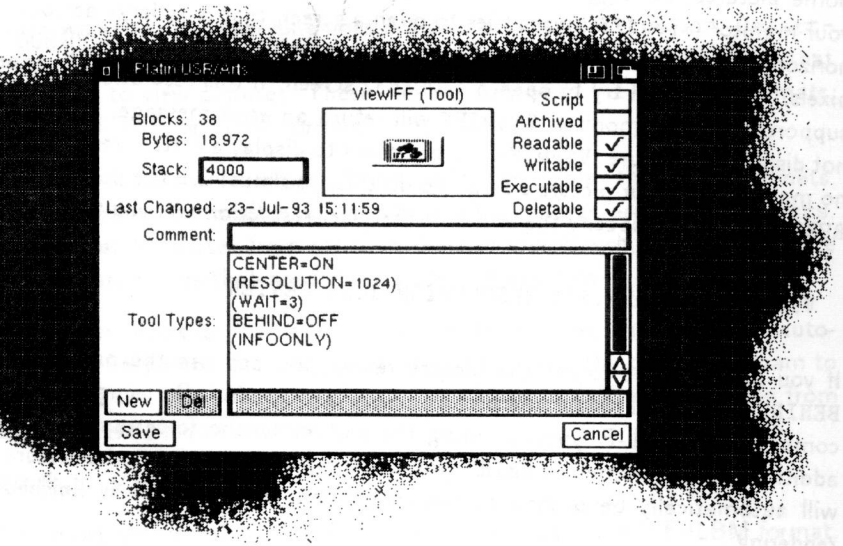
If you do not want to see the picture render you can use the parameter **BEHIND**, which will render the picture in the background. By using the key combination CTRL-F, you can bring the picture to the foreground. If, in addition to the parameter **BEHIND**, you use the parameter **WAIT**, the picture will automatically be pushed to the foreground as soon as it is finished rendering.

INFOONLY will display information about the specified picture. If you are familiar with the IFF format, this information may tell you why a picture cannot be displayed.

Starting the ViewIFF program from the Workbench is very simple. Double click on the icon and a file requester will appear in which you can choose the files you want to display. Once your picture has been displayed, simply click the mouse button and you will be returned to the file requester. If you want to exit ViewIFF, click on the **Cancel** button.

If you want to show multiple pictures without loading each picture separately, you can use the extended select method. Single click on **ViewIFF** and then, while holding the shift key, single click on all the pictures you want to display. By double clicking on the last picture, **ViewIFF** will display all the selected pictures continuously.

In the Shell environment, you are able to add parameters to **ViewIFF** to change resolution and to center the picture. This is also possible from Workbench by selecting the **ViewIFF** icon and pressing **AMIGA-I**. This looks like this:



After single clicking on ViewIFF and pressing AMIGA-i

After clicking on **ViewIFF** and the key combination **AMIGA-I**

The list in the Tool Types box has entries that resemble the parameters that could be entered in the Shell. For instance, the entry **CENTER=ON** means the same thing as **CENTER** in the Shell. Your pictures will be centered in the middle of the screen if the resolution of the picture is smaller than the screen resolution. Should any of the entries be in parenthesis, they will not be active, this is simply a reminder of what options are available. Values can be entered after the parameters **RESOLUTION** and **WAIT**, which will have the same effect as in the Shell environment described above.

An advantage we have under the Workbench environment is that the Tool Types can be added to the picture icon. This means **ViewIFF** will be forced to display the picture according to the Tool Types of the picture's icon.

Unfortunately, the picture data is stored in a different format when IFF images are rendered in 256 or more colors. Each single bit has to be converted, which means that pictures with a resolution of 800x600 in 256 colors, 1.8 million bits would have to be converted. With a resolution of 640x480 in

16 million colors, about 7.3 million bits would have to be converted. Since this takes time, a faster CPU will speed this process up.

10.5.2. ViewGIF

Besides IFF pictures, you will frequently encounter GIF pictures, especially on mainframes at universities. These pictures can be viewed with the program **ViewGIF**. The parameter set used is the same as that used with ViewIFF, with the exception of **INFOONLY**. **INFOONLY** has been omitted since only minimum information is provided with the GIF file, and this information is always displayed on the screen. If you do not want this information displayed, you can enter the following:

```
ViewGIF >NIL: test.gif
```

Just like **ViewIFF**, you can start **ViewGIF** from the Workbench or Shell. Through Workbench, the parameters will be entered in the Tool Types, the same as described for ViewIFF (see section 10.6.1).

Since GIF pictures are compressed, it will take longer before the picture is completely rendered.

10.5.3. ViewJPEG

Hi-Resolution, high color pictures usually require much space on your hard drive or disks. To avoid this, recent developments have come up with a format call JPEG, which very efficiently compresses pictures. JPEG is very useful for TrueColor pictures, however, when using 8 bit pictures, you may wish to stay with the GIF format.

The parameters for **ViewJPEG** are:

```
FILE/M,8BIT/S,15BIT/S,16BIT/S,24BIT/S,VERBOSE/S
```

Just like the previous-viewers, you can add as many files as you wish behind the **FILE/M**.

The quality of the display is determined by the parameters **8BIT**, **15BIT**, **16BIT**, and **24BIT**. By using these parameters you can indicate the color depth of the pictures. **8BIT** indicates 256 colors, **15BIT** indicates 32768 colors, **16BIT** indicates 65536 colors, and **24BIT** indicates 16 million colors. If you do not specify any of these parameters, **ViewJPEG** will automatically use **15BIT**.

Verbose will display information about the size and color of the picture being displayed.

CTRL-C, q, Q, and ESC will interrupt the display at anytime. Clicking on the mouse button or hitting n will display the next picture if you have chosen more than one picture to be displayed.

Just like the other viewer programs, you can enter the Tool Types or Shell parameters in the Icon of the program. If you do not indicate a file name when you start ViewJPEG, a file requester will appear in which you can choose the files to be displayed. When running under Workbench 3.x, you can choose more than one file at a time.

The same is true for ViewJPEG as with the rest of the viewers, file compression requires time and a faster CPU will increase the speed of the picture rendering.

10.6. Animations

An animation sequence played from a hard drive has always been fascinating. To save hard drive space, a widely used format has been developed: MPEG. The compression quality is very high: a 1.7 MByte MPEG file has a decompressed size of 28 MBytes!

Compressing procedures are so calculation and time intensive that software does not currently exist to do this efficiently, it requires special hardware. In principle this is also true for the decompression, however, for small film sequences there is software that will provide decent results. We have included two programs, one to decompress MPEG files that are in the RAW format (not compressed), and another to play these RAW files.

10.6.1. PlayMPEG

The program **PlayMPEG** is used to unpack and play an animation on the screen. Various options, all of which will not be described here, can be tried out. Again, if you enter the parameter ?, a list of all possible parameters will be displayed.

By entering

```
PlayMPEG example.mpeg
```

you will be able to display an MPEG file on your screen. This is still, however, being rastered to display 256 colors. To view an MPEG file in TrueColor, you will have to add the option, **-dither color**,

```
PlayMPEG -dither color example.mpeg
```

As was previously explained, the speed of this is not very fast, which has nothing to do with the programming. If you want to save the picture in RAW format to a file, the option **-save file** can be used, for instance:

```
PlayMPEG -dither color -save example.play beispiel.mpeg
```

However, an uncompressed file can easily use 15 times more space than a compressed one, and the decompression routine can take quite awhile.

10.6.2. Play

For the correct speed of animation playback we have included the program Play. This program is capable of displaying animations with a resolution of 160x120 pixels at 25 frames per second. However, you need a hard drive with at least a 500 KByte per second transfer rate.

Play recognizes three parameters

```
Play FILE/A,LOOP/S,SYNC/S
```

For the option **FILE** you should indicate an animation filename. This file must have been created with **PlayMPEG** using the option **-save**. **Play** can only display such files.

If you want the animation to play continuously from the beginning, use the option **LOOP**.

If the animations do not play as smoothly as you would like, you may want to use the option **SYNC**. **SYNC** will optimize the speed of your hard drive and prioritize the transfer from the hard drive to video memory.

10.7. IntuiSpeed

We have included a program that will allow you to measure the speed at which the Picasso II will function. The program we have included is called IntuiSpeed. On top of the screen you will have a text line indicating the computer type, the speed at which it operates, the amount of memory your machine possesses, and the type of graphics card you are using.

Each test will run for exactly 10 seconds. During those 10 seconds we are measuring how often the selected action can be executed. The values resulting from the 10 second test will be displayed in the box behind the gadget.

You can save the screen of results by clicking on the **save** gadget. A requester will appear asking for the file name you want to save under. You can also print the results by clicking on the **print** gadget. If you want to do all the tests, one after each other, proceed by clicking on the **start all** gadget.

After changing either the number of colors, or the screen mode, all previous values will be reset to zero. This happens to guarantee consistent results.

Partial source code can be found on your installation disk. We did not include the entire source code to inhibit the release of optimized versions of the program, which may lead to inaccurate test results and speed comparisons. Any suggestions you may have concerning this program would be appreciated, please contact us directly.

11. Drivers for Other Programs

Programs that are capable of displaying higher resolutions and greater numbers of colors, but have been limited so far by the Amiga's internal graphics, are available for different graphics cards on the market.

Some of these programs work under a module concept, which means that without modifying the program, a support module can be written to support new graphics cards. Of course, we are trying to support as many of these programs as possible.

Programs without this module concept, or that do not allow one to choose a new ScreenMode, present a compatibility problem for the Picasso II and as well as other graphics boards. The only way to make these programs compatible is to contact the manufacturer and convince them to develop a version that will be compatible with the new Commodore Amiga programming conventions and the Picasso II. Some companies may choose to take advantage of the special features of the Picasso II directly. An example of a company who has made a commitment to creating a specialized version of their software for the Picasso II is TecSoft, the creators of TVPaint, who are working on a Picasso II specific version of the program.

There are programs, some very popular programs, that do not follow the Amiga programming guidelines at all, or choose to use the Amiga custom chips directly. While this was an accepted practice in the past, with the release of the new chipsets and even more advanced third party graphics boards, it is no longer a feasible option. An example of a program that uses the hardware directly is Deluxe Paint (all current versions). Electronic Arts is aware of the problem and is currently developing a version of Deluxe Paint that will use the operating system and will support the Picasso II.

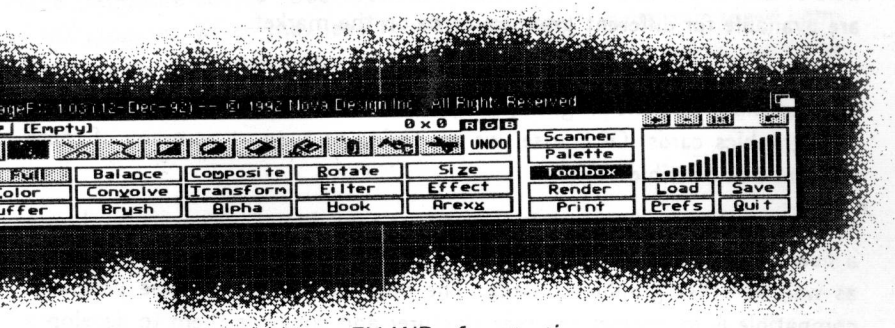
Other programs, like Maxon Paint, which runs within the Workbench environment, will not have any compatibility problems.

The functionality of programs that support the module concept varies from program to program. The next section describes some of the various implementations.

11.1. ImageFX Viewer Module

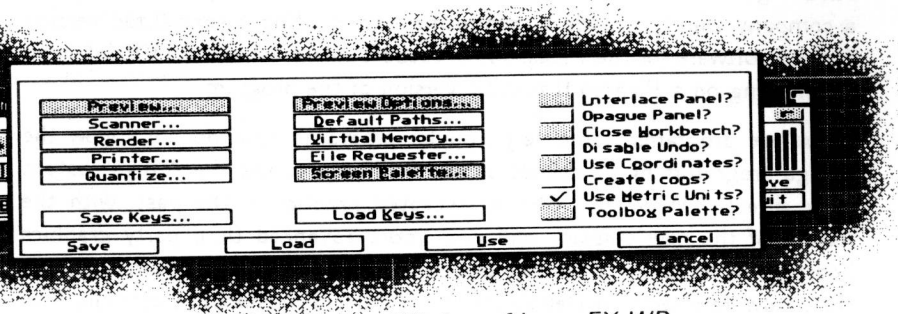
The Picasso viewer module will be installed in the path .../ImageFX/Modules/Render. The Installer will determine the full path for your machine, such as Work:ImageFX/Modules/Render.

You can choose the Picasso viewer module from ImageFX by choosing the Workbench version ImageFX_WB, and then clicking on the gadget Prefs.



ImageFX_WB after starting

A new window will appear with the title PREFERENCES.

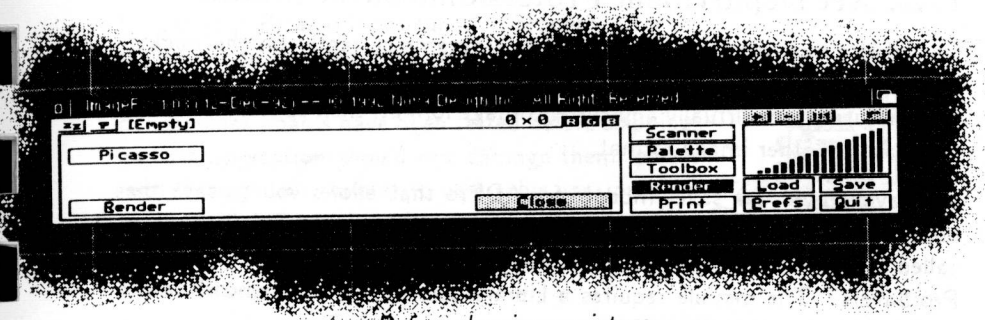


The Preference Window of ImageFX_WB

In the left of the PREFERENCE window there will be a gadget called Render. By clicking on this gadget a file requester will appear in which you can choose your render module. You should be able to identify the file called Picasso, double click on Picasso, and then close the PREFERENCE window by clicking on Save and Okay when the requester appears.

From now on, all pictures will be displayed through the Picasso II. All you need to do is click on the gadget LOAD to select the picture you want to view.

Now click on the gadget RENDER and a requester should appear which looks like:



Before showing a picture ...

By clicking on Render, the preloaded picture will be displayed on your screen. The Picasso II will always try to use 24-Bit mode (TrueColor). If you only have 1 MByte of ram on the Picasso II, you will be restricted to a resolution of 640x480 pixels. Pictures that are smaller will be centered. If the picture is larger, only a part of the picture will be displayed. If this is the case, you can use the arrows to scroll the picture around the screen. To scroll the picture by 10 pixels at a time, use the arrow and shift keys at the same time. Besides this key combination, you can use CTRL-arrow key to scroll sideways.

If you wish to move the picture into the background, use either ESC, RETURN, or q. This will not close the screen. Also, pressing one of the mouse buttons will have the same effect.

The three gadgets in the view module invoke the following functions:

`Picasso` By clicking on this gadget you can load a new Render module.

`Render` Your loaded picture will be displayed in the TrueColor mode. If no picture is loaded, the error message "No buffer to work on" will appear. You can also use the keyboard short cut R.

`Close` You can only use this gadget once you have loaded a picture. After clicking on CLOSE, or using the keyboard short cut C, the screen and picture will be closed.

11.2. Art Department Professional Saver Module

One of the best display and conversion programs currently on the market is Art Department Professional (also known as ADPro) from ASDG. It can load pictures from virtually any current image format, then convert and save them as any other image format.

We have provided a saver module for ADPro that allows you to save the picture directly to the Picasso II instead of the hard drive. This module is called Picasso and the Installer automatically copies it into the path ADPro:Savers2. The module requires a Library called vtr.library, which the Installer copies in the LIBS: directory.

Using this module is very simple, all you have to do is choose Saver PICASSO and click on Save. To return to the ADPro interface while viewing an image, click the left mouse button.

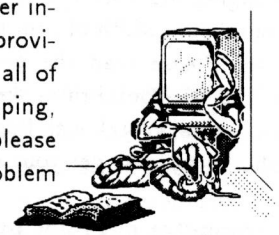
11.3. Real3D Viewer Library

We are also currently supplying a driver for Real3D. This driver is still currently under development and testing. Information on the functioning of this module will be provided when the development is finished.

Make sure to read the ReadMe file on the installation disk, it will give details of the library when it becomes available.

12. Trouble Shooting

Should your Picasso II not function properly after installation, do not panic, this chapter has been provided to solve any problems that may arise. Since all of our graphics boards have been tested before shipping, and transportation should not damage them, please read through this chapter to help solve your problem before calling for technical support.



Blank Screen

First, make sure that your monitor is turned on and that the cables between the computer, board and monitor are correct. Compare them with the drawing on page 9

The Picasso II does not produce a picture

After booting (or switching on your computer) you will get a normal Amiga screen. However, when you switch to a Picasso II screen, the monitor remains black. This usually indicates that the Picasso II monitor cable and the Amiga monitor cable are plugged into the wrong (opposite) connectors. Turn off your Amiga, exchange the cables, and you should then get a proper display.

It is also possible that your monitor is not capable of displaying the chosen resolution because the vertical refresh from the Picasso II is too low, or even too high. This can be changed in the icon for village.library, which can be found in the directory Sys:Expansion.

By changing the Tool Type MONITOR in the Village.Library icon to reflect the horizontal bandwidth of your monitor, you should be able to get your monitor working. Please read the section on specifications to find out what the supported horizontal rates are. If you get lost, you can boot your computer from a normal Workbench floppy disk. If all the cables are hooked up properly you should at least get your normal Amiga Workbench screen.

The screen is Flickering, Rolling, or Tearing

It is possible that you have not configured your software correctly, or your monitor is not capable of displaying the requested resolution since the vertical frequency is either too high or too low. You can modify this in the icon for village.library, which can be found in the directory Sys:Expansion.

By changing the Tool Type MONITOR in the Village.Library icon to reflect the horizontal bandwidth of your monitor, you should be able to get your monitor working. Please read the section on specifications to find out what the supported horizontal rates are. If you get lost, you can boot your computer from a normal Workbench floppy disk. If all the cables are hooked up properly you should at least get your normal Amiga Workbench screen.

The computer does not boot

Either the computer does not have any power, the board is not installed correctly, or you have a hardware conflict. First make sure you have power connected properly to your computer. Should this be the case, you will have to open the computer and check to see if the Picasso II is aligned correctly in the Zorro slot and seated properly. Sometimes, simply reseating the card will solve this problem.

No Picasso II resolutions within the ScreenMode Requester

There could be several reasons for this. First, check to see if the village.library is in the directory Sys:Expansion and that Binddrivers is being run from S:Startup-Sequence. Also check to see if the Picasso monitor file is in the DEVS:Monitors drawer (Workbench 2.1 or higher) or in the Sys:WBStartup (Workbench 2.0).

Should all of this be correct, then your Picasso II is not being initialized when your system starts. This could either mean that the Picasso II board is defective, or there is a hardware conflict. To verify this, you should run the program ShowConfig (provided with all versions of Workbench newer than 2.04) and compare it with the following:

```
BOARDS:

Board (...): Prod=2167/11($877/$B) (@$200000 2meg)

Board (...): Prod=2167/12($877/$C) (@$E90000 64K)
```

The important part of this information is the product ID numbers which should have the values 2167/11 and 2167/12. If both of these are not present, the card is not functioning correctly. Instead of 2167/12, the value 2167/13 can appear when operating in a segmented environment.

If these entries are missing, you will have to open the computer to check to see if the board is installed correctly.

After verifying that the above is correct, and the Picasso II is still not functioning, please try the board in another machine if possible. This may help us determine the problem with the board.

The screen appears fragmented or parts of the screen are missing.

If suddenly you realize that some parts of your screen are not being redrawn, you are probably running the program CPUBlit or another public domain screen enhancement. Using CPUBlit or other screen enhancements after starting the Picasso monitor file can cause these types of problems.

Help: Remove CPUBlit and all other public domain screen or system enhancement programs from your system and make sure that it is not being automatically started. Since all drawing functions are being taken over by the Picasso II's blitter, these types of programs are no longer required for fast operations.

Error Message: Could not open village.library

The Picasso II may be incorrectly installed or it is not functioning. When the Village.Library is opened, it looks for the Picasso II hardware. If the Village.Library cannot locate the Picasso II board, the library will not be started and this message will appear. To check to see if the Picasso II has been installed correctly, you can use the program ShowConfig, which is included with Workbench 2.0 and above. ShowConfig should show the following lines:

```
BOARDS:

Board (...): Prod=2167/11($877/$B) (@$200000 2meg)

Board (...): Prod=2167/12($877/$C) (@$E90000 64K)
```

The important part of this information is the product ID numbers which should have the values 2167/11 and 2167/12. If both of these are not present, the card is not functioning correctly. Instead of 2167/12, the value 2167/13 can appear when operating in a segmented environment.

Error Message: Could not open vilintuisup.library

This could happen because vilintuisup.library, which should in the LIBS: directory, is missing. It could also be that the vilintuisup.library could not be opened, or the Picasso II's memory is segmented. Older versions of vilintuisup.library are not capable of working with segmented memory.

Should this be the case, please refer to page 92 on how to reverse the segmented installation.

13. Programming the Picasso II

The Picasso II is capable of much more than what is available normally through the intuition and graphics libraries. With the Picasso II you can access screen modes of 32768, 65536, or 16,777,216 different colors, at many different resolutions.

These screens are organized in memory in many different ways. Since the operating system does not support these types of screens yet, it is not possible to use standard intuition drawing techniques on these displays.

For you to program in these specific modes, we have included a few example programs, programming libraries, and documentation.

The following examples will hopefully give you enough information so that you will be able to program the Picasso II. In the following chapters you will find the Autodocs for vilintuisup.library, with examples. As an example of how to write your own programs, we refer you to the example programs 8BitLineDemo.c, 8BitBliterDemo.c, and 8BitFillDemo.c.

13.1. Introduction

The Picasso II combines the functions of a framebuffer with those of a programmable, intelligent graphics coprocessor card. Normally a framebuffer is pure video memory into which the CPU cannot directly read or write information. Usually the memory in a framebuffer is accessed through a small area of memory that is shared between the framebuffer and the host computer. Some framebuffers have memory that is both directly readable and writable, but these are rare. A graphics coprocessor card will usually have a small amount of memory that is shared with the host computer. The drawing functions are accessed through either programmable registers, or by loading instructions into special locations on the card. The main memory of the graphics coprocessor card is usually not available to the host computer directly, so drawing functions are limited to those provided by the card.

A combination of the above features would result in a card that is programmable, capable of processing graphics instructions independently of the host computer. All the memory on the card would be accessible by the host computer directly, without having to go through a small shared memory port. The card should be capable of executing certain graphics primitives such as line drawing, area filling, area moves, and bit level logical operations, removing the work load from the host CPU. The Picasso II is such a card

(retargetable graphics card). It combines the direct memory access of the better framebuffer boards with the intelligent drawing functions of graphics coprocessor boards.

To give a programmer of the Picasso II all the freedom that is necessary, he/she can directly address the video memory and write to it with a command. Certain conventions do have to be maintained with this functionality. The memory is not always organized the same way, this neither makes it very easy nor very difficult. All you need is some programming information that you will obtain in the following chapters.

13.2. Memory Organization

The Picasso II can be operated in different modes. These are:

1. Planar (up to 16 colors)
2. Packed Pixel (256 colors, also called Chunky Pixel)
3. Two 16-Bit-Mode (32768 and 65536 colors)
4. TrueColor Mode (16777216 colors)

In each mode the video memory will be managed differently.

13.2.1. Planar

This is referring to the Amiga's standard way of organizing the video memory: separate bitplanes. The color information for one pixel is spread over a few bitplanes. From each plane, you take a bit, add them to each other into a byte, and take the resulting number to look for the red, green, and blue values in your color table. These are then transferred to the monitor, which will display the correct color.

The bitplane with the lowest number (=0) contains the lowest value bit of the color index. In this mode, the Picasso II is being run with the intuition driver active. This organization pattern corresponds exactly to that of the Amiga.

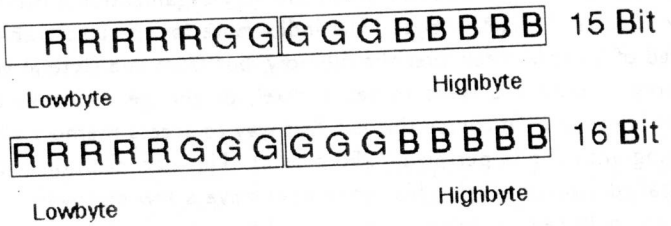
13.2.2. Packed Pixel

Planar mode has a big disadvantage. If you have 8 planes and you want to modify one Pixel, you will have to access the memory 8 times since the 8 requested bits are spread throughout the memory. Even the Amiga 4000 has this problem, which tends to slow the graphic operations down.

In the meantime, a new and different video memory organizational method has become popular: 1 Pixel, 1 Byte. The index Byte for the color table is not composed of bits scattered over the memory, but from one Byte in your screen memory. Should you want to set a Pixel, or change its color, one simple access is enough. Of course this is a fast way to access memory when you are dealing with 8 bits per pixel (256 colors). This organizational pattern is also called Packed Pixel. This mode does have a few disadvantages, however. If you only need 4 colors, you will still have to sacrifice one Byte per Pixel. For instance, in the Planar mode, which we described previously, 2 Bitplanes only take 1/4 of the memory required when compared to the Packed Pixel mode.

13.2.3. 16-Bit-Modi (also known as HiColor)

There are times when 256 colors may not be enough, therefore other modes have been developed. In PC's, they are known as HiColor modes. In these modes one Word (two Bytes) is used to display one Pixel. The values in the Word do not refer to the index of the color table, but instead contain the Red, Green, and Blue values. Each color is being assigned 5 bits, which gives a total of 15 bits (32,000 colors), leaving one extra bit. The 16th Bit is what will differentiate between the two 16 bit modes that the Picasso II supports. One mode will leave the 16th Bit unused, the other will use 6 Bits for the Green portion instead of 5 Bits. The organization of this Word is like this:



Since the Word is set up like that of a PC, the Byte containing the values for Red is set in the low address, the Byte for the Blue values is set in the high address. This is commonly referred to as BGR (as opposed to RGB).

13.2.4. TrueColor (24Bit)

TrueColor allocates one byte for each color component. In other words, you need 3 Bytes for 1 Pixel. The first byte of the lower address contains the values for red while the byte for the blue values is set in the high address (MSB). This is commonly referred to as BGR (as opposed to RGB).

The amount of data required for one pixel will increase considerably, however the processing is easier and the possible number of colors on screen increases to 16,777,216. This number of colors is more than the human eye can differentiate. In this mode the screen is said to have a photographic, or TrueColor, quality. Since three bytes per pixel are being used in this mode, very large resolutions are not possible. The total resolution available is a combination of the resolution of the image (width * height), and the depth of the image (how many bits per pixel). Using 24 bits per pixel to achieve the 16 million color mode requires a lot of memory.

13.3. Opening a Screen

With only a few simple programming instructions you will be able to access all the previously mentioned modes. The board must be fully initialized before you start your program. Once the board is initialized (this is the normal mode of the board, and if it is running at all, it is already initialized) you will be able to access the board through some simple commands. Your entry and exit points to the Picasso II will be through the following commands:

`OpenVillageScreen (...)` und

`CloseVillageScreen(...)`

These will allow you to open and close screens with the previously mentioned color depths and resolutions. The functions OpenVillageScreen() and CloseVillageScreen() are not required when the planar mode is being used. You can access the board when using planar modes through normal intuition programming conventions. In these modes (planar), screen modes are accessed by specifying the ViewModeID (if you are not familiar with this, please refer to the Amiga ROM Kernal Reference Manual:Libraries).

The function OpenVillageScreen() requires as a parameter a pointer and a structure of the type struct Dimensions, in which the wanted resolution and color depth have to be entered. For instance, the values 800, 600 and 8 are required for a screen of the resolution 800x600 in 256 colors ($= 2^8$).

Possible resolutions are

640 x 480
800 x 600
1024 x 768
1120 x 832
1152 x 900
1280 x 1024

Possible color depths are

8 (up to 1280 x 1024)
15 (up to 800 x 600 noninterlaced)
16 (up to 800 x 600 noninterlaced)
24 (only 640 x 480)

Theoretically it is possible to combine these freely, however some physical limitations are set (the maximum resolutions for HiColor modes may still change).

The OpenVillageScreen function will round up values that do not correspond to both tables. Should you request a screen with resolution values 600,700, 6, you will get a screen with a resolution of 1024 x 768 Pixels and 256 colors.

If the OpenVillageScreen() call is successful, you will get a pointer to the screen structure. If you then try to open a window on this screen, you will succeed but you will not see anything. The intuition driver is not allowing an intuition or graphics function to be drawn on such a screen as intuition itself does not understand these screen modes.

13.4. Drawing

Before you can start drawing into the screen you have to obtain the actual memory address of the screen. To get the screen's memory address, you must use the function: LockVillageScreen(ScreenPointer). This function will return the screen memory starting address of the screen being used. Only then can you start drawing directly into memory. You should finish all screen manipulations with the matching function: UnlockVillageScreen(ScreenPointer).

While you are using the function LockVillageScreen(ScreenPointer), you cannot switch between two Picasso II screens.

Here is an example

```
#include <Other System Things>
#include "vilintuisup.h"

void HandleVillageCard()
{
   struct Screen          *ScreenPointer;
   struct Dimensions      dm = { 0, 800, 600, 8 };
   UBYTE                  *Thescreenstartaddress;

   if (ScreenPointer = OpenVillageScreen(&dm))
   {
      Thescreenstartaddress = LockVillageScreen(ScreenPointer);

      // ... execute something in the screen memory

      UnLockVillageScreen(ScreenPointer);

      // ... wait for a user entry

      CloseVillageScreen(ScreenPointer);
   }
   else
   {
      PutStr("The desired screen cannot be open\n");
   }
}
```

You can insert your own routines where you see the //. Our example opens a Screen with a resolution of 800 by 600 in the packed pixel mode. The values 800, 600 and 8 are input into the structure dm:


```
struct Dimensions dm = { 0, 800, 600, 8 };
```

The rest should be fairly clear. The sequence of pixels on the screen is very simple, the function LockVillageScreen() returns the address of the start of the screen memory that is the address of the first pixel in memory. The pixel to the right of this has the next higher address (one byte in pixel mode, two bytes in any of the two HiColor modes, and three bytes in TrueColor mode). The pixel addresses run from left to right and the rows from top to bottom.

13.5. The Functions

In the function collection vilintuisup.c, you can see how to maneuver in your screen memory.

The first collection of routines defines how to set pixels in the different screen modes. Remember, when you are using planar modes you can use the standard intuition functions for drawing.

```
void SetTrueColorPixel(Screen, x, y, r, g, b);
void SetPackedPixel   (Screen, x, y, color);
void Set15BitPixel    (Screen, x, y, r, g, b);
void Set16BitPixel    (Screen, x, y, r, g, b);
```

Remember, while using these functions you have to surround them with the appropriate LockVillageScreen() and UnlockVillageScreen() function calls. This is for speed consideration and not for the functions themselves.

The next two routines are for line drawing. One to draw lines in the packed pixel mode called LinePacked(), and the other to draw lines in TrueColor mode, called LineTrueColor().

```
void LinePacked   (Screen,xstart,ystart,xend,yend,color);
void LineTrueColor(Screen,xstart,ystart,xend,yend,rgbcolor);
```

Both functions will execute the "Locking" functions automatically. You do not have to use LockVillageScreen() or UnLockVillageScreen() in this case.

In the packed pixel mode, we are working with color registers since one byte in the screen does not contain the red-green-blue values. Instead, it contains the number of an RGB entry in the color table, so we will need a function to set the colors in the table. This is the old function SetRGB4() which still functions under the packed pixel screens.

A more exact description of the functions and explanations of them can be retrieved from the chapter, vilintuisup.library, functions starting on page 73.

14. Programming in C

The chapter on how to program the Picasso II should have made you curious enough to make you want to program it yourself. There is one more task that must be performed before you can do this, you must copy the includes, libraries, and other files to the correct directories of your compiler environment.

For programmers who are not familiar with C, we unfortunately do not have any other translations at this time. Before you begin any programming you may want to call us to see if any new translations have been done.

The demo programs written in C have been tested with five of the most current compilers

- SAS C Version 5.1 und 6.x
- Aztec 5.2a
- DICE Version 2.06.39
- Maxon C/C++ Version 1.02
- GNU C/C++ Version 2.2.2

Only Maxon C/C++ (Version 1.02.5) had problems with the 8BitBlitter demo, which can be explained by a mistake in the Library Function memset().

The source codes to the Demo Programs, Includes, and Libraries can be found in the directory that you chose during the installation. If you chose the option Beginner or Novice during the installation, these files were not installed to your hard drive. You will need to refer either to the installation disk for these files, or perform a reinstall this time choosing Intermediate User.

For the compiled programs to function properly, you need the vilintuisup.library that should be located in the LIBS: directory.

Before starting to write your own programs you should read the chapter about vilintuisup.library function on 73. Also, you should read the the notes below the keyword BUGS if you do not want to waste hours trying to find a problem that may not be in your programming.

14.1. SAS C Version 5.1 and 6.x

The installation for SAS C is very straightforward. The compiler uses the #PRAGMA directive to set Library Opening Conventions, you do not even need a Link Library. Just enter the following line

```
smake -f makefile_SAS6
```

from the directory that contains your source code files. For Version 5.1 the line should look like this:

```
lmk -f makefile_SAS5
```


14.2. Aztec 5.2a

In Aztec C, just like SAS C, the installation is also very straightforward. To make the Demo source code into executable programs, you need to enter the following line from the source code directory:

```
make -f makefile_Aztec5
```

14.3. DICE - Registered Version 2.06.39

With DICE you have to copy the Libraries vilintuisup.lib and vilintuisupr.lib from the directory linklibs to DLIB:. Since the compiler does not know any directories, you have to enter the following line:

```
dmake -f makefile_DICE
```

Make sure that you are in the source code directory.

14.4. Maxon C/C++ Version 1.02

For Maxon C/C++ the installation is the same as SAS C and Aztec C. You do not have to do anything extra since Maxon C/C++ also has a special #PRAGMA that determines your library origins.

To translate the demo files you will need to get the public domain program Make, which can be found on Fred Fish Disk 523 (BMake including source). With this program you should enter the following line:

```
make -f makefile_MaxonC
```

Make sure to be in the source code directory.

14.5. GNU C/C++ Version 2.2.2 and higher

Just like DICE you have to copy a Library named libvilintuisup.a from the directory linklibs to the directory gcc:lib.

To translate the demo files you will need to get the public domain program Make, which can be found on Fred Fish Disk 523 (BMake including source). With this program you should enter the following line:

```
make -f makefile_gcc
```

Make sure to be in the source code directory.

15. vilintuisup.library-Funktionen

We have included a description of all the functions of the vilintuisup.library below. The first versions of these libraries do not work with the board running in segmented memory mode. If you have upgraded your system to 8 MBytes of Zorro II memory and are using the Picasso II in segmented mode, none of the libraries described below can be used. Also, any programs which open screens with more than 256 colors, including drivers and viewers, cannot use these library functions.

TABLE OF CONTENTS

15.1. CloseVillageScreen

```
NAME
     CloseVillageScreen - Close a special village screen

SYNOPSIS
     CloseVillageScreen (screen)
                         A0

     void CloseVillageScreen (struct Screen *screen);

FUNCTION
     This routine closes a screen, which was opened by
     OpenVillageScreen(). It is save to call the function
     with 0.

INPUTS
     a screen pointer, pointing to a screen structure.

RESULTS

EXAMPLE
     look at the example given by OpenVillageScreen().

NOTES

SEE ALSO
     OpenVillageScreen(), LockVillageScreen(),
     UnLockVillageScreen()

BUGS
```


15.2. GetMemSize

```
NAME
     GetMemSize - get the memory size of the graphic board

SYNOPSIS
     GetMemSize(screen)
     D0          A0

     ULONG GetMemSize(struct Screen *screen);

FUNCTION
     This routine returns the graphic board memory size in bytes.
     Please supply a NULL as screen pointer to get the board size.
     This is nesseary for future enhancements.

     A call to LockVillageScreen() is not nessecary.

INPUTS
     a screen pointer, currently NULL

RESULTS
     ULONG - size of board memory in bytes.

NOTES
     In the future a value <>0 for the screen pointer will return
     the size, the screen occupy in memory.

SEE ALSO

BUGS
```

15.3. IsVillageScreen

```
NAME
     IsVillageScreen - determine the type of a screen

SYNOPSIS
     IsVillageScreen(screen)
     D0                   A0

     BOOL IsVillageScreen(struct Screen *screen);

FUNCTION
     This routine returns TRUE, if the screen represented by the
     screen pointer s is a Village screen, FALSE otherwise.

     A call to LockVillageScreen() is not nessecary.

INPUTS
     a screen pointer, pointing to a screen structure.

RESULTS
     TRUE  - if the related screen is a Village screen
     FALSE - otherwise

NOTES

SEE ALSO

BUGS
```


15.4. LockVillageScreen

```
NAME
     LockVillageScreen - lock screen to prevent screen movements

SYNOPSIS
     address = LockVillageScreen(screen)
     D0                          A0

     APTR LockVillageScreen(struct Screen *screen);

FUNCTION
     This routine locks the screen. You should only draw into the
     screen if you locked it befor. It is save to call this
     function with screen pointer, which does not belong to a
     village screen.

     If the screen is a special village screen, LockVillageScreen()
     returns the address of the first byte in the screen bitmap.
     According to the color depth of the screen a pixel occupies 1,
     2 or 3 bytes. One line occupies the number of pixel per line
     multiplied with the number of byte per pixel.

        ByteInOneLine = BytePerPixel * Screen.Width

     For further documention see "basics.doc".

INPUTS
     struct Screen *screen

     a screen pointer, pointing to a screen structure.

RESULTS
     APTR address

     address of the first byte in the screen bitmap.

EXAMPLE
     look at the example given by OpenVillageScreen().

NOTES

SEE ALSO
     OpenVillageScreen(), CloseVillageScreen(),
     UnLockVillageScreen()

BUGS
```

15.5. OpenVillageScreen

```
NAME
     OpenVillageScreen - Open a special village screen

SYNOPSIS
     Screen = OpenVillageScreen(dimensions)
     D0                         A0

     struct Screen *OpenVillageScreen  (struct Dimensions *dm);

FUNCTION
     This routine opens a screen, which will be displayed on the
     graphic board. If the screen could be opened, the function
     returns the address of the screen structure. If the opening
     fails, OpenVillageScreen() returns 0.

     OpenVillageScreen() performs all checks and will open the
     screen only, if the monitor is capable to display the desired
     resolution and colors.

     If the desired resolution is NOT one of

        640 x 480
        800 x 600
       1024 x 768
       1120 x 832
       1152 x 900
       1280 x 1024

     or the color depth is NOT one of

          8
          15
          16
          24

     then OpenVillageScreen() will round up the resolution or color
     depth and after that perform the display checks.

INPUTS
     struct Dimensions *dm.

     You have to fill in

       dm->Width       one of 640, 800, 1024, 1120, 1152 or 1280
       dm->Height      one of 480, 600,  768,  832,  900 or 1024
       dm->Depth       one of  8, 15, 16 or 24

RESULTS
     struct Screen *screen  a pointer to a struct Screen structure
                            or 0L, if the screen could not be opened

EXAMPLE
```



```
     #include <system_includes>
     #include "vilintuisub.h"

     void HandleVillageCard()
     {
        struct Screen     *ScreenPointer;
        struct Dimensions dm = { 0, 800, 600, 8 };
        UBYTE             *FirstByteInScreen;

        if (ScreenPointer = OpenVillageScreen(&dm))
        {
           FirstByteInScreen = LockVillageScreen(ScreenPointer);

           // do some actions to show the user something nice

           UnLockVillageScreen(ScreenPointer);

           // ... wait for user input bevor closing the screen

           CloseVillageScreen(ScreenPointer);
        }
        else
        {  PutStr("Can't open the desired screen\n");
        }
     }

NOTES
     Do not decompile or disassemble this function. The internal
     function calls and algorithm will change in the future.

SEE ALSO
     CloseVillageScreen(), LockVillageScreen(),
     UnLockVillageScreen(), vilintuisup.h

BUGS
```

15.6. UnLockVillageScreen

```
NAME
     UnLockVillageScreen - unlock screen to allow screen movements

SYNOPSIS
     UnLockVillageScreen(screen)
     D0                  A0

     void UnLockVillageScreen(struct Screen *screen);

FUNCTION
     This routine unlocks a locked screen. After unlocking please
     don't draw into the screen bitmap. It is save to call this
     function with a screen pointer, which does not belong to a
     village screen.

INPUTS
     a screen pointer, pointing to a screen structure.

RESULTS
     -- (void)

EXAMPLE
     look at the example given by OpenVillageScreen().

NOTES

SEE ALSO
     OpenVillageScreen(), CloseVillageScreen(), LockVillageScreen()

BUGS
```


15.7. VillageBlitCopy

```
NAME
     VillageBlitCopy - Move memory using the graphic board blitter

SYNOPSIS
     VillageBlitCopy(screen,vilcopyrec)
     D0              A0     A1

     LONG VillageBlitCopy(struct Screen *s,
                          struct VilCopyRecord *rec);

FUNCTION
     This routine is complex. It uses the graphic board blitter.
     The blitter can manipulate, copy and fill memory regions
     located on the graphic board memory. In addition the source
     (x)or destination region can be located in the Amiga system
     memory (one region - not both).

     To do the action the blitter should do for you, you have to
     supply several information. This has to be done using a
     VilCopyRecord structure:

       struct VilCopyRecord {
           UBYTE    *SrcAdr;    // Source address start
           UBYTE    *DestAdr;   // Destination address start
           UWORD     SrcPitch;  // Width of source area in pixels
           UWORD     DestPitch; // Width of dest. area in pixels
           UWORD     Width;     // Rect width in pixels to deal with
           UWORD     Height;    // Rect height in pixels to deal with
           ULONG     ROP;       // Blit Raster Operation - see below
       };

     SrcAdr and DestAdr are normal Amiga memory addresses. The
     pitches specify the width of the area you copy from/to, e.g.
     if the rectangle you want to copy is located on a screen with
     a width of 1024 pixel, SrcPitch has to be 1024. If the screen
     you want to copy to has a width of 800 pixel, DestPitch has to
     be 800. Width and Height are the width and height of the
     rectangle you deal with.

     ROP can be:

       VIL_ZERO           // Clear all bits
       VIL_ONE            // Set all bits
       VIL_SRCCOPY        // Copy src to dest
       VIL_NOTSRCCOPY     // Copy inverted src to dest
       VIL_SRCAND         // logical AND src with dest
       VIL_NOTSRCAND      // logical AND inverted src with dest
       VIL_SRCPAINT       // logical OR src with dest
       VIL_MERGEPAINT     // logical OR inverted src with dest
       VIL_SRCINVERT      // logical EXOR src with dest
       VIL_NOTSRCINVERT   // logical EXNOR src with dest
       VIL_SRCERASE       // logical AND src with inverted dest
```

```
    VIL_NOTSRCERASE     // logical AND inv. src with inv. dest
    VIL_SRCORNOT        // logical OR src with inverted dest
    VIL_NOTSRCORNOT     // logical OR inverted src with inv. dest
    VIL_DSTINVERT       // invert Destination

    The routine tests the parameter block. If anything is not ok,
    it will do nothing and will return 0.

    Overlapping rectangles are handled correctly by
    VillageBlitCopy().

    With the call of "VillageBlitCopy()" you start a blit. You
    have to use WaitVillageBlit() do determine the end of the
    blit. This feature allows you to use the CPU while the blitter
    is working (see exmaple below).

    IMPORTANT: VillageBlitCopy() will immediatly return 0 if the
    screen you want to act on is not the "FirstScreen" of
    Intuition. In other words: You can use the blitter only if you
    can see the screen you want to blit to/from.

    You have to call LockVillageScreen()/UnLockVillageScreen() to
    use the blitter, VillageBlitCopy() does not.

INPUTS
    a screen pointer, pointing to a Screen structure.
    a pointer, pointing to a VilCopyRecord structure.

RESULTS
    0  - if something goes wrong
    -1 - if no blitter is available
    1  - if all was o.k.

EXAMPLE

    /* this example implements a falling rectangle like the    */
    /* one in 8BitBlitterDemo                                  */

    struct VilCopyRecord vilcopy =
    {  0,           /* Source address in memory              */
       0,           /* Destination address in memory         */
       1024,        /* Width of source display in pixels     */
       1024,        /* Width of destination display in pixels */
       256,         /* Width of rectangle box in pixels      */
       256,         /* Height of rectangle box in lines      */
       VIL_SRCCOPY  /* Copy all */
    };

    Forbid();               // screen must be in displaymem !!! */
    ScreenToFront(s);       // bring the screen to front        */
    memstart = LockVillageScreen(s); /* snap the memory address */
    Permit();               // allow other tasks to run         */
    WaitVillageBlit();      // wait for blitter actions to quit */
```



```
    /* create a nice background scenery */
    for (y=0; y < 256; y++)
    {  memset(memstart+(y*3*1024),y,1024*3); }

    /* set the source address, it is the first line */
    xpos = 200;
    vilcopy.SrcAdr = memstart+xpos;

    /* for each loop iteration calculate the DestAdr and blit */
    for (y=1; y < 121; y++ )
    {
       (UBYTE *)vilcopy.DestAdr = memstart + xpos + (ULONG)
                                  ((float)(y*y)*0.034534653f)
                                  * vilcopy.SrcPitch;
       VillageBlitCopy(s,&vilcopy);
       WaitVillageBlit();

       /* the old DestAdr is the new SrcAdr */
       (UBYTE *)vilcopy.SrcAdr  = (UBYTE *)vilcopy.DestAdr;
    }
    UnLockVillageScreen(s); // don't forget this

NOTES
    Maximum value for VilCopyRecord.Width is 2048 bytes (2048
    pixels in Chunky Pixel mode, 1024 pixels in HiColor mode, 682
    pixels in TrueColor mode). Minimum value is 2 bytes.

    Maximum value for VilCopyRecord.Height is 1024 pixels in all
    modes.

    Maximum values for VilCopyRecord.SrcPitch and
    VilCopyRecord.DestPitch are 4096.

SEE ALSO
    WaitVillageBlit(), LockVillageScreen(), UnLockVillageScreen()

BUGS
    Blits from display memory to system memory does not work in
    version 1.1 and 1.2.
```

15.8. VillageRectFill

```
NAME
     VillageRectFill - Fills a rectangle using the on board blitter

SYNOPSIS
     VillageRectFill(screen,vilfillrec)
     D0              A0      A1

     LONG VillageRectFill(struct Screen *s,
                          struct VilFillRecord *rec);

FUNCTION
     This routine works similiar to VillageBlitCopy. Instead of
     copying it fills a rectangle with a specified color using the
     blitter. This is 10 times faster than it can be done by the
     CPU.

     Please remember that this function will only work, if the
     Screen you wants to work on, is in front of all other screens
     and is a village screen (Use "IsVillageScreen()" to determin
     this).

     To fill the rectangle, you have to supply several information.
     This has to be done using a VilFillRecord structure:

       struct VilFillRecord {
           UBYTE    *DestAdr;   /* Destination address start */
           UWORD     DestPitch; /* Width of destination area */
           UWORD     Width;     /* Rectangle width to deal with */
           UWORD     Height;    /* Rectangle height to deal with */
            LONG     Color;     /* special, see below */
       };


     DestAdr is a normal Amiga memory address. The pitch specifies
     the width of the area the rectangle should be filled, e.g. if
     the rectangle you want to fill is located on a screen with a
     width of 1024 pixel, DestPitch has to be 1024. If the screen
     you want to fill a rectangle has a width of 800 pixel,
     DestPitch has to be 800. Width and Height are the width and
     height of the rectangle you deal with in pixel (not BYTE!).

     Color has different meanings, depending on the color depth of
     the screen. If in Chunky Pixel mode, Color must have a value
     between 0 ... 255. In both HiColor or TrueColor mode the value
     is a merged rgb-value. The lowest significant byte contains
     the blue value, the next byte the green value, the third the
     red value. The most significant byte should be zero. A value
     like 255 is a very intensive blue.

     The routine tests the parameter block. If anything is not ok,
     it will do nothing and will return 0.
```



```
     With the call of "VillageRectFill()" you start a fill. You
     have to use WaitVillageBlit() do determine the end of the
     blit. This feature allows you to use the CPU while the blitter
     is working (see exmaple below). "VillageRectFill()" starts
     with a WaitVillageBlit() before performing the fill operation.

     IMPORTANT: VillageRectFill() will immediatly return 0 if the
     screen you want to act on is not the "FirstScreen" of
     Intuition. In other words: You can use the blitter only if you
     can see the screen you want to blit to/from.

     You have to call LockVillageScreen()/UnLockVillageScreen() to
     use the blitter, VillageRectFill() does not.

INPUTS
     a screen pointer, pointing to a Screen structure.
     a pointer, pointing to a VilCopyRecord structure.

RESULTS
     0  - if something goes wrong
     -1 - if no blitter is available
     1  - if all was o.k.

EXAMPLE

     /* this example do random rectangles like the program */
     /* 8BitFillDemo.c */

     {  struct VilFillRecord vilfill =
        {  0,                  /* DestAdr, will be set later */
           1024,               /* Width of destination display */
           256,                /* Width of rectangle box */
           256,                /* Height of rectangle box */
           0                   /* Color of rectangle */
        };

        /* This is important for the blitter use */

        Forbid();              /* screen must be in displaymem !*/
        ScreenToFront(s);      /* bring screen to front */
        memstart = LockVillageScreen(s);
        Permit();

        /* stop on CTRL-C */

        while (CheckSignal(SIGBREAKF_CTRL_C) == 0)
        {  xpos   = rand() % s->Width;
           ypos   = rand() % s->Height;

           vilfill.DestAdr   = (APTR)(memstart +(xpos +
                                (ypos * s->Width)));
           vilfill.DestPitch = s->Width;
           vilfill.Width     = rand() % (s->Width-xpos);
```

```
        vilfill.Height    = rand() % (s->Height-ypos);
        vilfill.Color     = rand() % 256;

        VillageRectFill(s,&vilfill); // does it's own
                                     //WaitVillageBlit() inside
    }

    UnLockVillageScreen(s);
}
```

NOTES

Maximum value for VilFillRecord.Width is 2048 bytes (2048 pixels in Chunky Pixel mode, 1024 pixels in HiColor mode, 682 pixels in TrueColor mode). Minimum value is 2 bytes.

Maximum value for VilCopyRecord.Height is 1024 pixels in all modes.

Maximum value for VilFillRecord.DestPitch is 4096.

SEE ALSO

VillageBlitCopy(), WaitVillageBlit(), LockVillageScreen(), UnLockVillageScreen()

BUGS

The operation has a bug in both HiColor modi. If you really need the fill operation in HiColor, contact us.

15.9. WaitVillageBlit

NAME

WaitVillageBlit - wait for blitter to finish

SYNOPSIS

```
WaitVillageBlit()

void WaitVillageBlit(void)
```

FUNCTION

This routine waits until the blitter on the graphic board has finished its work.

This is a busy wait in the current implementation!

If your actions allow a concurrent processing of data, you can use the blitter and the CPU at the same time. First start a blit, second do actions with the CPU and third call WaitVillageBlit().

INPUTS

-- (void)

RESULTS

-- (void)

EXAMPLE

```
/* this demo source fills a 1024x768 wide screen with */
/* zeros and ones */

struct VilCopyRecord vilcopy =
{  0,          /* Source offset from beginning of displaymem */
   0,          /* Dest offset from beginning of displaymem   */
   1024,       /* Source pitch == width of display           */
   1024,       /* Destination pitch == width of display      */
   1024,       /* Width of rectangle box */
   768,        /* Height of rectangle box */
   VIL_ZERO    /* Zero all bytes */
};

/* first get the address of the first byte in the screen */

(UBYTE *)vilcopy.DestAdr = LockVillageScreen(s);

for(y=768;y > 0;y-=4)
{
   WaitVillageBlit();           // wait for blitter to finish
   VillageBlitCopy(s,&vilcopy);     // o.k. start Blitter
   (UBYTE *)vilcopy.DestAdr += 2050;// do now the calculation
   vilcopy.Width   -= 4;            // for the next loop
   vilcopy.Height  -= 4;            // while the blitter is...
   if (vilcopy.ROP == VIL_ZERO)     // ...working
```

```
        { vilcopy.ROP  = VIL_ONE; }
        else
        { vilcopy.ROP  = VIL_ZERO; }
    }

    /* all o.k., free the screen */

    UnLockVillageScreen(s);

NOTES
    Bltting will only work, if the screen you wish to blit on is
    the FirstScreen, that means the screen must be located in the
    graphic board memory.

SEE ALSO
    LockVillageScreen(), UnLockVillageScreen(), VillageBlitCopy()

BUGS
```


A. Technical Data

Output Display Signal	Analog Red, Green, Blue: max. 1V VSS H-Sync, V-Sync: TTL-Pegel 15pol. High-Density VGA-Connector (female)
Input Display Signal	Analog Red, Green, Blue: max. 1V VSS H-Sync, V-Sync: TTL-Pegel 15pol. High-Density VGA-Connector (female)
Monitor Connector	Two 15 pin VGA connectors
Video Memory	1MByte RAM
Bus Interface	Amiga-ZORRO-II
Color Choice	256 colors from 262144 in Workbench Mode 256 colors from 262144 in Chunky Pixel Mode 32768 colors together in HiColor Mode 65536 colors together in HiColor2 Mode 16777216 colors together in TrueColor Mode
Resolutions	See table below
Monitor	At least a multiscan monitor with a 38kHz vertical frequency (a NEC 2A is not enough!)

The different resolutions, the amount of colors, and the picture display frequency can be obtained from the following tables. The "i" stands for Interlace, the "-" for "impossible".

38 kHz-Monitors

Resolutions	Amount of Colors		
	up to 256 Colors	32k & 64k Colors	16 Mio. Colors
640 x 480	71 Hz / 37.27 kHz	71 Hz / 37.59 kHz	64 Hz / 34.02 kHz
800 x 600	60 Hz / 37.86 kHz	60 Hz / 37.89 kHz	—
1024 x 768	87i Hz / 35.49 kHz	—	—
1120 x 832	—	—	—
1152 x 900	—	—	—
1280 x 1024	—	—	—

50 kHz Monitors

Resolutions	Amount of Colors		
	up to 256 Colors	32k & 64k Colors	16 Mio. Colors
640 x 480	73 Hz / 38.32 kHz	71 Hz / 37.59 kHz	64 Hz / 34.02 kHz
800 x 600	72 Hz / 47.95 kHz	60 Hz / 37.89 kHz	—
1024 x 768	62 Hz / 49.97 kHz	—	—
1120 x 832	—	—	—
1152 x 900	—	—	—
1280 x 1024	87i Hz / 48.37 kHz	—	—

57 kHz Monitors

Resolutions	Amount of Colors		
	up to 256 Colors	32k & 64k Colors	16 Mio. Colors
640 x 480	73 Hz / 38.32 kHz	71 Hz / 37.59 kHz	64 Hz / 34.02 kHz
800 x 600	82 Hz / 54.61 kHz	60 Hz / 37.89 kHz	—
1024 x 768	70 Hz / 56.42 kHz	—	—
1120 x 832	65 Hz / 56.75 kHz	—	—
1152 x 900	60 Hz / 56.46 kHz	—	—
1280 x 1024	87i Hz / 48.37 kHz	—	—

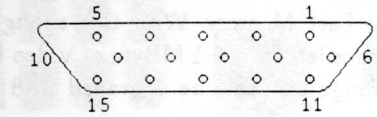
64 kHz-Monitors

Resolutions	Amount of Colors		
	up to 256 Colors	32k & 64k Colors	16 Mio. Colors
640 x 480	73 Hz / 38.32 kHz	71 Hz / 37.59 kHz	64 Hz / 34.02 kHz
800 x 600	82 Hz / 54.61 kHz	60 Hz / 37.89 kHz	—
1024 x 768	78 Hz / 63.19 kHz	—	—
1120 x 832	73 Hz / 63.31 kHz	—	—
1152 x 900	68 Hz / 63.62 kHz	—	—
1280 x 1024	87i Hz / 48.37 kHz	—	—

The values indicated above may vary slightly from those displayed in Screen-Mode, this may be due to rounding errors. The resolution 1280 x 1024 only exists with 16 colors.

B. Pin Outs

The VGA connector has the following pin outs:



1. Red
2. Green
3. Blue
4. - not used
5. - not used
6. Red Ground
7. Green Ground
8. Blue Ground
9. - not used
10. Sync Ground
11. - not used
12. - not used
13. H - Sync
14. V - Sync
15. - not used

C. Jumpers and Segmentation

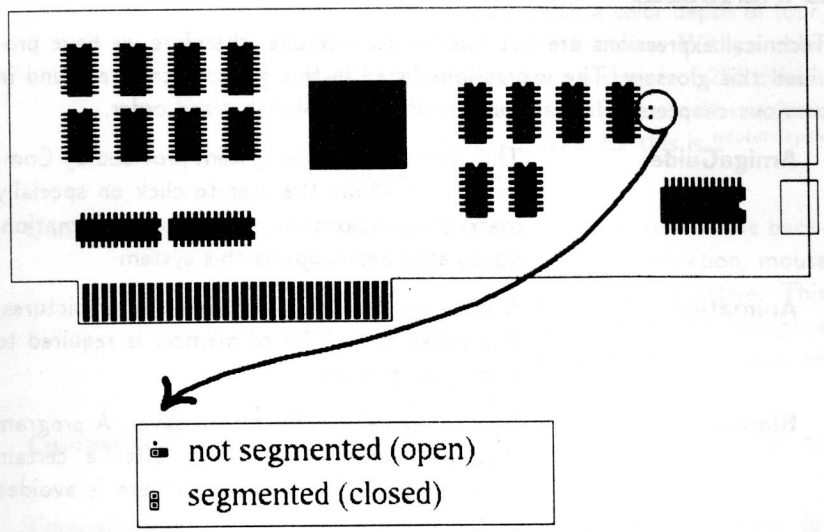
With the Amiga 2000 there is a possibility that the machine may have been upgraded to 8 MBytes of normal Fast Memory. With this configuration, there will not be enough address space left for the 1 MByte of Video Memory required by the Picasso II. The Amiga can only be upgraded to 8 MBytes of normal FastRAM.

Normal, in this context, means DRAM not installed on an accelerator board. With an accelerator board, the Amiga can be expanded to over a GByte without conflicting with the Picasso II.

If there is no address space left for 1 MByte of memory for the Picasso II, you need to use a different addressing method to access it. Instead of using a full 1 MByte, you could access it in small chunks through the I/O bus. For this reason, the video memory will be segmented into small parts. Should you want to write something on the screen, (for instance draw dots or fill areas), you will have to determine the correct segment to use, and then display it. This will slow down all graphics operations and make algorithms more complex for programmers.

The advantage in using this method is clear: you can use a Picasso II with fully expanded memory, although with limitation. The intuition driver functions properly, but programs written with vilintuisup.library will not work under segmented memory.

Factory settings for the board are for a non-segmented memory configuration. If you wish to operate the Picasso II in a segmented memory configuration, you will have to move a jumper.



The jumper is located on the right hand section of the board, next to four chips.

Make sure you ground yourself before handling the Picasso II board. This will avoid possible static electrical damage to the board.

Remove the jumper and make sure that you replace it over both pins, ensuring that it makes contact with both pins.

Refer to page 11 to open and close your computer to change this jumper.

D. Glossar

Technical expressions are not familiar to everyone, therefore we have provided this glossary. The expressions listed in this glossary can be found in previous chapters. They are presented here in alphabetized order.

AmigaGuide This refers to the help system provided by Commodore. It allows the user to click on specially marked expressions to find out more information. `ChangeScreen` supports this system.

Animation A word describing a fast sequence of pictures, also called film. A lot of memory is required to display animations.

Blanker Another expression for screen saver. A program that will darken your screen after a certain amount of time so that screen burn is avoided on the monitor.

Blitter The component of your computer or graphics card that specializes in the fast copying and manipulating of bits. The Amiga has such a blitter that can draw lines, copy rectangular areas, and process raw data. On the Picasso II there is a blitter that specializes in the copying of rectangles. The blitter on the Picasso II is also responsible for the fast scrolling speed of the board.

Boot Expression for the action of starting your computer. This happens automatically when you turn on your computer. You can also force this by pressing the key combination CTRL LEFT-AMIGA RIGHT-AMIGA.

Button Another word for gadget.

ChipMem This is the memory area which Amiga custom chips have access to. Amiga graphics use ChipMem to save the screen, for instance. With newer Amiga models this memory is limited to two MBytes.

CLI See Shell

Color Depth Color depth determines how many colors can be displayed on screen. With a color depth of four, you have a choice of 16 colors. With a color depth of eight, you have a choice of 256 colors. With a color depth of 24, you can choose from 16777216. The calculation for this is $2^{color\,depth}$ = number of colors.

Commodity A special kind of program that runs in the background and waits for a key combination, mouse movement, or timing to become active. This kind of program can be manipulated though a program called Exchange, which is included on your system disks.

Compatible Compatible means two or more items can be used in conjunction with each other.

Compression To save disk space, a technique has been developed to squeeze together data. Compressed files usually take one half or one third the space of the original file. The disadvantage is that the data has to be decompressed to be used.

CPU The Central Processing Unit of your computer. It is the chip that performs the most work in your computer. It calculates, moves, and modifies data, evaluates key strokes, and mouse movements. It is manufactured by a company called Motorola, which makes the 68000, 68020, 68030, and 68040 that are used in the Amiga family of computers.

Cycle-Gadget A special gadget that allows the user to select one of several options. One option is displayed at a time, and as the gadget is selected, the other options become visible. The displayed option is the selected option.

DPaint The best known paint program for the Amiga.

DRAM This is the cheapest form of RAM, which is used for memory. Unfortunately it is also the slowest.

flickerFixer A video card, by MicroWay, which will eliminate the interlaced flicker of the Amiga.

Floppy Another word for diskette.

Font Another expression for the style of text that is used on the screen or in a word processor.

Gadget An element in a window that you can click to execute specific actions.

Guru The Amiga will produce a Guru message if something is wrong in the system and cannot be executed correctly.

H-Sync A special pulse that the graphics card transmits to the monitor to display the end of a row.

Hardware Everything around your computer that you can touch, such as the hard drive, the keyboard, the mouse (the opposite would be software).

Horizontal Frequency The number of lines that the monitor can display on the screen per second. This is usually referred to in kHz (Kilohertz).

Icon A small picture that can be clicked with the mouse and moved around.

Installation For everything to function properly, the hardware and software must be installed correctly. The procedure of setting up the right parameters and placing everything in the right place is called the installation.

Interlace Interlace is the opposite of non-interlace. Interlaced expresses the way the screen is being redrawn; instead of every line being drawn, only every other line is being drawn. This means that for the entire screen to be redrawn, it takes two cycles instead of one. The advantage of this is that higher resolutions can be displayed, the disadvantage is the flicker will be more noticeable.

Intuition The part of the operating system that creates and manages Screens, Windows, Gadgets, and Icons.

Jumpers Small plastic squares that connect two pins. Jumpers are used to set certain hardware parameters. (see page 92

Library A collection of functions that can be used by several programs. On the Amiga, these files can be found in the LIBS: directory and will be loaded as needed into the main memory.

Listview A special gadget that is displayed in the form of a list. You can select a line from this list by clicking on it.

Multiscan A monitor that can display more than one frequency. Monitors of this type are capable of displaying different resolutions at different frequencies. The can also be called MultiSync, but this term is copyrighted by NEC Corporation.

Overscan Border areas of the video picture that can still be used by the Amiga graphics.

Packer A program that can compress files; see compression.

Picasso, Pablo A Spanish artist, sculptor, and graphic artist born in Malaga on October 25, 1881. In 1904 he moved to Paris. His pictures are known in the cubistic and classic style. His main art work was called Guernica.

Pixel Another expression for a dot on the screen.

Pixel Clock Video bandwidth.

Plane The Amiga memorizes the screen contents in planes. To do so, the screen information is memorized in layers. An example would be if you were laying transparent sheets on top of each other. If you have a sheet with yellow dots, one with blue, and one with red, you could display pictures in up to eight different colors.

Preference Another word for settings. The setting, or preference of your operating system, allows you to determine how a window looks, your text looks, the date, and/or which language to use.

RAM Random Access Memory. Also called fast main memory.

Requester Another word for window. A window that requests you to enter or choose something.

Resolution The number of dots on a screen in the horizontal and vertical direction. Resolution is usually expressed as how many pixels can be displayed in each direction. For instance: 800 x 600 Pixels.

Screen Another word for picture.

ScreenMode By ScreenMode we mean resolution. Examples would be NTSC:HighRes Interlace or PICASSO:800x600.

Scroll This describes the upward, downward, and sideways movement of a picture.

Segmentation Should you not have address space left for the one or two MBytes required for the Picasso II, you will want to use the method of segmentation. This method will allow the video memory to access small segments of the memory at certain times. Please refer to page 92 for more information.

Shell Window in which you can enter DOS commands and/or run programs that can display results in the same window. Shell is also referred to as the CLI.

Software All programs, drivers, and libraries. Contrary to hardware, you cannot touch or directly see software.

Tool Type If you click on an Icon and then use the key combination AMIGA -i, a window will appear

in which you will be able to set certain parameters. This is usually displayed in the form OPTION=<value>, and are also called Tool Types.

Transfer Rate The amount of bytes you can move or transfer per second.

TrueColor A way in which to express the possibility of displaying 16 million colors on your monitor. This are more colors than the human eye can distinguish. This is also sometimes called real color.

V-Sync V-Sync determines when the beam movement has to stop and jump back to the top of the screen on the vertical axis.

Vertical Refresh The number of times that your monitor can update in one second. The higher this value, the less your monitor will flicker. The human eye cannot distinguish flickering of non-interlaced displays above 70Hz.

VGA *V*irtual *G*raphic *A*dapter IBM has developed this name to refer to a specific type of graphic card that can be used with IBM compatible computers.

Video Has its origin from Latin and means "I SEE". This word is commonly used in connection with electronic pictures that are being displayed.

Video Band Width The amount of Bytes that have to be read per second to be transferred into a picture on the monitor. This is usually expressed in MHz (MegaHertz).

Video Slot A slot that has been specifically designed for video cards such as Genlocks. The Picasso II does not require a video slot.

Viewer A program to display pictures.

VRAM Memory that has been specifically designed for video, and is usually more expensive that

DRAM.

Workbench Usually referred to with a version number such as Workbench 2.0 of the Amiga operating system. The number following Workbench indicates the actual version of the Workbench, the higher the number, the newer the operating system. After version 3.0, you can display more than 16 colors on your screens and windows. This word has a second meaning, it refers to the work surface of the Amiga that displays icons and directories as well as windows and the ability to click on icons in those windows. Depending on which icons or windows are clicked, certain commands are executed. The operating system that comes with your Amiga is called Workbench.

Zorro Bus Add-on cards can only be installed in a computer that has free slots. Slots are part of a bus system. Depending on which computer you are talking about, these slots have a specific name. For the Amiga line of computers, we are talking about a Zorro II or Zorro III bus systems. Amiga 3000s and 4000s have a Zorro III bus that can also accept Zorro II cards. The Picasso II is a Zorro II card and can be installed into all current Amigas with a Zorro bus.

Picasso II

Seriel # 931200240